

UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

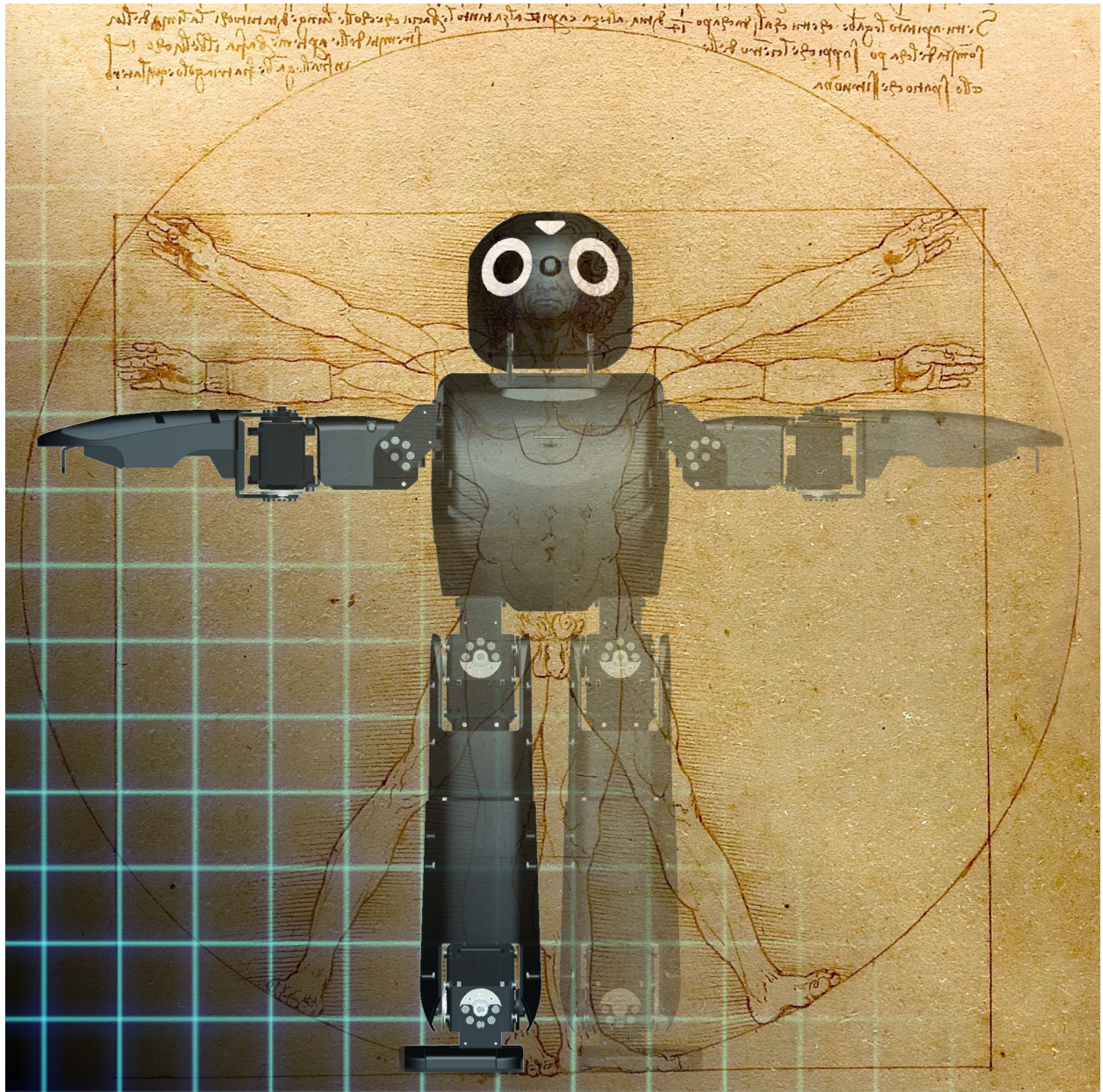
DEPARTMENT OF COMPUTER SCIENCE

Rocking the Bongo Board: Humanoid Robotic Balancing on Dynamic Terrain

by
Christopher James
IVERACH-BRERETON



A thesis submitted to the FACULTY OF GRADUATE STUDIES of
the UNIVERSITY OF MANITOBA in partial fulfilment of the
requirements of the degree of MASTER OF SCIENCE
Copyright ©2015 Christopher James Iverach-Brereton



There are an endless number of things to discover about robotics.
A lot of it is just too fantastic for people to believe.

— Daniel H. Wilson

Abstract

This thesis presents a comparison of multiple control algorithms designed to allow a robot to balance on unstable terrain. To evaluate these algorithms I use a bongo board, a simple apparatus consisting of a deck positioned above a free-rolling wheel. I program a small humanoid robot to stand on the deck, controlling its pose to keep the deck from falling off the wheel.

I implement three different control algorithms, derived from solutions to the well-known cart-and-rod inverted pendulum problem, including PID control, Fuzzy Logic, and Always-On Artificial Neural Networks. These algorithms are used with two different control policies: Do the Shake, where the robot reacts to external forces and Let's Sway, where the robot introduces a rhythmic oscillation to the system to promote dynamic stability.

Using identical experimental conditions with a physical robot, I show that both PID and Fuzzy Logic control are well-suited to active balancing on unstable terrain.

*This thesis is dedicated to the memory of my grandparents, John and Peggy
Iverach, and Donald and Brenda Brereton.*

I wish you could have been here to share this accomplishment with me.

Acknowledgements

I would like to thank everyone who helped contribute to this work. I have had the good fortune to find myself surrounded by people able and willing to provide technical advice and suggestions, moral support, and assistance with editing and proofreading. A heartfelt thank-you to you all.

I would like to officially acknowledge Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Manitoba for their financial support. Without their funding I may not have been able to return to school to pursue my Master's degree.

Thanks also to Drs Jacky Baltes, my supervisor, and John Anderson who, together, run the University of Manitoba's Autonomous Agents Laboratory where this research was conducted. Without Jacky's motivation and encouragement, as well as his knowledge and technical advice, this research would have taken me much longer to accomplish.

Over the last several years I have had the pleasure to work on humanoid robots with a long list of fellow students and researchers. I would especially like to thank my team-mates from the FIRA HuroCup and RoboCup competitions: Chi-Tai Cheng and Derek Cormier (FIRA 2011); Diana Carrier and Simon Barber-Dueck (FIRA 2012); Joshua Jung (without whom we would not have become world champions at FIRA 2013); Brittany Postnikoff (FIRA 2014); and Geoff Nagy, Andrew Winton and Amirhossein Hosseinmemar (RoboCup 2013-14). I hope future students in the lab are able to continue our success.

Finally I would like to thank my girlfriend Diana Carrier, my siblings Ben and Jennifer Iverach-Brereton, and my parents Bill Brereton and Lorraine Iverach for their hours of patiently listening to me as I talked about inverted pendula, their endless encouragement, and their moral support.

Without all of you behind me I would not be where I am today.



The robots Jimmy (left) and Jeff, with the trophies and medals we won at FIRA 2013 in Malaysia.

Contents

1	Introduction	1
2	Related Work	6
2.1	Inverted Pendula	6
2.1.1	Equations of Motion of Inverted Pendula	8
2.1.2	Humanoids as Inverted Pendula	12
2.2	Control Systems	13
2.2.1	PID Controllers	14
2.2.2	Fuzzy Logic	16
2.2.3	Yamakawa's Fuzzy Controller	19
2.2.4	Artificial Neural Networks	21
2.3	Forward and Inverse Kinematics	23
2.4	Active Balancing	26
3	Analysis of the Bongo Board as an Inverted Pendulum	29
3.1	Components of the Bongo Board System	29
3.2	Behaviour of a Bongo Board	31
3.2.1	Passive Rider	34
3.2.2	Active Rider	35
3.3	Comparison of Human and Robot Riders	47
3.4	Preliminary Research	52
3.4.1	Stiff Upper Lip	53
3.4.2	Do the Shake	54
3.4.3	Let's Sway	54
3.4.4	Predictive Sensor Model	55
4	Experimental Design	60
4.1	Degrees of Freedom Used	61
4.2	PID Control	64
4.3	Fuzzy Logic Controller	65
4.4	Always-On Artificial Neural Network	69
4.4.1	Input Layer	70
4.4.2	Output Layer	71
4.4.3	Hidden Layer	73
4.4.4	Training	74
5	Results and Evaluation	76
5.1	Observations, Self-Recovery, and Catapulting	78
5.2	Sensor Analysis	79
5.2.1	Comparison of Control Algorithms	91
5.2.2	Comparison of Control Policies	93

5.3	Failure Rate	97
5.4	AOANN Failure Analysis	98
5.4.1	PID-800 and Fuz-800	98
5.4.2	PID-28 and Fuz-28	100
5.4.3	Evaluation of Online Learning	101
6	Applications, Future Work and Conclusions	105
6.1	AOANN Improvement	106
6.2	Expanded Fuzzy Logic Rules	106
6.3	Other Control Algorithms	107
6.4	Hardware and Framework Improvements	107
6.5	Improved Sensor Filtering	107
6.6	Alpine and Cross-Country Skiing	108
6.7	Ice Skating	109
6.8	Wobble Board	110
6.9	Combining with Foot Placement Strategies	111
6.10	Conclusion	112
	References	118
	Appendices	119
A	Deck Inclination During All Trials	119
A.1	Do the Shake Policy	119
A.2	Let's Sway Policy	140
B	Sensor Data From All Trials	161
B.1	Do the Shake Policy	161
B.2	Let's Sway Policy	182

List of Figures

1.1	HuroCup marathon and sprint	2
1.2	HuroCup Stepping Field	2
2.1	Classical and card-and-rod inverted pendula	7
2.2	An idealized inverted pendulum on a mobile cart	9
2.3	Free body diagrams of the cart and rod	10
2.4	Humanoid form as an inverted pendulum	12
2.5	PID control loop	14
2.6	Fuzzy set membership functions	17
2.7	Configuration of a fuzzy system	18
2.8	Yamakawa's control rule map for the cart's velocity	20
2.9	A three-layer artificial neural network	21
2.10	Humanoid robot's degrees of freedom	25
2.11	Bongo board vs see-saw	27
3.1	Humanoid robot on bongo board	29
3.2	Anatomy of a bongo board	30
3.3	Wooden bongo board	32
3.4	A bongo board in the stable position	33
3.5	A bongo board in an unstable position	33
3.6	Anatomical planes	36
3.7	DoFs used by the rider	37
3.8	Raising and lowering the rider's CoM	38
3.9	Moving the rider's CoM with the arms	39
3.10	Rider's arm detail	40
3.11	Idealised bongo board with rider	42
3.12	Motion of the rider's arm	43
3.13	Vitruvian Man with robot	48
3.14	Dimensions of a DARwIn-OP robot	51
3.15	A human riding a bongo board	53
3.16	Raw gyroscope data	57
3.17	Error in inclination prediction	58
3.18	Error in angular velocity prediction	59
4.1	Software flowchart	60
4.2	Initial pose of the robot	62
4.3	Configuration of PID controllers	65
4.4	Arm control map	67
4.5	Lateral hip translation control map	67
4.6	Vertical hip translation control map	68
4.7	Torso inclination control map	68
5.1	Number of failures recorded during each trial	79
5.2	Recorded deck inclination using PID and Fuzzy Logic	80
5.3	Recorded deck inclination using AOANNs	81

5.4	Recorded difference in deck inclination using PID and Fuzzy Logic	82
5.5	Recorded difference in deck inclination using AOANNs	83
5.6	Recorded torso inclination using PID and Fuzzy Logic	86
5.7	Recorded torso inclination using AOANNs	87
5.8	Recorded torso velocity using PID and Fuzzy Logic	88
5.9	Recorded torso velocity using AOANNs	89
5.10	Difference between deck inclination and torso inclination	92
5.11	Z-axis accelerometer data using Fuz-28	96
5.12	Leaning in an unstable position	99
5.13	Results of training on deck inclination	102
5.14	Deck inclination using PID-28 and Do the Shake policy	103
6.1	Jennifer demonstrating alpine skiing	109
6.2	Jennifer the hockey robot	110
6.3	Wobble board	111
A.1	Inclination with PID, Do the Shake during Trial 1	120
A.2	Angular velocity with PID, Do the Shake during Trial 1	120
A.3	Inclination with PID, Do the Shake during Trial 2	121
A.4	Angular velocity with PID, Do the Shake during Trial 2	121
A.5	Inclination with PID, Do the Shake during Trial 3	122
A.6	Angular velocity with PID, Do the Shake during Trial 3	122
A.7	Inclination with PID, Do the Shake during Trial 4	123
A.8	Angular velocity with PID, Do the Shake during Trial 4	123
A.9	Inclination with PID, Do the Shake during Trial 5	124
A.10	Angular velocity with PID, Do the Shake during Trial 5	124
A.11	Inclination with Fuzzy Logic, Do the Shake during Trial 1	125
A.12	Angular velocity with Fuzzy Logic, Do the Shake during Trial 1	125
A.13	Inclination with Fuzzy Logic, Do the Shake during Trial 2	126
A.14	Angular velocity with Fuzzy Logic, Do the Shake during Trial 2	126
A.15	Inclination with Fuzzy Logic, Do the Shake during Trial 3	127
A.16	Angular velocity with Fuzzy Logic, Do the Shake during Trial 3	127
A.17	Inclination with Fuzzy Logic, Do the Shake during Trial 4	128
A.18	Angular velocity with Fuzzy Logic, Do the Shake during Trial 4	128
A.19	Inclination with Fuzzy Logic, Do the Shake during Trial 5	129
A.20	Angular velocity with Fuzzy Logic, Do the Shake during Trial 5	129
A.21	Inclination with AOANN PID-28, Do the Shake during Trial 1	130
A.22	Angular velocity with AOANN PID-28, Do the Shake during Trial 1	130
A.23	Inclination with AOANN PID-28, Do the Shake during Trial 2	131
A.24	Angular velocity with AOANN PID-28, Do the Shake during Trial 2	131
A.25	Inclination with AOANN PID-28, Do the Shake during Trial 3	132
A.26	Angular velocity with AOANN PID-28, Do the Shake during Trial 3	132
A.27	Inclination with AOANN PID-28, Do the Shake during Trial 4	133
A.28	Angular velocity with AOANN PID-28, Do the Shake during Trial 4	133

A.29 Inclination with AOANN PID-28, Do the Shake during Trial 5	134
A.30 Angular velocity with AOANN PID-28, Do the Shake during Trial 5	134
A.31 Inclination with AOANN Fuz-28, Do the Shake during Trial 1	135
A.32 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 1	135
A.33 Inclination with AOANN Fuz-28, Do the Shake during Trial 2	136
A.34 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 2	136
A.35 Inclination with AOANN Fuz-28, Do the Shake during Trial 3	137
A.36 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 3	137
A.37 Inclination with AOANN Fuz-28, Do the Shake during Trial 4	138
A.38 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 4	138
A.39 Inclination with AOANN Fuz-28, Do the Shake during Trial 5	139
A.40 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 5	139
A.41 Inclination with PID, Let's Sway during Trial 1	141
A.42 Angular velocity with PID, Let's Sway during Trial 1	141
A.43 Inclination with PID, Let's Sway during Trial 2	142
A.44 Angular velocity with PID, Let's Sway during Trial 2	142
A.45 Inclination with PID, Let's Sway during Trial 3	143
A.46 Angular velocity with PID, Let's Sway during Trial 3	143
A.47 Inclination with PID, Let's Sway during Trial 4	144
A.48 Angular velocity with PID, Let's Sway during Trial 4	144
A.49 Inclination with PID, Let's Sway during Trial 5	145
A.50 Angular velocity with PID, Let's Sway during Trial 5	145
A.51 Inclination with Fuzzy Logic, Let's Sway during Trial 1	146
A.52 Angular velocity with Fuzzy Logic, Let's Sway during Trial 1	146
A.53 Inclination with Fuzzy Logic, Let's Sway during Trial 2	147
A.54 Angular velocity with Fuzzy Logic, Let's Sway during Trial 2	147
A.55 Inclination with Fuzzy Logic, Let's Sway during Trial 3	148
A.56 Angular velocity with Fuzzy Logic, Let's Sway during Trial 3	148
A.57 Inclination with Fuzzy Logic, Let's Sway during Trial 4	149
A.58 Angular velocity with Fuzzy Logic, Let's Sway during Trial 4	149
A.59 Inclination with Fuzzy Logic, Let's Sway during Trial 5	150
A.60 Angular velocity with Fuzzy Logic, Let's Sway during Trial 5	150
A.61 Inclination with AOANN PID-28, Let's Sway during Trial 1	151
A.62 Angular velocity with AOANN PID-28, Let's Sway during Trial 1	151
A.63 Inclination with AOANN PID-28, Let's Sway during Trial 2	152
A.64 Angular velocity with AOANN PID-28, Let's Sway during Trial 2	152
A.65 Inclination with AOANN PID-28, Let's Sway during Trial 3	153
A.66 Angular velocity with AOANN PID-28, Let's Sway during Trial 3	153
A.67 Inclination with AOANN PID-28, Let's Sway during Trial 4	154
A.68 Angular velocity with AOANN PID-28, Let's Sway during Trial 4	154
A.69 Inclination with AOANN PID-28, Let's Sway during Trial 5	155
A.70 Angular velocity with AOANN PID-28, Let's Sway during Trial 5	155

A.71	Inclination with AOANN Fuz-28, Let's Sway during Trial 1	156
A.72	Angular velocity with AOANN Fuz-28, Let's Sway during Trial 1	156
A.73	Inclination with AOANN Fuz-28, Let's Sway during Trial 2	157
A.74	Angular velocity with AOANN Fuz-28, Let's Sway during Trial 2	157
A.75	Inclination with AOANN Fuz-28, Let's Sway during Trial 3	158
A.76	Angular velocity with AOANN Fuz-28, Let's Sway during Trial 3	158
A.77	Inclination with AOANN Fuz-28, Let's Sway during Trial 4	159
A.78	Angular velocity with AOANN Fuz-28, Let's Sway during Trial 4	159
A.79	Inclination with AOANN Fuz-28, Let's Sway during Trial 5	160
A.80	Angular velocity with AOANN Fuz-28, Let's Sway during Trial 5	160
B.1	Inclination with PID, Do the Shake during Trial 1	162
B.2	Angular velocity with PID, Do the Shake during Trial 1	162
B.3	Inclination with PID, Do the Shake during Trial 2	163
B.4	Angular velocity with PID, Do the Shake during Trial 2	163
B.5	Inclination with PID, Do the Shake during Trial 3	164
B.6	Angular velocity with PID, Do the Shake during Trial 3	164
B.7	Inclination with PID, Do the Shake during Trial 4	165
B.8	Angular velocity with PID, Do the Shake during Trial 4	165
B.9	Inclination with PID, Do the Shake during Trial 5	166
B.10	Angular velocity with PID, Do the Shake during Trial 5	166
B.11	Inclination with Fuzzy Logic, Do the Shake during Trial 1	167
B.12	Angular velocity with Fuzzy Logic, Do the Shake during Trial 1	167
B.13	Inclination with Fuzzy Logic, Do the Shake during Trial 2	168
B.14	Angular velocity with Fuzzy Logic, Do the Shake during Trial 2	168
B.15	Inclination with Fuzzy Logic, Do the Shake during Trial 3	169
B.16	Angular velocity with Fuzzy Logic, Do the Shake during Trial 3	169
B.17	Inclination with Fuzzy Logic, Do the Shake during Trial 4	170
B.18	Angular velocity with Fuzzy Logic, Do the Shake during Trial 4	170
B.19	Inclination with Fuzzy Logic, Do the Shake during Trial 5	171
B.20	Angular velocity with Fuzzy Logic, Do the Shake during Trial 5	171
B.21	Inclination with AOANN PID-28, Do the Shake during Trial 1	172
B.22	Angular velocity with AOANN PID-28, Do the Shake during Trial 1	172
B.23	Inclination with AOANN PID-28, Do the Shake during Trial 2	173
B.24	Angular velocity with AOANN PID-28, Do the Shake during Trial 2	173
B.25	Inclination with AOANN PID-28, Do the Shake during Trial 3	174
B.26	Angular velocity with AOANN PID-28, Do the Shake during Trial 3	174
B.27	Inclination with AOANN PID-28, Do the Shake during Trial 4	175
B.28	Angular velocity with AOANN PID-28, Do the Shake during Trial 4	175
B.29	Inclination with AOANN PID-28, Do the Shake during Trial 5	176
B.30	Angular velocity with AOANN PID-28, Do the Shake during Trial 5	176
B.31	Inclination with AOANN Fuz-28, Do the Shake during Trial 1	177
B.32	Angular velocity with AOANN Fuz-28, Do the Shake during Trial 1	177

B.33 Inclination with AOANN Fuz-28, Do the Shake during Trial 2	178
B.34 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 2	178
B.35 Inclination with AOANN Fuz-28, Do the Shake during Trial 3	179
B.36 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 3	179
B.37 Inclination with AOANN Fuz-28, Do the Shake during Trial 4	180
B.38 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 4	180
B.39 Inclination with AOANN Fuz-28, Do the Shake during Trial 5	181
B.40 Angular velocity with AOANN Fuz-28, Do the Shake during Trial 5	181
B.41 Inclination with PID, Let's Sway during Trial 1	183
B.42 Angular velocity with PID, Let's Sway during Trial 1	183
B.43 Inclination with PID, Let's Sway during Trial 2	184
B.44 Angular velocity with PID, Let's Sway during Trial 2	184
B.45 Inclination with PID, Let's Sway during Trial 3	185
B.46 Angular velocity with PID, Let's Sway during Trial 3	185
B.47 Inclination with PID, Let's Sway during Trial 4	186
B.48 Angular velocity with PID, Let's Sway during Trial 4	186
B.49 Inclination with PID, Let's Sway during Trial 5	187
B.50 Angular velocity with PID, Let's Sway during Trial 5	187
B.51 Inclination with Fuzzy Logic, Let's Sway during Trial 1	188
B.52 Angular velocity with Fuzzy Logic, Let's Sway during Trial 1	188
B.53 Inclination with Fuzzy Logic, Let's Sway during Trial 2	189
B.54 Angular velocity with Fuzzy Logic, Let's Sway during Trial 2	189
B.55 Inclination with Fuzzy Logic, Let's Sway during Trial 3	190
B.56 Angular velocity with Fuzzy Logic, Let's Sway during Trial 3	190
B.57 Inclination with Fuzzy Logic, Let's Sway during Trial 4	191
B.58 Angular velocity with Fuzzy Logic, Let's Sway during Trial 4	191
B.59 Inclination with Fuzzy Logic, Let's Sway during Trial 5	192
B.60 Angular velocity with Fuzzy Logic, Let's Sway during Trial 5	192
B.61 Inclination with AOANN PID-28, Let's Sway during Trial 1	193
B.62 Angular velocity with AOANN PID-28, Let's Sway during Trial 1	193
B.63 Inclination with AOANN PID-28, Let's Sway during Trial 2	194
B.64 Angular velocity with AOANN PID-28, Let's Sway during Trial 2	194
B.65 Inclination with AOANN PID-28, Let's Sway during Trial 3	195
B.66 Angular velocity with AOANN PID-28, Let's Sway during Trial 3	195
B.67 Inclination with AOANN PID-28, Let's Sway during Trial 4	196
B.68 Angular velocity with AOANN PID-28, Let's Sway during Trial 4	196
B.69 Inclination with AOANN PID-28, Let's Sway during Trial 5	197
B.70 Angular velocity with AOANN PID-28, Let's Sway during Trial 5	197
B.71 Inclination with AOANN Fuz-28, Let's Sway during Trial 1	198
B.72 Angular velocity with AOANN Fuz-28, Let's Sway during Trial 1	198
B.73 Inclination with AOANN Fuz-28, Let's Sway during Trial 2	199
B.74 Angular velocity with AOANN Fuz-28, Let's Sway during Trial 2	199

B.75 Inclination with AOANN Fuz-28, Let's Sway during Trial 3	200
B.76 Angular velocity with AOANN Fuz-28, Let's Sway during Trial 3	200
B.77 Inclination with AOANN Fuz-28, Let's Sway during Trial 4	201
B.78 Angular velocity with AOANN Fuz-28, Let's Sway during Trial 4	201
B.79 Inclination with AOANN Fuz-28, Let's Sway during Trial 5	202
B.80 Angular velocity with AOANN Fuz-28, Let's Sway during Trial 5	202

List of Tables

2.1	Effects of changing PID gains	15
2.2	Abbreviations for linguistic rules	19
3.1	Average human body dimensions	49
3.2	Robot body dimensions	49
3.3	Human and robot body ratios	52
4.1	Degrees of freedom used by the rider	62
4.2	Initial pose of the robot	63
4.3	Fuzzy membership function thresholds	67
4.4	ANN input ranges	71
4.5	Maximum and minimum observed DoF changes	72
4.6	Number of neurons per output	73
4.7	Accuracy of ANNs after initial training	74
5.1	Raw results of the experiment	78
5.2	Min, max, and average deck inclinations and difference in inclination	84
5.3	Min, max, and average inclinations and angular velocities	90

List of Equations

2.1	Torque applied to a card-and-rod system	10
2.2	Horizontal force acting on a cart-and-rod system	11
2.3	Vertical force acting on a cart-and-rod system	11
2.4	Driving force on a cart-and-rod system	11
2.5	Yamakawa's first equation of motion	11
2.6	Linear approximation of Yamakawa's first equation of motion	12
2.7	Linear approximation of Yamakawa's second equation of motion	12
2.8	Piecewise membership function	16
2.9	Symmetric sigmoid activation function	22
2.10	Inverse kinematics for a 6-DoF leg	26
3.1	Lateral force exerted on the deck by a falling rider	34
3.2	Force exerted by gravity pulling the deck downhill	34
3.3	Lateral forces acting on a bongo board with passive rider	35
3.4	Torque applied to a bongo board with a passive rider	35
3.5	Lateral and vertical positions of the arm's CoM	39
3.6	Lateral position of the rider's CoM	40
3.7	Vertical position of the rider's CoM	40
3.8	Torque applied by rotating the shoulder	41
3.9	Torque applied by rotating the torso	41
3.10	Total torque applied by the rider	41
3.11	Tangential velocity given angular velocity	44
3.12	Vertical and horizontal components of the force exerted by the arm's CoM	44
3.13	Horizontal and vertical forces exerted by rotating the shoulder	44
3.14	Component force exerted by translating the hips	44
3.15	Horizontal and vertical forces exerted by translating the hips	45
3.16	Horizontal and vertical forces exerted by torso inclination	45
3.17	Lateral force exerted by the rider	46
3.18	Vertical force exerted by the rider	46
3.19	Forces exerted by the rider in the global coordinate system	46
3.20	Total force acting on a bongo board	46
3.21	Total torque acting on a bongo board	47
3.22	Stiff Upper Lip control law	53
3.23	Do the Shake control law	54
3.24	Let's Sway control law	55
3.25	Pendulum period	55
3.26	Linear approximation for sensor prediction	56
4.1	Expanded Do the Shake control law	64
4.2	Expanded Let's Sway control law	64

List of Algorithms

2.1	Fuzzy implication	18
2.2	Yamakawa's linguistic rules containing fuzziness	20
2.3	Back-propagation algorithm for training artificial neural networks	23
4.1	Linguistic rules for the bongo board	66

1 Introduction

This thesis describes the development and evaluation of three control systems and two control policies used to allow a small humanoid robot to balance on highly unstable terrain. In order for humanoid robots to become useful in uncontrolled environments they must be able to traverse terrain that may exhibit a wide range of surface properties. A robot may be expected to traverse terrain ranging from well-maintained, indoor environments such as carpeted office floors, to unmaintained, outdoor settings, such as icy sidewalks and forest paths, to highly-dangerous settings, such as rubble-strewn disaster areas.

Modern humanoid robots demonstrate little difficulty traversing flat ground with high traction (e.g. concrete, hardwood, thin carpet). At the annual FIRA¹ HuroCup² competitions humanoid robots have demonstrated a consistent improvement in the speed and reliability of their walking gaits. Between 2009 and 2013 they improved the world record time for the sprint event³ from 01:07.50 in 2009 to 00:25.50 in 2013 [7]. Robots' performance in the HuroCup marathon event⁴ have similarly improved from a time of 07:30.00 over 42.195m in 2007 to 13:24.39 over 120m in 2013. [7]

While modern humanoid robots are able to traverse flat ground, the ability to cross arbitrary terrain, such as icy paths and rubble fields, remains beyond the state of the art. Research toward this goal is ongoing, as demonstrated by the stepping field used in HuroCup lift-and-carry competition, shown in Figure 1.2, and the development of Jennifer, the world's first ice-skating humanoid robot [26, 27].

¹FIRA: Federation of International Robot-soccer Association

²HuroCup: HUMANoid RObot Cup

³In the HuroCup sprint event the robots must walk/run 3m forwards followed by 3m backwards

⁴In the HuroCup marathon event robots must follow a marked path on the ground from beginning to end. Robots are required to complete the marathon on a single battery. The marathon is normally done outdoors, local weather conditions permitting



Figure 1.1: Jimmy, the robot used in this research, competing in the 2013 FIRA HuroCup marathon (left) and sprint (right) competitions



Figure 1.2: A robot walking across a stepping field at the HuroCup competition. The field is divided into multiple elevations, each indicated with a different colour. (HuroCup [8])

Central to any walking task is balance; the bipedal form is a naturally unstable one when compared with other form-factors seen in nature. Insects, with six legs, can keep their Centre Of Mass (CoM) positioned over a tripod formed by three of their legs while moving the other three, maintaining a statically stable position at all times.

Humans, having only two legs, must constantly make fine adjustments with the muscles in their feet and legs to stay standing upright. When walking humans must balance on a single foot, requiring a high level of control to keep from falling over. When jogging or

running both feet are off the ground for a short period of time and the impact forces upon landing must be compensated for to avoid stumbling and falling. These same principles apply to humanoid robots.

In order to balance on unstable or low-friction ground a robot must rely on active balancing; it is insufficient to simply lock the motors in such a way that the robot’s centre of mass is positioned over the feet. Because the feet may slip, the robot must be able to react and shift its centre of mass at any time.

In order to study the problem of active balancing with a humanoid robot I selected the bongo board as a sample problem. The bongo board is a simple apparatus consisting of a single free-rolling wheel and a flat deck used by humans for strength and balance training. The bongo board offers a simple example of a highly dynamic, unstable system that requires active control to keep stable.

The bongo board features a smooth, high-traction surface for the rider to stand upon. This surface rests on a single cylindrical wheel, allowing the deck’s inclination to vary from moment-to-moment. Because the wheel is free to roll, the location of the pivot point may also vary. This necessitates a high degree of active control to keep the deck level and keep the rider’s CoM positioned above the wheel. Otherwise the deck will slide down along the wheel similar to a mass sliding down an inclined plane.

In this thesis I examine the dynamics of the bongo board and how it relates to the well-known cart-and-rod inverted pendulum problem. I adapt three different control algorithms that have been shown to be successful at balancing a cart-and-rod inverted pendulum system for use with a small humanoid robot standing on a bongo board: Proportional-Integral-Derivative (PID) controllers, Fuzzy Logic, and Always-On Artificial Neural Networks (AOANNs)⁵. In addition to the three control algorithms, I develop three different control policies for use with the bongo board problem: *Stiff Upper Lip*, where the robot

⁵A detailed description of these three control algorithms and why I chose them can be found in Section 2.2.

attempts to keep its torso stationary while using lower-body flexibility to control the bongo board; *Do The Shake*, where the robot’s motions are entirely controlled by the control algorithm; and *Let’s Sway*, where a regular oscillation is introduced to the legs in an attempt to introduce a dynamically stable motion to the system. These control algorithms and control policies are used in different combinations under identical experimental conditions to evaluate their performance compared to one-another.

The rest of this thesis is organised as follows:

Section 2 describes relevant background material including analysis of classic inverted pendulum problems (Section 2.1.1), applications of inverted pendula to humanoid robotics (Section 2.1.2), and an overview of the control algorithms implemented to control the robot used in the experiments (Section 2.2).

Section 3 describes my preliminary research on controlling a humanoid robot on a bongo board published in Baltes et al. [6]. The construction of the bongo board is summarised in Section 3.1. In Section 3.2 I describe the dynamics of the bongo board, including an analysis of the external forces (e.g. gravity) the bongo board system is subjected to and the forces the rider can exert to counteract the external forces. Section 3.4 summarises the three different control policies published in Baltes et al. [6] designed to allow a humanoid robot to control a bongo board system: the human-inspired Stiff Upper Lip policy, the Do the Shake policy, and the Let’s Sway policy.

In Section 4 I describe the implementation of the control algorithms and control policies used for the experiment and the procedure used when collecting data. Section 4.1 shows how I translate the Degrees Of Freedom (DoFs) used by humans on a bongo board into equivalent DoFs on a small humanoid robot. The implementation of the PID controller is explained in Section 4.2. The fuzzy rule set and implementation of the Fuzzy Logic controller is given in Section 4.3. Finally, the design of the Artificial Neural Networks (ANNs) and the procedure used for training them is provided in Section 4.4.

In Section 5 I analyse the data collected during the experimental trials. I begin the analysis with qualitative observations in Section 5.1, followed by a quantitative analysis of the failure rate of each combination of control algorithm and control policy in Section 5.3. In Section 5.2 I analyse the sensor data recorded during the trials and evaluate the ability of each control algorithm and control policy to keep the bongo board in a stable position. Lastly I analyse the failure of the AOANNs in Section 5.4 to determine why their performance was so poor relative to the Fuzzy Logic and PID controllers.

Finally, future work and applications of this research are presented in Section 6.

2 Related Work

Balancing is a major topic of research in the area of humanoid robotics. The humanoid form naturally forms an inverted pendulum; the point where the feet contact the ground forms a fulcrum, with the body's mass being supported above it. Such an arrangement is naturally unstable, necessitating the ability to control the motion of the mass. In humanoid robots such control is normally done by controlling the Zero Moment Point (ZMP). [26, 27, 29]

The remainder of this section is organised as follows:

Section 2.1 defines the inverted pendulum problem. A mathematical analysis of the forces at work in the cart-and-rod variation of the inverted pendulum is found in Section 2.1.1. Section 2.1.2 describes how the inverted pendulum is used when simulating the behaviour of humanoid robots.

Section 2.2 summarises the three different control systems used in this research to control the bongo board: PID control (Section 2.2.1), Fuzzy Logic (Section 2.2.2), and Artificial Neural Networks (Section 2.2.4). Section 2.2.3 outlines an implementation of a Fuzzy Logic controller for the cart-and-rod inverted pendulum that formed the basis of my Fuzzy Logic system for controlling the bongo board.

Section 2.3 defines the problems of forward and inverse kinematics, concepts essential for controlling the motion of a robot's limbs.

Finally Section 2.4 describes prior research in the area of active balancing for humanoid robots on variable terrain, including standing on a see-saw and push-recovery techniques.

2.1 Inverted Pendula

An inverted pendulum is a dynamic system in which a mass m is connected to a fulcrum by a rod of negligible mass and length l , arranged such that the mass is located above the fulcrum, as shown in Figure 2.1a. As gravity acts on the mass the pendulum will naturally

tend to fall from its unstable position applying torque τ to the system. [41]

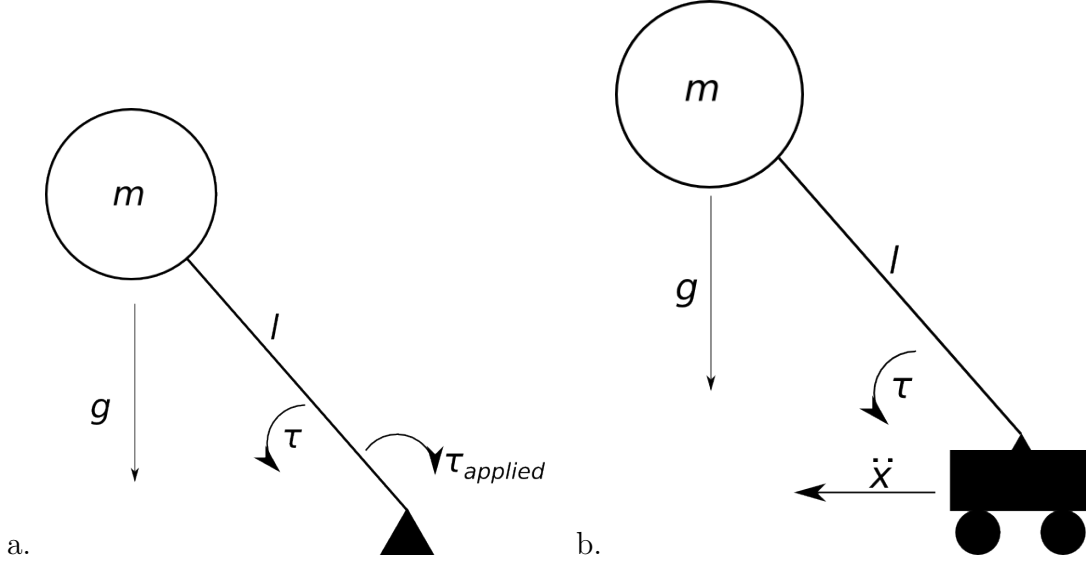


Figure 2.1: A classic inverted pendulum (left) and the cart-and-rod variant (right). A mass m is placed at the end of a rod of length l and positioned with the mass above the fulcrum. Gravity, g , acts on the mass, applying torque τ to the system. In the classic variant torque τ_{applied} is applied directly to the pendulum to force it back upright. In the card-and-rod variant a cart drives forward and backward with acceleration ($\ddot{x}$), indirectly applying torque to the pendulum and keeping the fulcrum positioned below the mass.

In the traditional inverted pendulum the system is controlled by applying torque τ_{applied} directly to the fulcrum, spinning the rod and mass back toward its vertical position. This can be accomplished in several ways, including placing an electric motor within the fulcrum itself.

The bongo board problem is a variation of the cart-and-rod problem [6]. Rather than the cart's movement controlling the motion of a passive mass, an active humanoid rider controls the motion of a passive wheel below them.

The cart-and-rod variant of the inverted pendulum, shown in Figure 2.1b – also known as the Carriage Balanced Inverted Pendulum [41] – replaces the applied torque with a mobile, powered cart containing the fulcrum. The position of the cart is continually adjusted by driving forward and backwards with speed $\dot{x}$ and acceleration $\ddot{x}$ to keep the fulcrum positioned

below the mass.

2.1.1 Equations of Motion of Inverted Pendula

Sultan and Mirza use the following setup for their mathematical analysis of the cart-and-rod problem [41]:

“The inverted pendulum is mounted on a moving cart. A servomotor is controlling the translation motion of the cart [...]. A rotary-potentiometer is used to feedback the angular motion of the pendulum to servo electronics to generate [the] actuating-signal.

“Controller circuits process the error signal, which then drives the cart through the servomotor [...]. To-or/and-fro motion of the cart applies moments on the inverted pendulum and thus it keeps the pendulum upright.” [41]

Note that the system is considered to be in equilibrium along the vertical axis; we assume any excess forces along the y -axis are cancelled out by the normal force⁶.

Given an inverted pendulum on a cart, as shown in Figure 2.2, we can define the following:

- the cart’s position along the x -axis, x ;
- the pendulum’s angle of inclination, θ ;
- the length of the rod, l ;
- the CoM of the pendulum, m ;
- the mass of the cart, M ;
- the force of gravity acting on the mass, mg ;

⁶Any mass under the effect of gravity exerts a downward force equal to $m \cdot g$. The ground exerts an equal and opposite force, called the normal force. These forces cancel each other out, allowing an object to remain stationary on the ground.

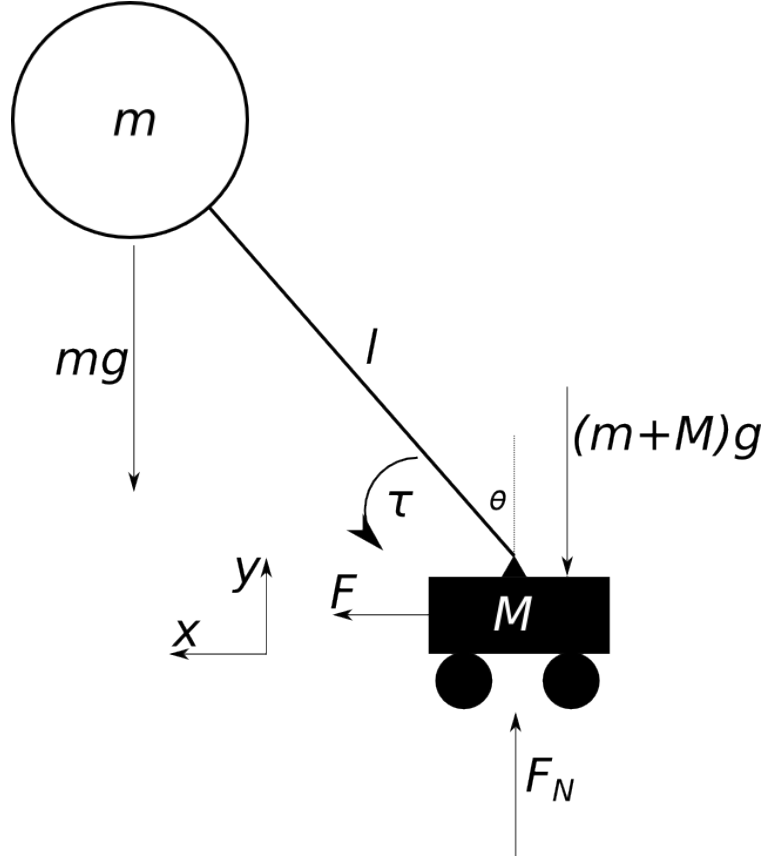


Figure 2.2: An idealized inverted pendulum on a mobile cart.

- the force of gravity acting on the entire system, $(m + M)g$;
- the normal force, F_N ; and
- the control force exerted by the cart on the system, $F = \ddot{x}(m + M)$.

Yamakawa [51] provides equations for a mathematical model of a cart-and-rod system. His system uses the cart-and-rod variant, where the only forces exerted on the system are:

- gravity pulling down on the mass and
- the cart accelerating forwards and backwards along the ground.

Yamakawa decomposes the forces on the system into their horizontal (F_H) and vertical (F_V) components. The torque on the system, τ , is given by the moment of inertia ($I = ml^2$)

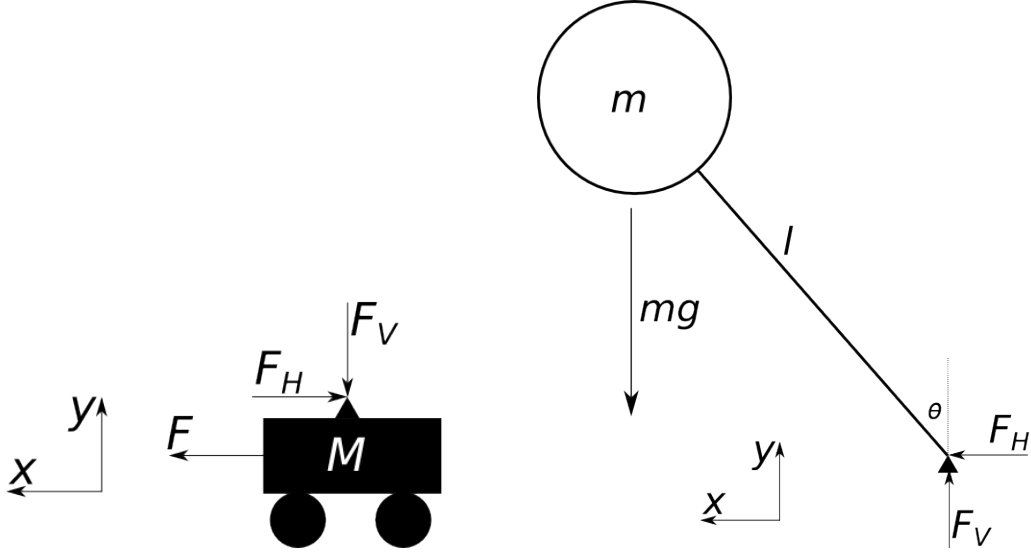


Figure 2.3: Free body diagrams of the cart (left) and rod(right) used in Yamakawa's analysis

multiplied by the angular acceleration about the fulcrum, $\ddot{\theta}$:

$$\tau = ml^2\ddot{\theta} = F_V l \sin \theta - F_H l \cos \theta \quad (2.1)$$

The falling pendulum causes vertical and horizontal forces to be applied to the cart. The magnitude of these forces is given by the inclination of the rod, the mass, m , and the angular acceleration of the mass.

The horizontal force, F_H , is derived from the cart's forward acceleration, $\ddot{x}$, multiplied by the mass, m , plus the horizontal force exerted by the rod pivoting about the fulcrum. The horizontal force exerted by the falling mass is equal to the mass multiplied by the length of the rod, l , multiplied by the angular acceleration of the rod times the cosine of the rod's inclination.

$$F_H = m\ddot{x} + ml(\ddot{\theta} \cos \theta - \dot{\theta}^2 \sin \theta) \quad (2.2)$$

The vertical force, F_V , is similarly derived. The vertical force is equal to the force of gravity pulling down on the mass, mg minus the vertical force exerted by the rod as it pivots about the fulcrum. This vertical force is equal to ml multiplied by the rod's angular velocity, multiplied by the sine of the rod's inclination.

$$F_V = mg - ml(\ddot{\theta} \sin \theta + \dot{\theta}^2 \cos \theta) \quad (2.3)$$

Finally, the driving force of the cart, F , acts parallel to F_H , and is given by the acceleration of the cart multiplied by the mass of the cart plus the the mass of the rod, denoted as $M' = m + M$.

$$F - F_H = M'\ddot{x} \quad (2.4)$$

Substituting in F_H we get the full equation for F :

$$\begin{aligned} F &= M'\ddot{x} + F_H \\ &= (M' + m)\ddot{x} + ml(\ddot{\theta} \cos \theta - \dot{\theta}^2 \sin \theta) \end{aligned} \quad (2.5)$$

Equations 2.1 to 2.2 contain non-linear elements. Because the goal of the system is to keep the pendulum upright ($\theta \approx 0$) with minimal angular velocity ($\dot{\theta} \approx 0$), we can approximate these to linear equations about the equilibrium point – the point at which the torque applied by gravity is zero – by assuming $\sin \theta \approx 0$ and $\cos \theta \approx 1$. This allows us to condense the above equations into the following linear equations: [51]

$$F \approx (m + M)\ddot{x} + ml\ddot{\theta} \quad (2.6)$$

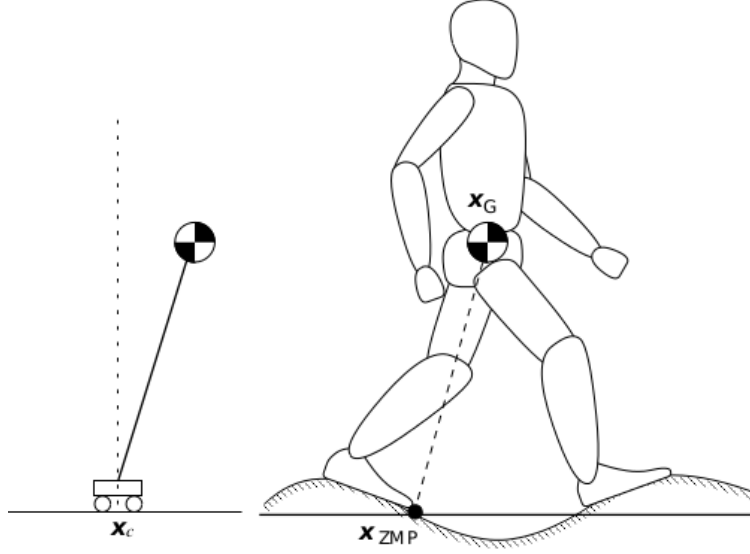


Figure 2.4: An inverted pendulum (left) and a humanoid system (right), showing the fulcrum, rod, and mass of the pendulum. (Sugihara et al. [40])

$$0 \approx (I + ml^2)\ddot{\theta} + ml\ddot{x} - mgl\theta \quad (2.7)$$

This technique of approximating a linear solution is also used by Bugeja [10]. Linear equations have the advantage of being easier to solve, while only sacrificing minimal accuracy, provided that the system behaves properly and stays near the equilibrium point.

2.1.2 Humanoids as Inverted Pendula

The humanoid form is easily modelled as an inverted pendulum [26, 27, 29]; the CoM is contained in the torso and the legs form the rod, with the fulcrum located at the ZMP, as shown in Figure 2.4. The linear inverted pendulum has been shown to be effective at generating dynamically stable humanoid walking gaits. [29, 40]

2.2 Control Systems

Many solutions have been used for balancing inverted pendulum systems, including Proportional-Integral-Derivative (PID) controllers [10], fuzzy logic controllers [51], and various machine learning algorithms including reinforcement learning [17, 19] and artificial neural networks [1].

For this research I chose to focus on three solutions to the inverted pendulum problem: PID control, Fuzzy Logic, and Artificial Neural Networks. PID is one of the most widely-referenced control schemes; Ziegler and Nichols' 1942 paper on automatic controllers [54] is a very widely-cited paper on controlling unstable systems (4317 citations as of the time of writing). PID controllers have been shown to be able to control a wide range of processes ranging from inverted pendula [10] to tuning engine idle speeds [20]. For these reasons PID control was selected as one of the three control algorithms to test on the bongo board system.

Yamakawa's fuzzy logic system was chosen as the basis for the second bongo board control algorithm because his paper [51] was one of the most widely-cited papers on the subject I could find (193 citations as of the time of writing). Yamakawa's solution is also very simple, consisting of only seven rules and seven fuzzy input and output sets, making it relatively simple to adapt for use with the bongo board. Yamakawa's mathematical analysis of the cart-and-rod problem was also instrumental to my analysis of the forces acting on the bongo board system.

The third control algorithm I selected is the Always-On Artificial Neural Network (AOANN), first proposed by Anderson [1]. Given the well-known nature of both PID and Fuzzy Logic control I felt that the inclusion of a lesser-known, slightly more novel solution was warranted. The bongo board, like the cart-and-rod problem, is a highly-dynamic problem that may be well-suited to classification by an artificial neural network. Furthermore, the fact that the AOANN is continuously adjusting itself as it controls the bongo board means that, unlike the PID and Fuzzy Logic controllers, the AOANN should improve its performance as the trials go on.

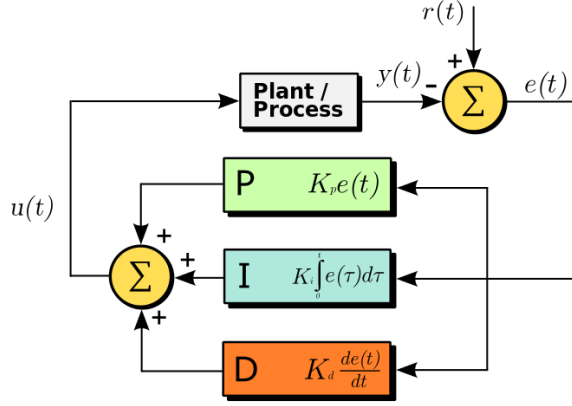


Figure 2.5: The PID control loop. (Wikimedia Commons [48])

2.2.1 PID Controllers

PID controllers are one of the most ubiquitous control systems currently in use [45]; a 1989 study found that over 90% of industrial process controllers in Japan used PID controllers [3]. Servo motors manufactured by Korea’s Robotis, such as the MX-28 motors used in the DARwIn-OP robot, use a PID controller to control the position of the actuator [34, 35]. PID controllers have been demonstrated to be effective at balancing inverted pendulums [10, 45].

PID controllers are designed to adjust a process variable, $y(t)$, towards a dynamic set-point, $r(t)$. To accomplish this they use the control loop shown in Figure 2.5. This control loop can be described as follows:

- The error in the system, $e(t)$, is equal to $r(t) - y(t)$ at any given time t .
- The manipulated variable, u , is changed by the plant/process to produce $y(t)$. The nature of this change is application-dependant. For example, in the case of the cart-and-rod inverted pendulum, the manipulated variable, u , might be the voltage applied to the cart’s drive motor to control the cart’s velocity, $y(t)$.
- Three constants, k_p , k_i , and k_d are used when calculating $u(t)$. These constants are tuned to the system.

Response	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
k_p	Decrease	Increase	Small Change	Decrease	Degrade
k_i	Decrease	Increase	Increase	Eliminate	Degrade
k_d	Minor Change	Decrease	Decrease	No effect	Improve if k_d is small

Table 2.1: The effects of independently increasing each of the three gains in a PID controller. (Zhong [53])

- $u(t)$ is set to $k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt}$
 - $k_p e(t)$ represents the correction based on the current error in the system
 - $k_i \int_0^t e(\tau) d\tau$ represents the correction based on previous errors and corrects for long-term biases
 - $k_d \frac{de(t)}{dt}$ represents the correction based on previous error, preventing or dampening oscillations in the system

The three gain constants – k_p , k_i , and k_d – must be properly tuned for the PID controller to perform its function correctly. Changing each gain has a different impact on the system, as shown in Table 2.1: Several techniques have been used for tuning these gains, including:

1. manual tuning (including Ziegler-Nichols tuning) [53, 54] and
2. continuous online tuning using machine learning algorithms [20, 50].

Manual tuning requires that a human manually adjust the gains for the PID controller. This process can be laborious, and requires frequent trial and error before a stable configuration is found. The Ziegler-Nichols tuning method provides a good baseline which can be manually tuned [53, 54]. With the Ziegler-Nichols method k_i and k_d are initially set to zero and k_p is gradually increased from zero until it reaches k_u , where the system oscillates with a fixed amplitude. The period of the oscillation, T_u , and the value of k_u are used to determine the final values of the PID gains:

- $k_p = 0.6k_u$
- $k_i = \frac{2k_p}{T_u}$
- $k_d = \frac{k_p T_u}{8}$

2.2.2 Fuzzy Logic

Fuzzy logic systems, unlike most traditional control methods, rely on the concept of fuzzy set membership. For all the inputs and outputs to the system we define fuzzy sets to which a given value is assigned a membership in the range $[0, 1]$. A membership of 0 indicates that the value is wholly outside the fuzzy set, while a membership of 1 indicates that the value is wholly inside the fuzzy set. Any other membership indicates that the value is partially within the fuzzy set. Equation 2.8 shows an example of a piecewise function used to establish the membership of a value, given three constants:

- x_{min} , the minimum value at which a variable has non-zero membership;
- x_{max} , the maximum value at which a variable has non-zero membership
- x_{peak} , the point at which a value has a membership of exactly 1 (often the average of x_{min} and x_{max})

$$m(x) = \begin{cases} 0 & |x < x_{min} \\ (x_{peak} - x)/(x_{peak} - x_{min}) & |x \in [x_{min}, x_{peak}) \\ 1 & |x = x_{peak} \\ 1 - (x - x_{peak})/(x_{max} - x_{peak}) & |x \in (x_{peak}, x_{max}] \\ 0 & |x > x_{max} \end{cases} \quad (2.8)$$

Membership is zero for all values less than x_{min} . The membership value increases linearly from 0 to 1 between x_{min} and x_{peak} . The function decreases linearly from 1 back to zero

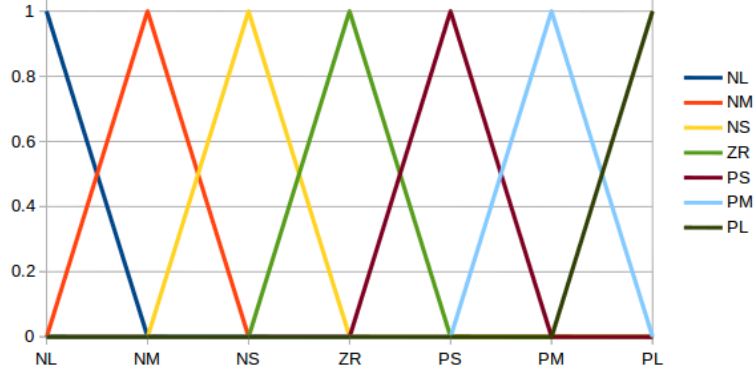


Figure 2.6: All input and output fuzzy sets use evenly-spaced, overlapping membership functions. The Negative-Large set (NL) is 1 for all values less than the NL threshold. The Positive-Large (PL) set is 1 for all values greater than the PL threshold.

between x_{peak} and x_{max} . Finally membership remains at zero for all values greater than x_{max} .

In the case of a control system for an inverted pendulum we can define fuzzy sets representing positive- and negative-large, -medium, and -small sets, as well as a near-zero set indicating if a given input or output should be considered large, medium, small, or near-zero for subsequent calculations. Membership in multiple sets is allowed, as the membership functions for each fuzzy set overlap one-another. Figure 2.6 shows the overlapping fuzzy sets used in one implementation of a fuzzy logic system to control a cart-and-rod inverted pendulum. [51]

Fuzzy logic controllers are frequently used in situations where there are no accurate mathematical models of the system being controlled, but human experts are able to provide linguistic rules – “fuzzy descriptions” – of the system’s behaviour. These fuzzy descriptions define the input and output sets of the system, U and V respectively. Fuzzy logic controllers consist of a fuzzy rule base, a fuzzy inference engine, a fuzzifier, and a defuzzifier, as shown in Figure 2.7. [46]

The linguistic rules provided by human experts fall into two main categories:

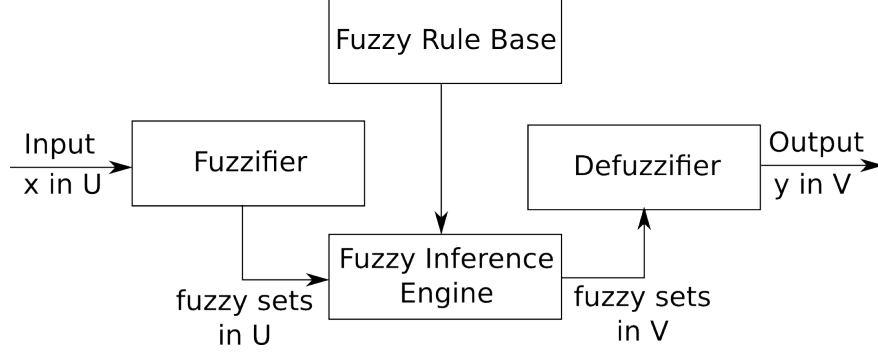


Figure 2.7: Configuration of a fuzzy system. (Wang [46])

1. fuzzy control rules which dictate what actions the system should undertake under certain conditions and
2. fuzzy if-then rules which describe how the system behaves when actions are undertaken.

The first set of rules can be exemplified by the phrase “If the car is driving too fast then apply the brakes.” The second set of rules can be exemplified by the phrase “If force is applied to the brakes then the car will decelerate.” [46]

The fuzzy rule base consists of rules from the second category above; these rules define fuzzy implications, as shown in Algorithm 2.1:

Algorithm 2.1 Fuzzy implication rule. (Wang [46])

R^l :

if x_1 is F_1^l and ... and x_n is F_n^l **then** y is G^l

end if

where $x = (x_1, \dots, x_n)^T \in U$ and $y \in R$ are the inputs and outputs of the fuzzy system respectively. F_i^l and G_i^l are labels for fuzzy sets in U_i and R_i respectively. $l = 1, 2, \dots, m$ enumerates these fuzzy sets. Therefore each rule defines an implication $F_1^l \times \dots \times F_n^l \rightarrow G^l$, a fuzzy set defined in the product $U \times R$. [46]

The fuzzy inference engine maps fuzzy sets in U to fuzzy sets in R , using the implications in the fuzzy rule base and the compositional rule of inference. [46]

Label	Abbreviation
Positive Large	PL
Positive Medium	PM
Positive Small	PS
Approximately Zero	ZR
Negative Small	NS
Negative Medium	NM
Negative Large	NL
Negation	NG

Table 2.2: Labels and abbreviations used by Yamakawa’s fuzzy logic rules. (Yamakawa [51])

The fuzzifier converts a crisp input point $x \in U$ to a fuzzy set $A_x \in U$, while the defuzzifier converts a fuzzy set in R into a crisp point in R for output. Several techniques exist for producing a crisp output, including: centre-average and centre of gravity defuzzifiers.

The centre-average defuzzifier is the most common method of defuzzification [46]. In this technique the crisp output is chosen by using the centres of each output membership function and the maximum certainty of each of the conclusions represented with the fuzzy sets. The centre of gravity defuzzifier chooses the crisp output using the centre of mass of each implied fuzzy set.

2.2.3 Yamakawa’s Fuzzy Controller

Yamakawa developed a successful implementation of a fuzzy logic controller to balance a cart-and-rod inverted pendulum. His solution consists of seven rules that are used to control the voltage applied to an electric motor that drives the cart forwards and backwards. Yamakawa’s rules are described in Algorithm 2.2. As a defuzzifier Yamakawa uses the centre-average method. [51] Note that these rules do not contain any equations nor numerical values. In Yamakawa’s implementation these rules are distributed in a control map for $\dot{y}$, shown in Figure 2.8. The control map shows the fuzzy input sets for both input variables (θ and $\dot{\theta}$) along the top and left respectively. At the intersection of each row and column is the corresponding fuzzy output set as defined by the linguistic rules. For example, when $\dot{\theta}$ is PS

Algorithm 2.2 Yamakawa’s linguistic rules containing fuzziness. (Yamakawa [51])

if θ is PM and $\dot{\theta}$ is ZR then $\dot{x}$ is PM end if if θ is PS and $\dot{\theta}$ is PS then $\dot{x}$ is PS end if if θ is PS and $\dot{\theta}$ is NS then $\dot{x}$ is ZR end if if θ is NM and $\dot{\theta}$ is ZR then $\dot{x}$ is NM	end if if θ is NS and $\dot{\theta}$ is NS then $\dot{x}$ is NS end if if θ is NS and $\dot{\theta}$ is PS then $\dot{x}$ is ZR end if if θ is ZR and $\dot{\theta}$ is ZR then $\dot{x}$ is ZR end if
---	--

	θ						
$\dot{\theta}$	NL	NM	NS	ZR	PS	PM	PL
PL							
PM							
PS			ZR		PS		
ZR		NM		ZR		PM	
NS			NS		ZR		
NM							
NL							

Figure 2.8: Yamakawa’s control rule map for the cart’s forward velocity, $\dot{x}$ (Yamakawa [51])

and θ is NS we see the fuzzy output is ZR. Blank intersections indicate instances where the rules do not have a specific behaviour defined (e.g. when $\dot{\theta}$ is PM and θ is ZR).

Of interest is the fact that the diagonal of the control map is made up entirely of ZR outputs. These correspond to instances where the system is already stable (i.e. θ and $\dot{\theta}$ are both ZR), and instances when the rod is inclined in one direction but is rotating slowly towards the stable position (i.e. θ is PS and $\dot{\theta}$ is NS, or vice-versa).

Note too that Yamakawa’s system does not define behaviour for instances where the system is highly unstable (e.g. when θ or $\dot{\theta}$ is PL). This is because the purpose of the system is to maintain the cart-and-rod in a stable (or near-stable) position. If a situation arises where the inclination or angular velocity is large then the system has already failed

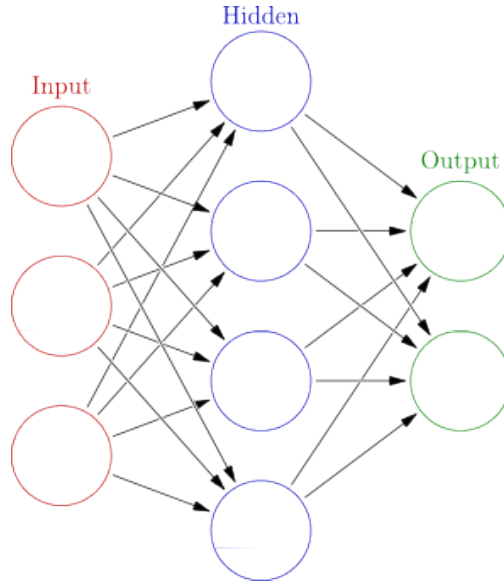


Figure 2.9: A three-layer artificial neural network. (Wikimedia Commons [49])

its stated purpose.

2.2.4 Artificial Neural Networks

Artificial Neural Networks (ANN) are computational models inspired by the behaviour of neurons in animal and human brains [52]. They generally consist of layers of artificial neurons – approximations of biological neurons implemented in software – which are interconnected, each neuron accepting inputs and producing outputs according to a mathematical function. Figure 2.9 shows an example of a three-layer ANN with three inputs and two outputs.

Each neuron has one or more inputs, $x_1, x_2, \dots, x_n$, each x_i assigned a weight k_i . The neuron calculates its output, y , using an activation function $f(x)$, where $x = \sum_{i=1}^n k_i x_i$. Typically $f(x)$ is a non-linear function, such as the symmetric sigmoid function used by the

open-source FANN (Fast Artificial Neural Networks) library [15], shown in Equation 2.9:

$$\begin{aligned}
y &\in (-1, 1) \\
y &= \tanh(s \cdot x) \\
&= \frac{1 - e^{-2sx}}{2e^{-2sx}} \\
d &= s(1 - (y^2))
\end{aligned} \tag{2.9}$$

where s and d are the steepness and derivation of the function. [15]

ANNs are an example of supervised learning [1]. Initially all weights are randomised and the ANN is provided a set of inputs and the correct outputs for these inputs as a training set. For some problems, such as the inverted pendulum, the correct action is not easily calculated for any given input state. Multiple courses of action may prevent the pendulum from falling. If the correct output is not available, Anderson proposes using reinforcement learning to evaluate the effect of the output on the system the ANN controls. [1]

Regardless of whether reinforcement or supervised learning techniques are used, the input weights of each neuron must be adjusted so that the output layer produces correct output. The most common technique for adjusting these weights is back-propagation. Pseudo-code for the back-propagation algorithm is shown in Algorithm 2.3. [5]

The back-propagation algorithm functions by calculating the error in each output neuron. The error for each neuron is calculated as the difference between its desired output (e.g. training goal) and its actual output. This error is propagated backwards through the ANN and the weight of each input is adjusted based on the magnitude of the error. In this way each weight is adjusted towards a local minimal error.

Algorithm 2.3 Back-propagation algorithm for training artificial neural networks

```
for all Neuron  $n$  in the ANN do
  for all Input weight  $w$  in  $n$  do
     $w \leftarrow$  some small, random, non-zero value
  end for
end for
for all Training example  $\langle (x_1, x_2, \dots, x_n), (t_1, t_2, \dots, t_m) \rangle$  do
  input  $(x_1, x_2, \dots, x_n)$  into the ANN and compute the network output  $y_k \forall k$ 
  for all Output unit  $k$  in the ANN do
     $\delta_k \leftarrow y_k(1 - y_k)(t_k - y_k)$ 
  end for
  for all Hidden unit  $h$  in the ANN do
     $\delta_h \leftarrow y_h \sum_k w_{k,h} \delta_k$ 
  end for
  for all Network weight  $w_{i,j}$  do
     $\Delta w_{i,j} \leftarrow \eta \delta_j x_{i,j}$ 
     $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$ 
  end for
end for
```

2.3 Forward and Inverse Kinematics

Motion planning for jointed robots (e.g. humanoid robots, robotic arms) can be largely split into two related problems: forward and inverse kinematics. Forward kinematics is defined as determining the locations of a robot's end-effectors (e.g. hands, feet) in three-dimensional space based on the the present angles of a robot's joints [38]. Inverse kinematics is defined as calculating the necessary joint angles such that an end effector can be placed in a specific location in three-dimensional space. [39]

Spong et al. [38] define forward kinematic equations for rigid robots. One can assume that the robot's limbs consist of n joints with a single degree of freedom separated by rigid members of known length. For the purposes of this thesis I make the assumption that all joints revolve about an axis, changing the angle θ between exactly two rigid members of the limb. Given the lengths of each rigid member and the angles between them one can compute the final position of the end-effector in three-dimensional space with respect to the limb's

origin using simple linear algebra.

General inverse kinematics are a much more complex problem than general forward kinematics; while forward kinematics will always provide a single unique solution given a robot's pose, inverse kinematics may provide multiple valid poses that will reach a given position. Spong et al. [39] propose using a simple geometric approach in which we define the origin of the coordinate system to be a fixed point attached to the robot. Typically this would be a point on the robot's shoulder in the case of a robotic arm, or on the hip in the case of a robot's leg, but can be any fixed location within the robot's body. From this origin we calculate the angles between the desired position of the end effector and each joint projected onto the x - y , x - z , and y - z planes. Using simple trigonometry we can determine the necessary joint angles for simple limbs. The advantage of this technique is that it is computationally simple, but it will not work for complex limbs with multiple spherical joints. This technique also allows for multiple solutions to be found, in which case a motion-planning algorithm needs a mechanism to choose one of the possible solutions.

Inverse kinematic equations can also be defined for a specific robot when necessary. The University of Manitoba's Autonomous Agents Laboratory [44] has developed inverse kinematics equations for the legs on the DARwIn-OP robots used in this thesis. The robot's leg consists of six motors: three in the hip (pitch, roll, yaw), one in the knee, and two in each ankle (pitch, roll). Given a point (x, y, z) in three dimensional space, the necessary

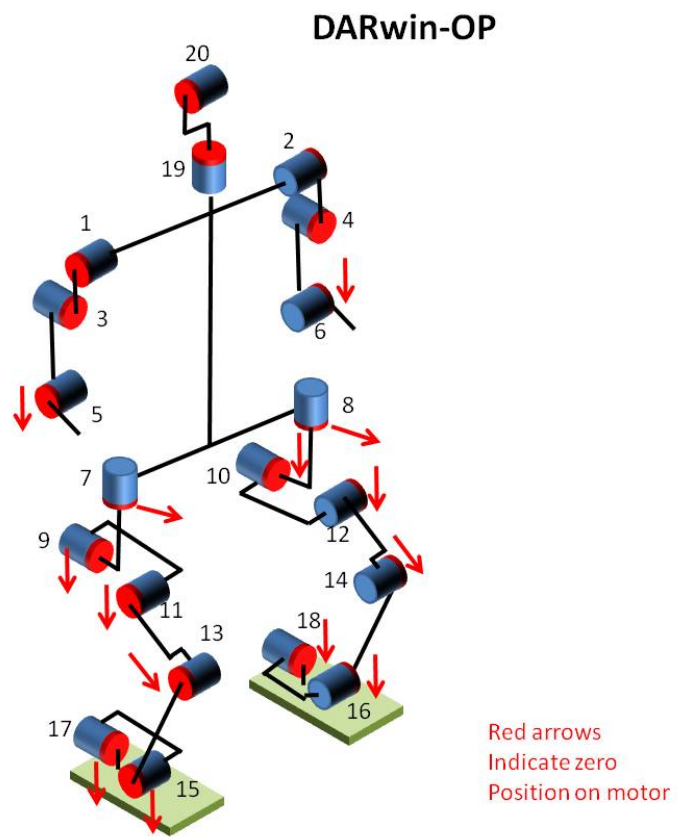


Figure 2.10: A humanoid robot with 20 degrees of freedom. (RoboSavvy [33])

joint angles are calculated as follows:

$$\begin{aligned}
r &= \sqrt{x^2 + y^2 + z^2} \\
\theta_x &= \tan^{-1}\left(\frac{x}{r}\right) \\
\theta_y &= \tan^{-1}\left(\frac{y}{r}\right) \\
\theta_{knee} &= \cos^{-1}\left(\frac{l_{thigh}^2 + l_{shin}^2 - r^2}{2 \cdot l_{thigh} \cdot l_{hip}}\right) \\
\theta_{roll} &= \sin(\theta_{yaw}) \cdot \theta_x + \cos(\theta_{yaw}) \cdot \theta_y + \sin(\theta_{yaw}) \cdot \sin(\theta_{knee}) \\
\theta_{pitch} &= \cos(\theta_{yaw}) \cdot \theta_x + \sin(\theta_{yaw}) \cdot \theta_y + \cos(\theta_{yaw}) \cdot \sin(\theta_{knee})
\end{aligned} \tag{2.10}$$

Note that the cosine law is used to calculate the angle of the knee.

For the right leg θ_{yaw} , θ_{roll} , θ_{pitch} , and θ_{knee} correspond to joint IDs 7, 9, 11, and 13 from Figure 2.10 respectively. For the left leg these angles correspond to joint IDs 8, 10, 12, and 14 from the same figure. θ_{yaw} is treated as a constant and is assumed to be zero if not otherwise specified. To keep the sole of the foot parallel to the ground – assuming the ground is level – the pitch and roll positions of the ankle are kept in-sync with those of the hip.

2.4 Active Balancing

Balancing is a major area of research for humanoid robotics. Because the humanoid form naturally forms an inverted pendulum [40] all humanoid robots require some method of balancing while standing and walking. As demonstrated at various robotics competitions, including the FIRA HuroCup and RoboCup events, modern humanoid robots have little difficulty traversing level terrain with good traction.

Relatively few examples of humanoid robots able to balance on dynamic terrain have been demonstrated in literature. McGrath et al. demonstrated a model-free balancing-reflex algorithm able to compensate for changes in ground inclination. Similarly, Anderson and

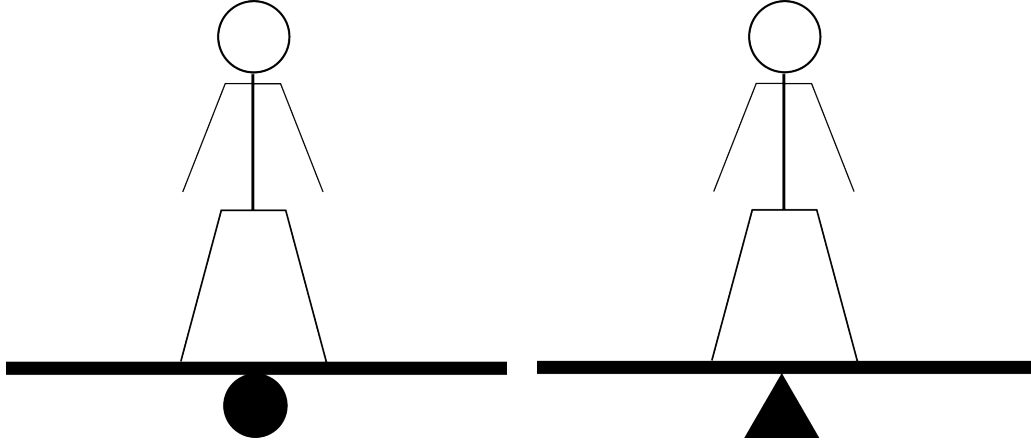


Figure 2.11: A bongo board with rider (left) and a see-saw with rider (right). The wheel of the bongo board is free to roll left-to-right along the deck of the bongo board, while the fulcrum of the see-saw is fixed.

Hodgins [2] work on the see-saw problem addresses the problem of balancing on a platform with a fixed pivot point.

The bongo board can be seen as a more unstable version of the see-saw problem. In the see-saw problem the rider stands on a deck anchored to the ground by a fixed fulcrum, as shown in Figure 2.11. The rider must move their body to control the motion of their CoM in order to keep the ends of the deck from striking the ground, as with the bongo board. Unlike the see-saw, the bongo board has no fixed fulcrum; the deck rotates around the point of contact between the wheel and the deck, as shown in Figure 2.11. Because the wheel is free to roll side-to-side the rider must also control the motion of the wheel in order for the system to remain stable.

Push-recovery strategies, such as the capture-point technique developed by Pratt et al. [32], have been used to assist in traversing unstable terrain and for compensating for disturbances when standing. However, these strategies rely on arresting the robot's movement by taking a step, relying on the ground-reaction force to decelerate the robot. In both the see-saw and bongo board problems the platform on which the robot stands is unsupported except for the region directly above the fulcrum. This means that the deck provides no

ground-reaction force, preventing foot-placement techniques from being effective at solving these problems.



Figure 3.1: Jimmy, a DARwIn-OP humanoid robot, on a bongo board. (Baltes et al. [6])

3 Analysis of the Bongo Board as an Inverted Pendulum

The bongo board can be viewed as a variation of the more traditional inverted pendulum problems [6]. While the inverted pendulum is a well-known problem, the bongo board is comparatively novel. No rigorous mathematical analysis of the bongo board has been done before. This section presents a detailed breakdown of the components of a bongo board in relation to the components of an inverted pendulum and explains the basic techniques used to control a bongo board.

3.1 Components of the Bongo Board System

The bongo board system is made up of three major components: the rider (which simultaneously forms the mass m and the rod of the pendulum), the deck (the platform on which

the rider stands), and the wheel. Figure 3.2 shows these three components. The wheel is not anchored to the deck in any way; it may freely roll along the entire length of the deck. The length of the rod, l , is determined by the height of the rider's CoM. We assume the rider's feet are securely anchored to the deck; the rider may not lift their feet off the deck, nor slide them along the deck's surface. The combination of the anchored rider and the deck shall be referred to as the rider-deck assembly.

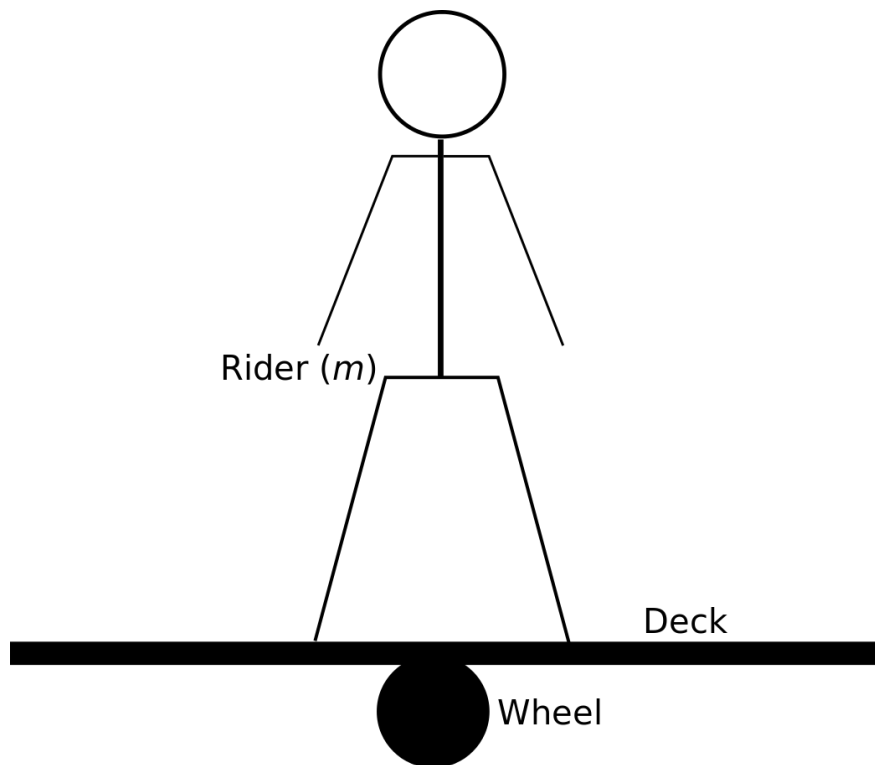


Figure 3.2: The three principal components that make up the bongo board: the rider, the deck, and the wheel. The wheel is not anchored to the board and is free to roll end-to-end.

A humanoid rider, such as the DARwIn-OP robot shown in Figure 3.1), used during my preliminary research [6] has many DoFs that can be used simultaneously to control the motion of the board; in its stock configuration the robot has 20 degrees of freedom [34], as shown in Figure 2.10. The robot used for this thesis is modified to include two single-DoF hands, bringing the total to 22 DoFs. These degrees of freedom are roughly analogous

to the range of motion in a human being; the robot can freely move its arms and legs independently of each other with a range of motion similar to that of a human. The robot lacks any flexibility in its spine however, which affects what motions the robot can use to balance (see Section 3.4).

Figure 3.3 shows the bongo board used during my experiments. The deck is made of a single piece of plywood with plywood struts underneath for stability and to provide a track for the wheel. The wheel is made of two circles of medium-density fibreboard glued to a wooden axle. Finally, the deck is covered in a sheet of high-traction foam rubber to help ensure that the rider's feet do not slide while balancing.

3.2 Behaviour of a Bongo Board

A bongo board is similar to the cart-and-rod variant of the inverted pendulum problem in that the position of the fulcrum may move independently of the position of the mass. In the stable position the rod is vertical, and the mass is centred directly above the wheel, as shown in Figure 3.4.

As the board is an unstable platform, it is necessary to define additional terms to describe its motion, as shown in Figure 3.5:

- the length of the rod (i.e. distance between the rider's CoM and the deck), l ;
- the mass of the rider, m ;
- the inclination of the rider relative to the deck, θ ;
- the inclination between the rider's CoM and the point of contact between the deck and the wheel, θ_m ;
- the distance between the point on the deck directly below the rider's CoM and the point of contact between the deck and the wheel, d ;

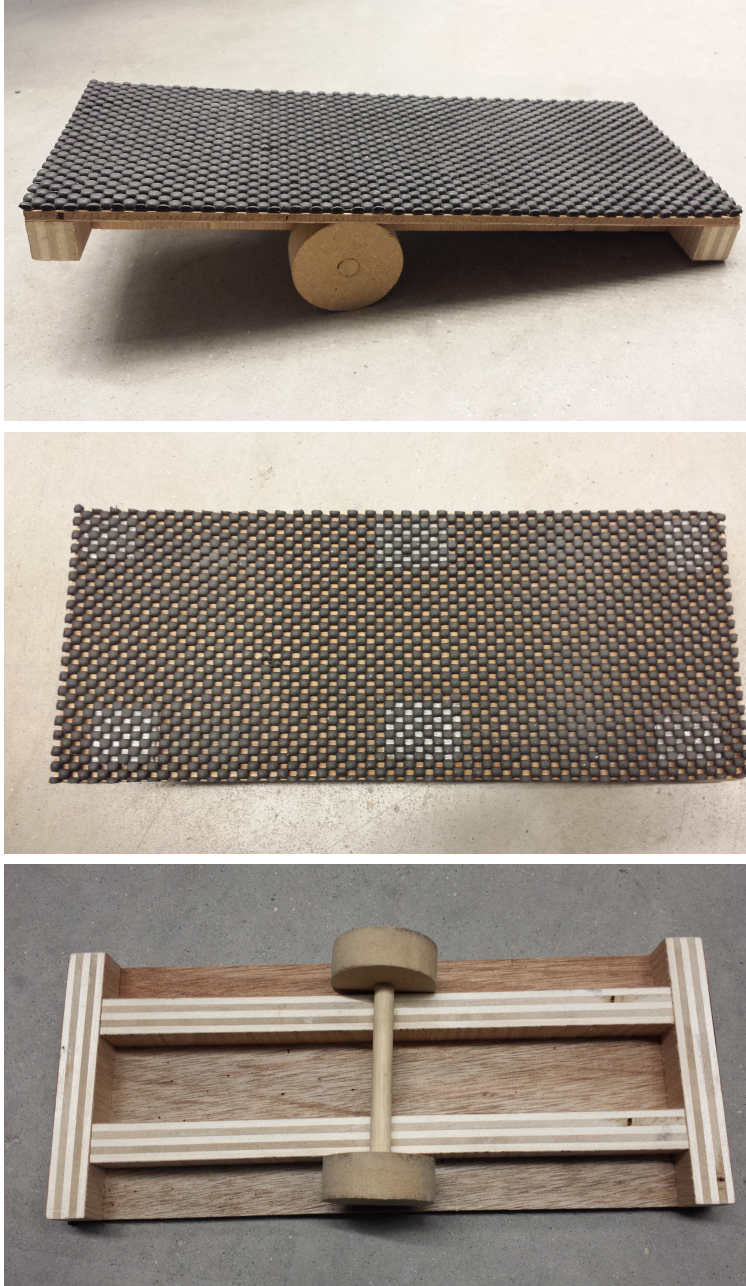


Figure 3.3: The bongo board used during the experiments. The deck is made of plywood covered in high-traction foam rubber with a wheel made of medium-density fibreboard.

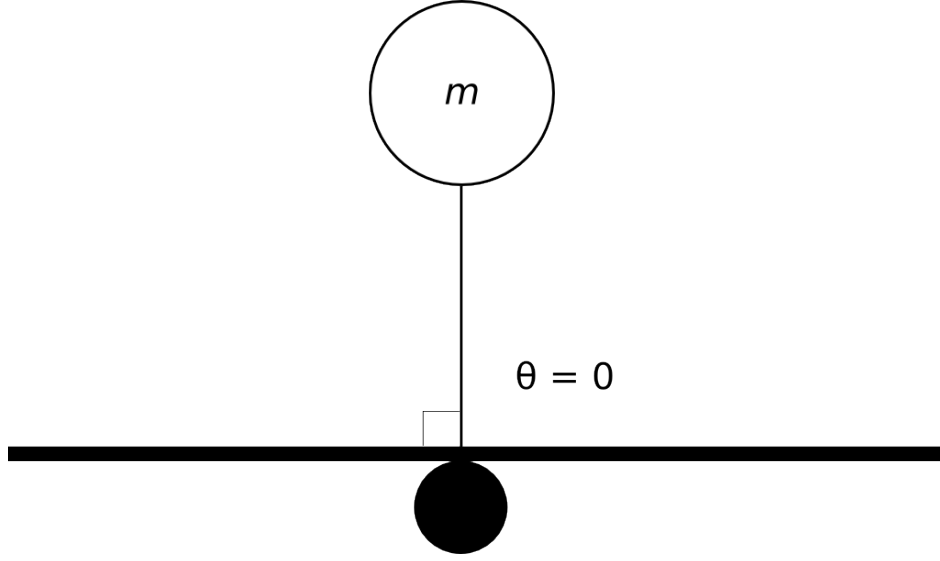


Figure 3.4: An idealised bongo board in a stable position. The mass is positioned directly above the wheel with an inclination, θ , of zero.

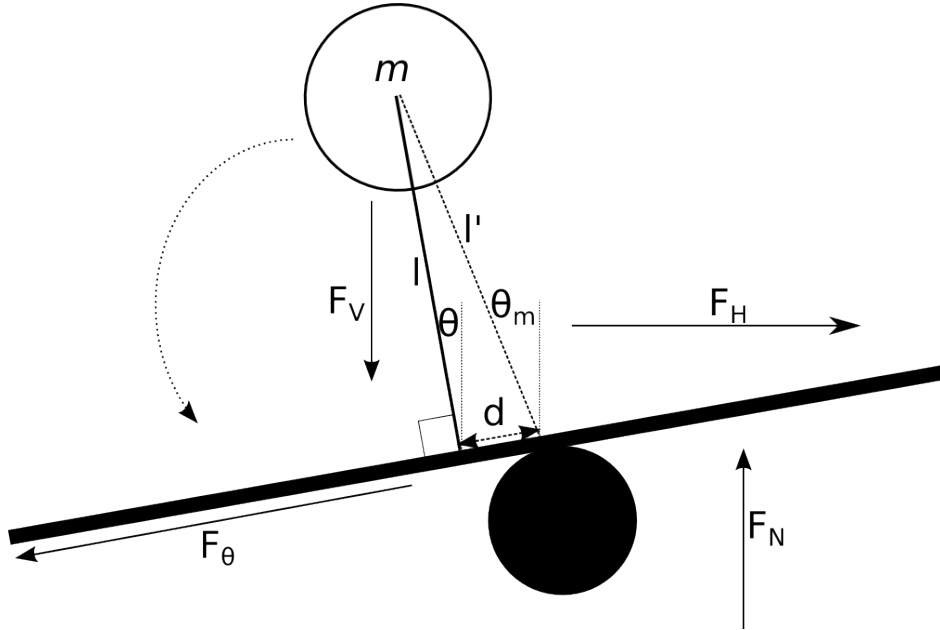


Figure 3.5: An idealised bongo board in an unstable position. θ shows the rider's inclination relative to the deck, while θ_m shows the rider's inclination relative to the point of contact between the wheel and the deck. θ_r gives the rider's torso's absolute inclination. The mass, m lies at height l above the deck. As the bongo board falls the mass rotates downward applying force F_α to the system, while the entire mass-rod-deck assembly slides downhill, applying F_θ to the system. l' gives the distance from the point of contact between the wheel and the deck to the rider's CoM.

- the distance from the point of contact between the wheel and the deck to m , l' ;
- the force applied to the system by the rotating mass, F_α ; and
- the force applied to the deck by the deck sliding downhill, F_θ .

If left alone the board will continue to fall, resulting from two phenomena:

1. the mass will continue to fall, rotating downward and applying lateral force to the deck, just as the falling mass applies F_H to the cart in the cart-and-rod inverted pendulum (see Section 2.1.1); and
2. the deck of the bongo board will slide downhill along the wheel, similar to a mass rolling down an inclined plane. This can alternatively be viewed as the wheel rolling toward the uphill end of the bongo board by changing the reference frame such that the centre of the deck remains fixed.

3.2.1 Passive Rider

We will assume that the deck is of negligible mass and that there is no friction between the deck and the wheel, nor between the wheel and the ground. Furthermore, we will assume that the rider is passive and exerting no forces on the system.

The two main forces acting on the system can be described using the following equations:

$$F_H = ml(\ddot{\theta} \cos \theta - \dot{\theta}^2 \sin \theta) \quad (3.1)$$

$$F_\theta = mg \cos \theta \quad (3.2)$$

Equation 3.1 is derived from Equation 2.5 and describes the horizontal force acting on the deck caused by gravity causing the mass to spin downward about the fulcrum. Note that because the deck-wheel assembly is passive the $m\ddot{x}$ term is omitted; the wheel exerts no force of its own on the system, unlike the cart-and-rod system described by Equation 2.5.

Equation 3.2 is derived from the standard equation for a mass sliding down an inclined plane under the force of gravity and describes the motion of the entire rider-deck-assembly as it rolls down the wheel under the effect of gravity.

From equations 3.1 and 3.2 we can trivially derive the total lateral force acting on the system:

$$\begin{aligned} F_{external} &= F_{\alpha} + F_{\theta} \sin \theta \\ &= -ml\ddot{\theta}_m \cos \theta_m + ml\dot{\theta}_m^2 \sin \theta_m + mg \cos \theta \sin \theta \end{aligned} \tag{3.3}$$

As the board rotates torque is applied to the system. This torque, τ , is given by the mass of the rider multiplied by the angular force applied to the point of contact between the deck and the wheel. Because the only force being exerted is due to gravity we calculate the torque as follows:

$$\tau_{external} = mgl' \sin \theta_m \tag{3.4}$$

where g is the acceleration due to gravity, approximately 9.81m/s^2 .

3.2.2 Active Rider

Removing the restriction of a passive rider requires analysis of the forces the rider can impart. Let us assume that the rider has 14 degrees of freedom, as shown in in Figure 3.7: the shoulders, hips and ankles can all rotate in both the frontal and sagittal planes while the knees rotate only in the sagittal plane. (These planes are shown in Figure 3.6.)

These degrees of freedom allow the rider to shift their CoM laterally and vertically, as

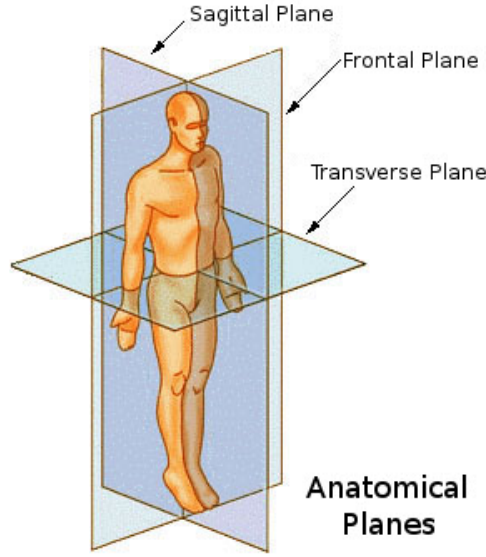


Figure 3.6: The three anatomical planes used when referring to the humanoid form. The frontal plane divides the body front-to-back, the sagittal plane divides the body side-to-side, and the transverse plane divides the body top-to-bottom. (Wikimedia Commons [47])

shown in Figure 3.8. By extending or contracting both legs simultaneously the rider may change the height of their CoM. The rider may also change the lateral position of their CoM by extending one leg and contracting the other while changing the roll angle of the hips and ankles. By extending one leg and contracting the other the rider may also incline their CoM to the left or right.

The rider may move their arms to provide fine control over the position of the CoM and to induce a moment in the rider-deck assembly, as shown in Figure 3.9. Let us define the mass of the rider's body without the arms as m_{torso} , and the mass of each arm as m_{arm} . Thus the mass of the entire rider is $m = 2m_{arm} + m_{torso}$. Let the angle of the left and right shoulders measured from vertical in the frontal plane be θ_L and θ_R respectively. θ_S gives the angle of the shoulder from vertical in the sagittal plane. We restrict θ_L and θ_R to lie in the range $[0, 180]^\circ$. We assume θ_S is zero.

From these definitions one can describe effect that moving the arms has on the motion of

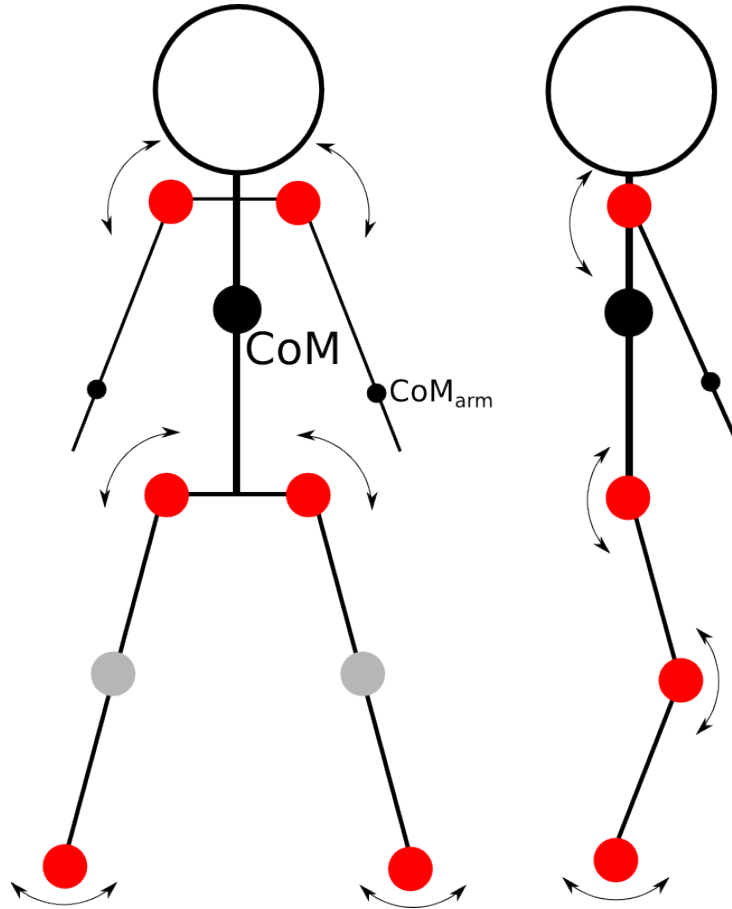


Figure 3.7: The degrees of freedom used by the rider on the bongo board that can act in the frontal plane (left) and sagittal plane (right). Note that the knees cannot move in the frontal plane, and are therefore marked in grey. The robot's CoM is located within the torso (marked with a large black circle). Each arm also has its own CoM (marked with a small black circle).

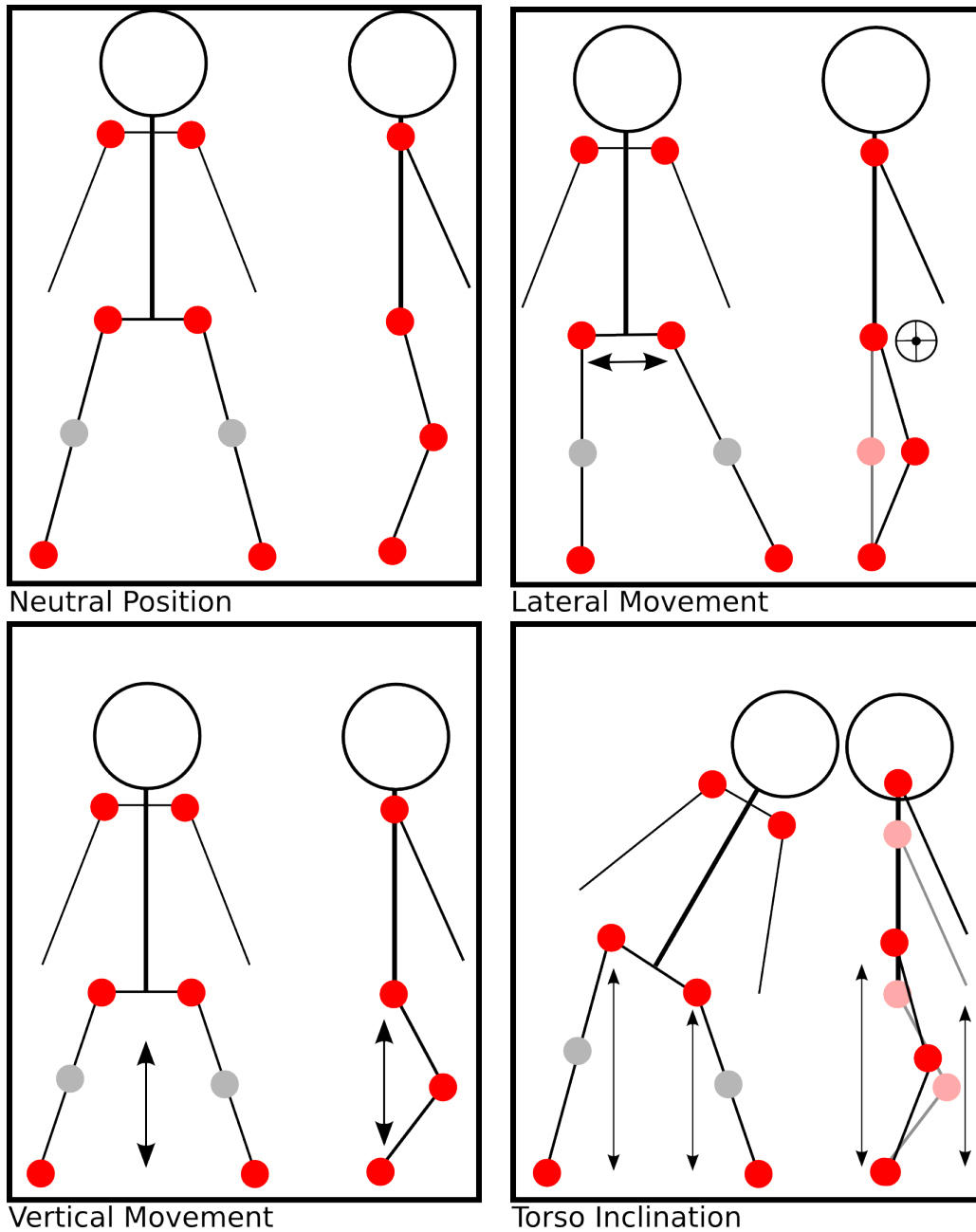


Figure 3.8: Diagrams showing how the rider in the neutral position (top left) and how it uses its degrees of freedom to raise/lower its CoM (bottom left), shift its CoM side-to-side (top right), and incline its CoM to either side (bottom right).

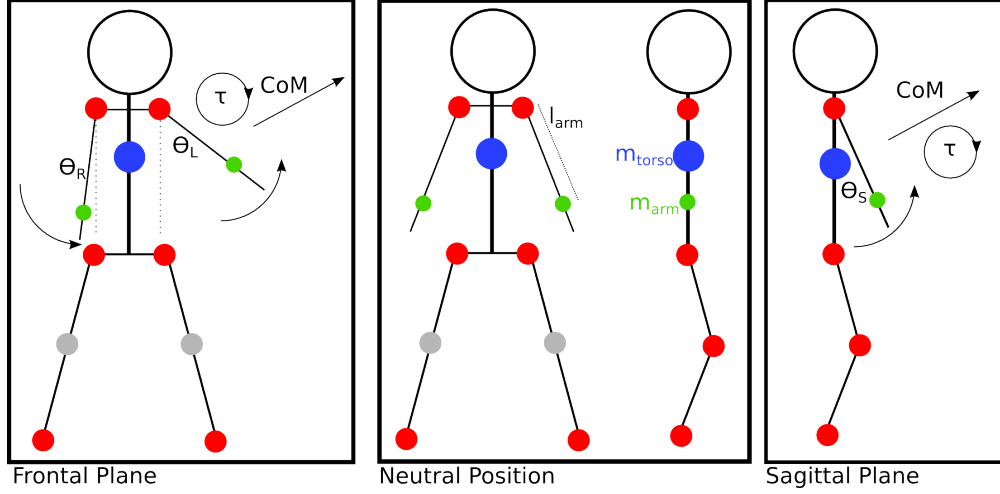


Figure 3.9: Moving the arms changes the vertical and lateral positions of the CoM, and induces a moment in the rider-deck assembly. The torso's CoM, marked in blue, remains fixed, regardless of the arm position. Each arm's centre of mass, marked in green, can be moved along the frontal and sagittal planes. As the arms move the rider's overall CoM moves, and torque τ is applied to the torso as a result of the movement. θ_S gives the angle of the arm in the sagittal plane, while θ_L and θ_R give the angles of the left and right arms in the frontal plane.

the rider's CoM: the position of the CoM is the average of the positions of the CoM of each arm and the torso, weighted by their respective masses. Because the bongo board rocks in the frontal plane we can ignore the effects of motion in the sagittal plane.

Figure 3.10 shows in detail how the shoulder joint connects with the torso. The shoulder lies h above the CoM, and d to the side. l_{arm} gives the distance from the shoulder joint to the arm's CoM. θ gives the angle of the arm with respect to the torso. Let us define the origin to be the CoM of the torso. Given this we can calculate the vertical and lateral position (x, y) of the arm's CoM as follows:

$$\begin{aligned} x &= l_{arm} \sin \theta + x_S \\ y &= y_S - l_{arm} \cos \theta \end{aligned} \tag{3.5}$$

Because we define the CoM of the torso to be the origin the rider's overall CoM is therefore

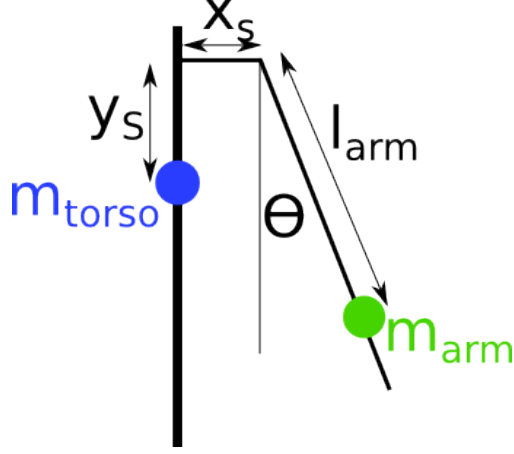


Figure 3.10: A detailed view of how the shoulder joint connects to the torso's CoM. The shoulder lies y_S above the torso's CoM and x_S to the side. The distance from the shoulder to the arm's CoM is given by l_{arm} . The shoulder's angle is given by θ .

determined entirely by the average of the CoM of the arms:

$$X_{CoM} = \frac{2m_{arm}(l_{arm} \sin \theta_L - l_{arm} \sin \theta_R)}{m_{torso} + 2m_{arm}} \quad (3.6)$$

$$Y_{CoM} = \frac{2m_{arm}(y_S - l_{arm} \cos \theta_L + y_S - l_{arm} \cos \theta_R)}{m_{torso} + 2m_{arm}} \quad (3.7)$$

Note that when calculating X_{CoM} the two arms are acting in opposite directions, so we subtract the effect of the right arm from that of the left. This causes the x_S terms to cancel each other out.

The torque applied to the system as a result of moving an arm is equal to the mass of the arm times the tangential force applied at the shoulder:

$$\tau_{arm} = l_{arm} \times F_{shoulder} \quad (3.8)$$

Similarly the torque applied by inclining the torso is equal to the mass of the entire rider multiplied by the tangential force applied to the torso:

$$\tau_{torso} = l \times F_{torso} \quad (3.9)$$

Summing the torques applied by each arm and the torso we can calculate the total torque applied to the system:

$$\tau_{exerted} = l \times F_{torso} + l_{arm} \times F_{leftShoulder} + l_{arm} \times F_{rightShoulder} \quad (3.10)$$

By moving the arms and adjusting the torso position the rider exerts vertical and horizontal forces on the system. The rider uses these forces to counteract the passive forces described in Equation 3.3. There are four motions the rider may exercise, each contributing a horizontal and vertical component:

1. rotating the arms about the shoulder,
2. translating the hips horizontally,
3. translating the hips vertically, and
4. inclining the torso to the left and right.

The following sections show the derivation of these four component forces. For these sections we define “horizontal” to mean “parallel to the bongo board’s deck,” and “vertical” as “perpendicular to the deck.” These definitions allow us to ignore the inclination of the deck of the board and analyse the forces the rider exerts on the system in isolation. Because

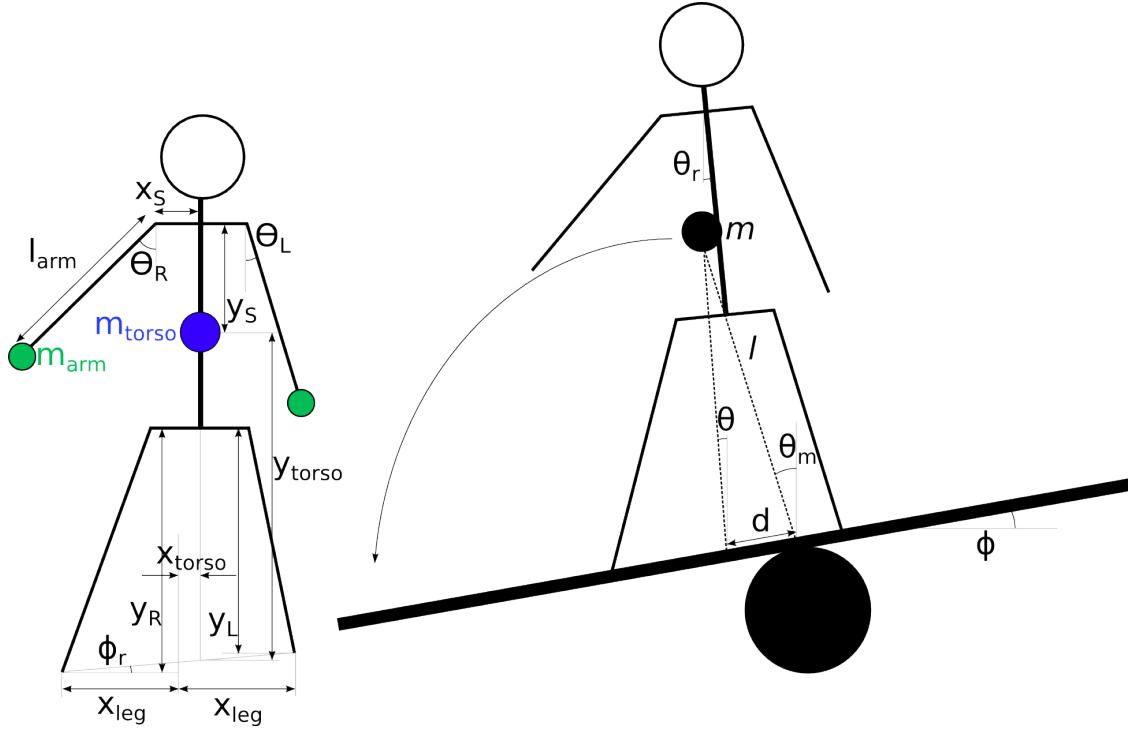


Figure 3.11: An idealised bongo board in an unstable position with the rider. The rider's torso is offset horizontally by x_{torso} and vertically by y_{torso} with respect to the midpoint between the robot's feet. The legs have been adjusted to lengths y_R and y_L for the right and left legs respectively. ϕ_r gives the inclination of the rider due to the difference in lengths of the legs. The arms have been set to angles θ_R and θ_L with respect to the right and left shoulders. The rider's CoM, m , is offset from the torso because of the arm positions. θ denotes the inclination of m with respect to the line drawn from the midpoint between the robot's feet. θ_m denotes the angle between m and the point of contact between the deck and the wheel. The distance from this point of contact and m is given by l . The deck is inclined by ϕ from the horizontal.

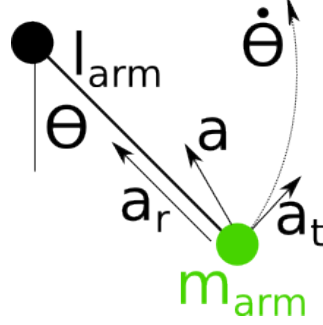


Figure 3.12: As the arm rotates about the shoulder the arm's CoM undergoes non-uniform circular motion, undergoing radial acceleration, a_r , and tangential acceleration, a_t . The total acceleration, a , is the sum of a_t and a_r .

the rider is inclined with angle ϕ_r relative to the deck, the terms $\sin \phi_r$ and $\cos \phi_r$ will occur frequently to account for this inclination.

Effects of Arm Rotation Figure 3.12 shows the motion of the arm as it rotates about the shoulder with angular velocity $\dot{\theta}$ and angular acceleration $\ddot{\theta}$. We treat the arm as a point-mass located at the arm's CoM with mass equal to the entire arm to simplify the following equations. Further, we do not allow the rider to extend nor contract the arm; the distance between the shoulder and the arm's CoM remains constant. From these assumptions we can say that the arm's CoM undergoes non-uniform circular motion. This implies that the arm's CoM simultaneously undergoes radial acceleration, a_r , toward the shoulder, and tangential acceleration, a_t , tangent to the path traced by the CoM. These forces act at right-angles to one another by definition. Summing these accelerations we get a , the total acceleration acting on the CoM.

The tangential velocity of a body undergoing circular motion is simply the radius of the circle, r , multiplied by the angular velocity of the body, $\dot{\theta}$. In the case of the arm $r = l_{\text{arm}}$.

$$v = r \cdot \dot{\theta} \quad (3.11)$$

The tangential acceleration is simply the derivative of v with respect to time. It can be approximated by simply taking the difference in angular velocities at two distinct times t_1 and t_2 divided by the elapsed time between them.

Let $F = m_{arm}a$ be the total force the arm's CoM exerts while under acceleration a . F can be decomposed into its horizontal and vertical components, X_{arm} and Y_{arm} respectively, assuming $\phi_r = 0$:

$$\begin{aligned} X_{arm} &= F \cdot \sin \theta \\ Y_{arm} &= F \cdot \cos \theta \end{aligned} \tag{3.12}$$

Allowing for non-zero ϕ_r yields H_{arm} and V_{arm} , the horizontal and vertical forces exerted by rotating the arm about the shoulder:

$$\begin{aligned} H_{arm} &= X_{arm} \cos \phi_r + Y_{arm} \sin \phi_r \\ V_{arm} &= X_{arm} \sin \phi_r + Y_{arm} \cos \phi_r \end{aligned} \tag{3.13}$$

Effects of Hip Translation As shown in Figure 3.8, the rider may easily translate their CoM both horizontally and vertically. By simply multiplying the mass of the rider by the acceleration along each axis we can find the force exerted along the horizontal and vertical axes, X_{hip} and Y_{hip} respectively, assuming $\phi_r = 0$:

$$\begin{aligned} X_{hip} &= (m_{torso} + 2m_{arm}) \cdot \ddot{x}_{torso} \\ Y_{hip} &= (m_{torso} + 2m_{arm}) \cdot \ddot{y}_{torso} \end{aligned} \tag{3.14}$$

Allowing for non-zero ϕ_r we get the following equations for the vertical and horizontal

forces exerted by translating the hips:

$$\begin{aligned} H_{hip} &= X_{hip} \cos \phi_r + Y_{hip} \sin \phi_r \\ V_{hip} &= X_{hip} \sin \phi_r + Y_{hip} \cos \phi_r \end{aligned} \tag{3.15}$$

Effects of Torso Inclination By extending one leg and/or contracting the other the rider may incline their CoM to the left or right. As shown in Figure 3.11, this inclination is ϕ_r . The analysis of torso inclination is similar to that of arm rotation; like the arm, the torso is undergoing non-uniform circular motion. We assume that the hip translation is negligible, giving us a constant radius with which the torso rotates. As with the arm, the torso simultaneously experiences tangential acceleration, a_t and radial acceleration, a_r , summing to a .

Let us define the following variables:

- ϕ_r , the inclination of the rider relative to the bongo board's deck;
- $\dot{\phi}_r$ and $\ddot{\phi}_r$, the rider's angular velocity and acceleration respectively; and
- l , the distance between the deck and the rider's CoM.

Given these variables and Equation 3.11 we can calculate the rider's tangential velocity, v . As with the arms, the rider's tangential acceleration is simply the derivative of v with respect to time. Finally, let $F = a \cdot (m_{torso} + 2m_{arm})$ be the force exerted by rotating the torso.

Decomposing F into its horizontal and vertical components we get

$$\begin{aligned} H_{torso} &= F \cdot \sin \phi_r \\ V_{torso} &= F \cdot \cos \phi_r \end{aligned} \tag{3.16}$$

Total Forces Exerted by the Rider Combining the equations for H_{arm} , V_{arm} , H_{hip} , V_{hip} , H_{torso} , and V_{torso} we can calculate the total forces acting along the rider's horizontal and vertical axes, F_h and F_v respectively:

$$F_h = H_{leftArm} + H_{rightArm} + H_{hip} + H_{torso} \quad (3.17)$$

$$F_v = V_{leftArm} + V_{rightArm} + V_{hip} + V_{torso} \quad (3.18)$$

Having derived F_h and F_v in the rider's coordinate system we must now translate these forces into the global coordinate system used to calculate $F_{external}$ in Equation 3.3, where the vertical axis is parallel to the direction of gravity. To do this we simply rotate the rider's coordinate system by the inclination of the bongo board's deck, ϕ :

$$\begin{aligned} F_x &= F_h \cos \phi + F_v \sin \phi \\ F_y &= F_h \sin \phi + F_v \cos \phi \end{aligned} \quad (3.19)$$

F_x is the force exerted by the rider acting parallel to $F_{external}$. We assume that F_y is counteracted by the normal force acting through the ground, and can therefore ignore it when examining the behaviour of the bongo board system. Let us call F_x $F_{exerted}$. Combining Equation 3.3 and 3.19 we can calculate the total force acting on the system, F :

$$\begin{aligned} F &= F_{external} + F_{exerted} \\ &= F_{external} + (F_h \cdot \cos \phi + F_v \cdot \sin \phi) \end{aligned} \quad (3.20)$$

The force exerted by the rider is in the opposite direction as the external forces applied to the system. Under ideal circumstances the force exerted by the rider should be equal to the external forces, resulting in a stable system with $F = 0$. Practically these forces deviate from one another, but the system should constrain this deviation such that the difference between them remains small, giving $F \approx 0$.

Finally, as given by equations 3.4 and 3.10, the total torque acting on the system is equal to the torque applied by gravity minus the torque applied by the rider moving its arms and torso:

$$\begin{aligned}\tau &= \tau_{external} - \tau_{exerted} \\ &= mgl' \sin \theta_m - l \times F_{torso} - l_{arm} \times F_{leftShoulder} - l_{arm} \times F_{rightShoulder}\end{aligned}\tag{3.21}$$

3.3 Comparison of Human and Robot Riders

Because the bongo board is a novel problem for a humanoid robot it is important to understand how the kinematics of a human being relate to the robot used in any experiments. Specifically, we must show that the mass-height ratios of the inverted pendula formed by a standing human and standing robot are comparable, and show that the effect of rotating the arms for stability is similar.

Anecdotally we can see that the proportions of the robot are similar to those of a human, though somewhat skewed. From Figure 3.13, showing a DARwIn-OP robot superimposed over Leonardo da Vinci’s “Vitruvian Man,” we can see that the robot’s legs are of a similar length, but the torso is shorter and broader. The robot’s shoulders, measured from the pivot point in the frontal plane, are significantly wider than Vitruvian Man’s, giving the robot a proportionally larger wingspan.

The effect of moving the arms will also be reduced as $l_{arm} \rightarrow 0$. Clauser et al. [11] note

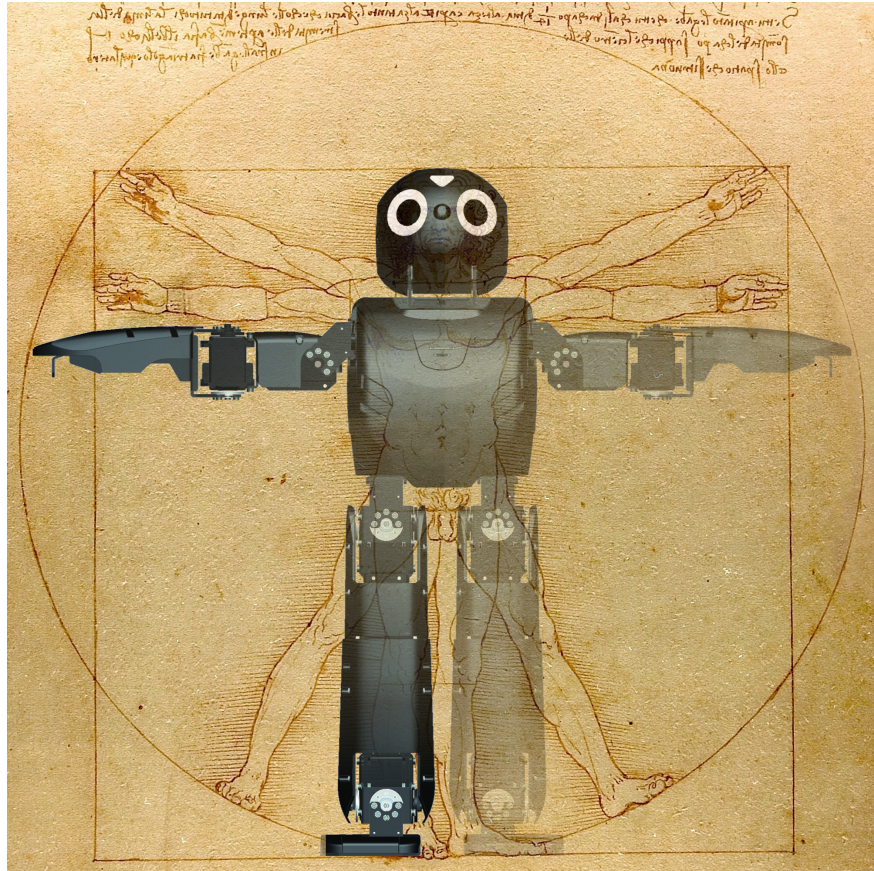


Figure 3.13: A scaled image of a DARwIn-OP robot overlaid with Leonardo da Vinci's "Vitruvian Man," showing the relative proportions of the robot with a human being. (da Vinci [13], Robotis [34])

Description	Variable	Value (kg,cm)
Total mass	m	65.6
Arm mass	m_{arm}	3.2
Height of CoM	l	101
Distance from shoulder to arm's CoM	l_{arm}	32
Vertical distance from torso's CoM to shoulder	y_S	48
Horizontal distance from torso's CoM to shoulder	x_S	16.5

Table 3.1: Lengths and masses of body segments from figures 3.5 and 3.10 for an average adult human male. (Clauser et al. [11])

Description	Variable	Value (kg,cm)
Total mass (including hand motors)	m	2.9
Arm mass (including hand motor)	m_{arm}	0.216
Height of CoM	l	26.0
Distance from shoulder to arm's CoM	l_{arm}	9.4
Vertical distance from torso's CoM to shoulder	y_S	8.15
Horizontal distance from torso's CoM to shoulder	x_S	8.2

Table 3.2: Lengths and masses of body segments from figures 3.5 and 3.10 for the DARwIn-OP robot used in this thesis.

that for an average adult human male the CoM of the arm lies approximately 32cm from the shoulder and the whole body's CoM lies approximately 71cm from the top of the head. Given an average total height of 172cm and a cervicale (shoulder-to-ground) height of 149cm and a chest breadth of 33cm we can calculate l from Figure 3.5 and l_{arm} , y_S and x_S from Figure 3.10 for a human, as shown in Table 3.1. From these we can estimate $\frac{l_{arm}}{l} \approx 0.32$, $\frac{y_S}{l} \approx 0.48$, and $\frac{x_S}{l} \approx 0.16$.

I measured the DARwIn-OP robot used in this research and determined that the robot's CoM lies approximately $\frac{2}{3}$ of the way from the shoulder to the hip, and the weight is approximately evenly distributed through the arm⁷. From the dimensions shown in Figure 3.14 I calculate l from Figure 3.5 and l_{arm} , y_S and x_S from Figure 3.10 for the robot, as shown in Table 3.2. From these we can estimate $\frac{l_{arm}}{l} \approx 0.36$, $\frac{y_S}{l} \approx 0.31$, and $\frac{x_S}{l} \approx 0.32$. With the

⁷The robot used during this research has a third motor in the hand, unlike the robot pictured in Figure 3.14. As this hand is an after-market addition, Robotis does not feature it in their schematics of the robot.

exceptions of $\frac{x_S}{l}$ and $\frac{y_S}{l}$ these values are consistent with those of an average human. The robot's shoulder width (x_S) is proportionally much larger than that of a human. However, because the x_S terms cancel out in Equation 3.6 this difference is insignificant. The height of the robot's shoulder (y_S) is also smaller than that of a human, owing largely to the robot's over-large head. The height of the shoulder has no bearing on the torque that can be applied by rotating the arms; the only result is that the CoM of each arm will be slightly closer to the ground than a human, meaning the bongo board's deck will rotate slightly more quickly when force is applied.

From Equation 3.8 we can see that as $m_{arm} \rightarrow 0$ the effect of moving the arms is reduced. According to Clauser et al. [11] the average mass for a human male's arm is approximately 3.2kg with an average total body mass of 65.6kg. From these estimates we can assume that, for an average human, $\frac{m_{arm}}{m} \approx 0.05$. Taking the extreme measurements recorded by Clauser et al. the arm makes up between 5 and 14% of the total body mass.

The humanoid robot used during this research has a total mass of 2.9kg [34]. Each arm consists of 3 motors with a total mass of approximately 216g [35], giving $\frac{m_{arm}}{m} \approx 0.07$. The arm making up 7% of the total mass of the body is slightly above what one would expect of an average human, but is within the normal range.

Comparing $\frac{m_{arm}}{l_{arm}}$ ratios for an average human and the robot we see that for the human $\frac{m_{arm}}{l_{arm}} \approx 0.1$ and for the robot $\frac{m_{arm}}{l_{arm}} \approx 0.02$. Clearly the robot's arm is significantly lighter for its length than a comparable human arm. This will require the robot to move its arms more quickly than a human would to apply the same torque to the system.

Similarly, comparing the $\frac{m}{l}$ ratios we see that for the human $\frac{m}{l} \approx 0.65$ and for the robot $\frac{m}{l} \approx 0.11$. Again, the robot's mass proportional to its size is significantly lower than what one would expect of an average human. This has the effect of lowering the torque generated by the deck of the board rotating, meaning the robot will need to exert less force to right itself.

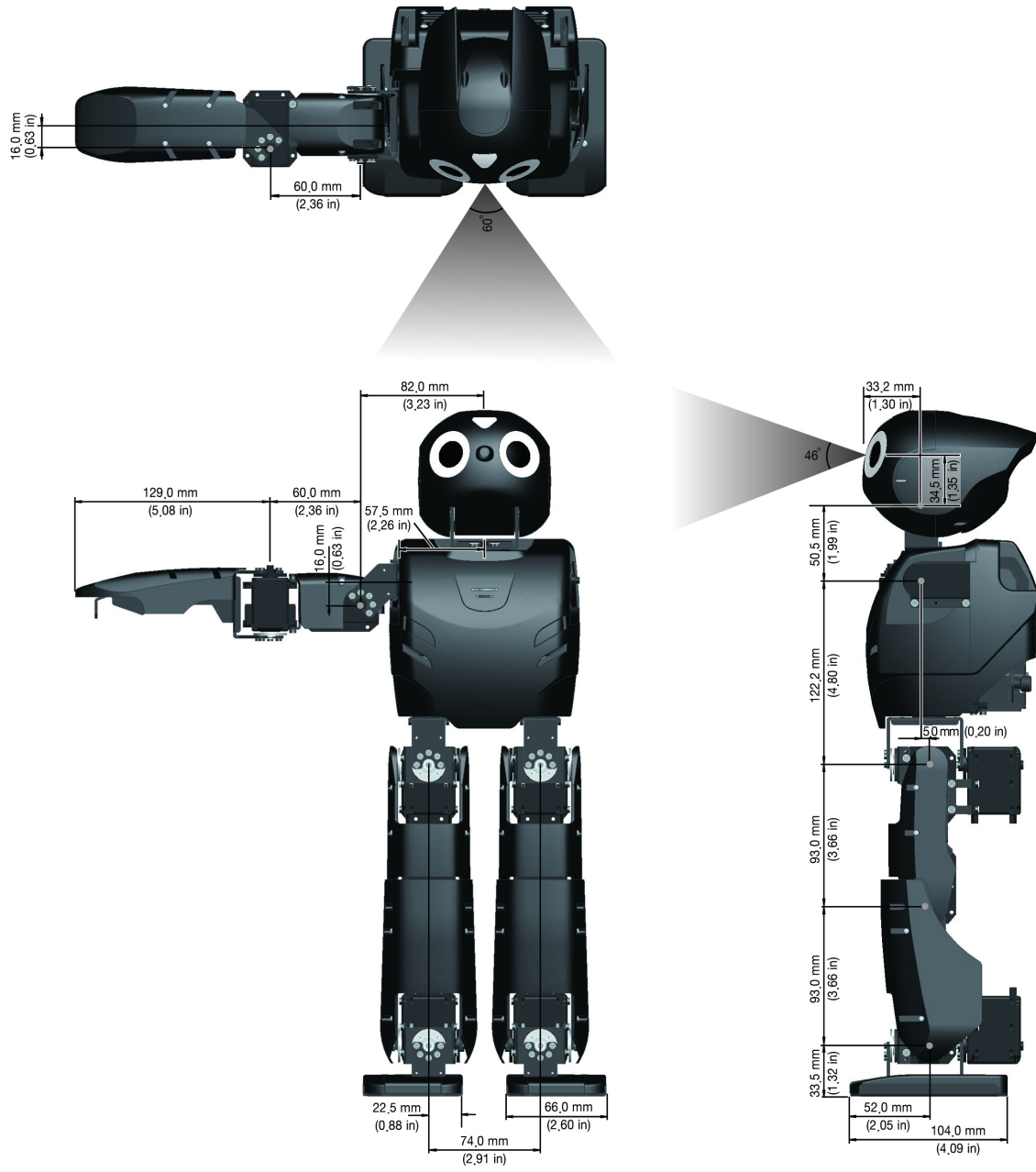


Figure 3.14: Physical dimensions of the DARwIn-OP robot used in this research. (Robotis [34])

Ratio Description	Formula	Human	Robot
Arm length : Height of CoM	$\frac{l_{arm}}{l}$	0.32	0.36
Shoulder height : Height of CoM	$\frac{y_S}{l}$	0.48	0.31
Shoulder width : Height of CoM	$\frac{x_S}{l}$	0.16	0.32
Arm mass : Total mass	$\frac{m_{arm}}{m}$	0.05	0.07
Arm mass : Arm length	$\frac{m_{arm}}{l_{arm}}$	0.1	0.02
Total mass : Height of CoM	$\frac{m}{l}$	0.65	0.11

Table 3.3: Ratios of the size and mass of body parts for an average human and a DARwIn-OP robot

3.4 Preliminary Research

During the spring of 2013 I did the majority of the programming on an initial bongo board balancing robot named Jimmy⁸. An overview of this work is published in Baltes et al. [6], and is summarised below.

This preliminary research was primarily intended for demonstration purposes at the 2013 DARwIn-OP Humanoid Application Challenge at the ICRA conference in Karlsruhe, Germany. As such we did not perform any mathematical analysis of the behaviour of the bongo board at that time, nor did we implement any control algorithms besides simple PID control.

The analysis of the three control policies developed – Stiff Upper Lip, Do the Shake, and Let’s Sway – was primarily qualitative, with minimal quantitative analysis done. The quantitative analysis was limited to the behaviour of the robot’s torso.

This thesis expands significantly on this initial research, including a much more thorough mathematical analysis of the forces at work in a bongo board system (see Section 3.2), the implementation of additional control algorithms (Fuzzy Logic and AOANN, see Sections 4.3 and 4.4), and a much more rigorous analysis of the behaviour of both the rider’s torso and the bongo board’s deck (see Section 5).

⁸Jimmy is named after three-time Olympic medal winner in the 400m sprint and 400m relay Jimmy Ball of Dauphin, Manitoba



Figure 3.15: A human riding a bongo board illustrating the Stiff Upper Lip control policy; the torso remains relatively upright and the knees are kept stiff while the lower spine and arms are used to maintain balance. (Baltes et al. [6])

3.4.1 Stiff Upper Lip

Humans observed riding a bongo board tend to use what Dr Jacky Baltes dubbed the Stiff Upper Lip policy; the torso remains relatively upright with the knees stiff while the lower spine is bent side-to-side for gross balance and the arms are used for fine balancing. Figure 3.15 shows two images captured from a video of a human riding a bongo board using this technique.

Inspired by this observed control policy I implemented a version of it on a DARwIn-OP humanoid robot. I used a PID controller to govern the inclination of the robot’s torso based on the following control law:

$$\theta_{torso} = K_p \cdot \theta_{torso} + K_d \cdot \dot{\theta}_{torso} \quad (3.22)$$

Because the robot used in this research lacks any kind of spine flexibility I had to rely on raising and lowering the hips by extending and contracting the legs to control θ_{torso} . Problems with this approach were readily apparent; the robot was too slow to respond to

disturbances, as no predictive model was used to compensate for sensor latency.

3.4.2 Do the Shake

I developed the Do the Shake control policy as a refinement of the Stiff Upper Lip policy. I implemented a simple predictive model to extrapolate the robot's current state based on sensor readings (see Section 3.4.4). The control scheme was further modified to use the position of the robot's torso for coarse corrections while the arms are used for fine corrections. The Do the Shake policy uses the following control laws:

$$\begin{aligned}\theta'_{torso} &= predicted(\theta_{torso}, \dot{\theta}_{torso}) \\ d_{arms} &= K_{p_{arm}} \cdot \theta'_{torso} + K_{d_{arm}} \cdot \dot{\theta}'_{torso} \\ \theta_{torso} &= K_{p_{torso}} \cdot \theta'_{torso} + K_{d_{torso}} \cdot \dot{\theta}'_{torso}\end{aligned}\tag{3.23}$$

When θ'_{torso} or $\dot{\theta}'_{torso}$ is small the torso calculations are not used; only the arms are adjusted for fine-tuning the robot's balance.

The Do the Shake policy was an improvement over the Stiff Upper Lip policy. Issues with jitter and latency persisted, preventing the robot from balancing independently for long periods of time.

3.4.3 Let's Sway

After the qualified success of the Do the Shake policy I developed a final control policy dubbed Let's Sway by Dr Baltes. Instead of attempting to maintain a statically stable position, the robot introduces a regular oscillation to the bongo board by pumping its legs up and down in addition to the coarse/fine control rules used by the Do the Shake policy.

The Let's Sway policy uses the following control laws:

$$\begin{aligned}
\theta'_{torso} &= predicted(\theta_{torso}, \dot{\theta}_{torso}) \\
\theta_{desired} &= \sin(\omega t) \\
d_{arms} &= K_{p_{arm}} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_{arm}} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired}) \\
\theta_{torso} &= K_{p_{torso}} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_{torso}} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired})
\end{aligned} \tag{3.24}$$

If the amplitude of the oscillation is set to zero, the Let's Sway policy is identical to the Do the Shake policy.

The Let's Sway policy demonstrated a marked improvement over the other two control policies; the robot was able to balance independently for several cycles before eventually becoming unstable, as can be seen in a YouTube video I published in 2013. [25]

The peak amplitude of 2.5mm for the legs' oscillation was determined experimentally. The period of the legs' oscillation was derived from the equation for the period of a standard pendulum's period where T is the period, l is the length of the pendulum, and g is the acceleration due to gravity:

$$T \approx 2\pi \sqrt{\frac{l}{g}} \tag{3.25}$$

In the case of the bongo board, l is determined by the hip height of the robot in its initial pose (y_{torso}). For an initial hip height of 200mm, this formula gives an oscillation period of approximately 0.9838s.

3.4.4 Predictive Sensor Model

Because of the highly-dynamic nature of the bongo board problem sensor latency was a significant problem when developing the control policies described above. The DARwIn-OP functions on an 8ms cycle time with the sensor and gyroscope readings delivered once per

cycle. This means that the robot’s sensor data is at-best 8ms old by the time it is input into the balancing program.

To deal with this latency I developed a simple predictive sensor model that could extrapolate the robot’s current inclination and angular velocity from previous data points [4, 6]. Given the design of the DARwIn-OP’s software it is safe to assume that data points are consistently 8ms apart.⁹ Given this assumption it is trivial to produce a linear approximation of the robot’s current inclination based on the latest two sensor readings:

$$\theta_{real} = \theta_{latest} + (\theta_{latest} - \theta_{previous}) \quad (3.26)$$

The robot’s current angular velocity is similarly calculated using a linear approximation.

During development of this predictive sensor model it was discovered that the robot’s gyroscope was not sensitive enough to detect the small changes in the robot’s angular velocity. As shown in Figure 3.16 [6] the gyroscope records data in approximately 3°/s increments. Because of this limitation the gyroscope was abandoned in favour of extrapolating the robot’s current angular velocity based on its recorded inclination using the three-axis accelerometer.

Because the robot’s motions will affect the sensor readings, I multiply the speed of each limb by an experimentally-derived gain and subtract the result from the angular velocity reported by the sensors. This has the effect of cancelling out the effect of the robot’s motions on the sensors, giving a more accurate reading of the effect of external forces (e.g. gravity) on the system.

The linear approximation given in Equation 3.26 is highly susceptible to noise; a single

⁹Some small timing errors can be expected when reading sensor data, and some data points may be missing due to corrupted serial communication between the sensor and the main processing board, but on average data is collected at 8ms intervals.

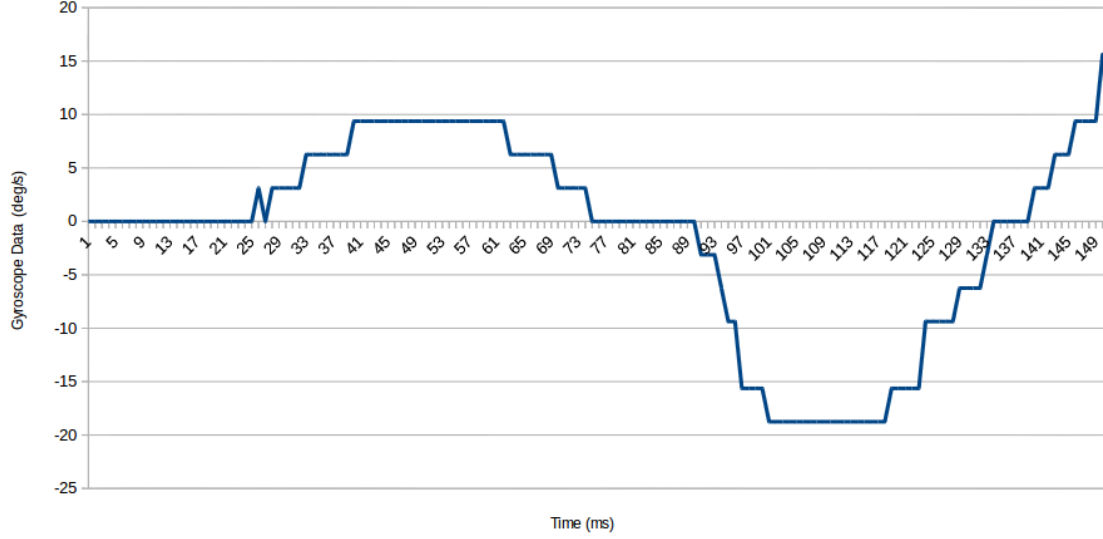


Figure 3.16: Raw gyroscope data recorded from the robot while executing the Stiff Upper Lip control policy. Note the coarse granularity of the data, approximating a step function. (Baltes et al. [6])

data point that is incorrect can cause the robot to erroneously believe that it is falling faster or slower than it really is. Therefore I implemented a nine-tab median filter to reduce sensor noise. While this filter introduces some latency to the sensor data, experiments showed that the linear approximation was sufficient to estimate the robot’s actual position with a reasonable degree of precision.

During preliminary research the average error in the prediction of the robot’s inclination was -0.002° with a standard deviation of 2.437. The average error of the predicted angular velocity was $0.0006^\circ/\text{s}$ with a standard deviation of 2.931. [6] Figures 3.17 and 3.18 show the actual recorded inclinations and angular velocities compared with the results of the linear approximation.

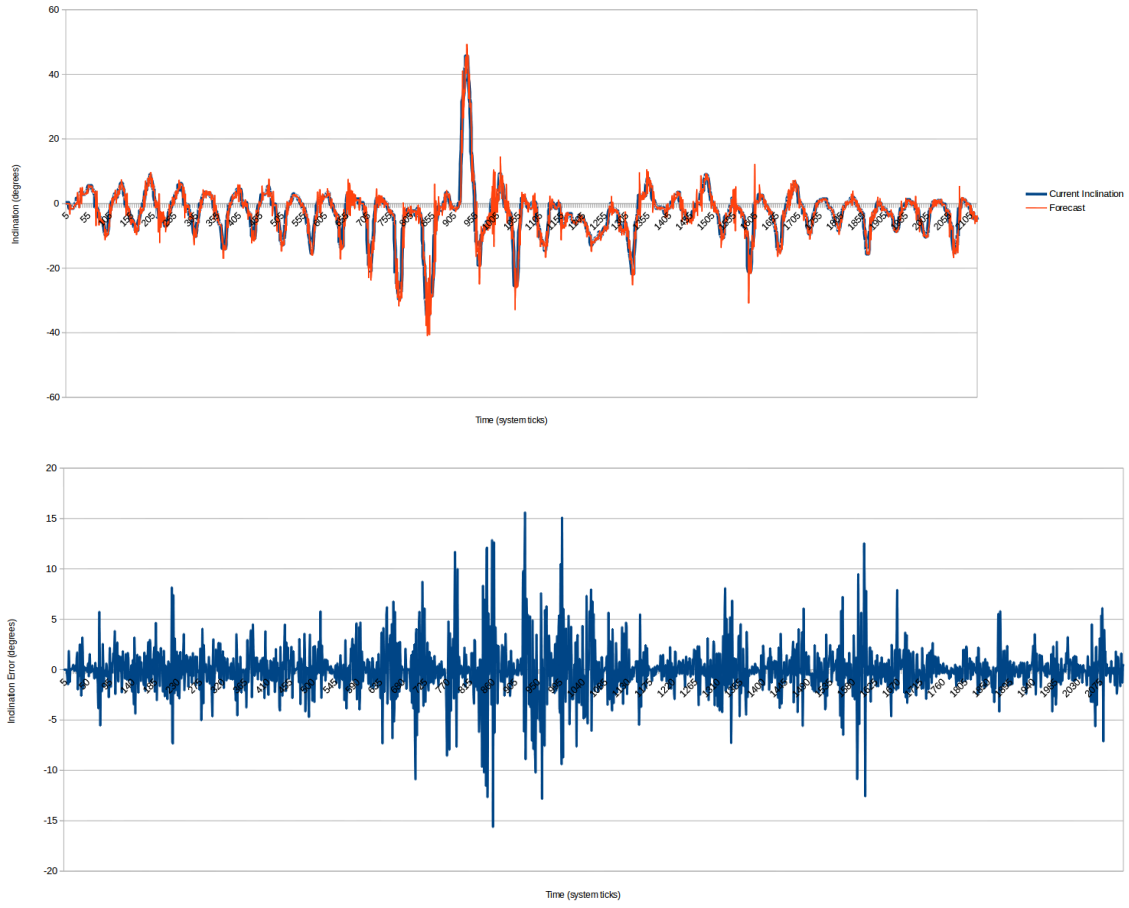


Figure 3.17: The inclination predicted by the sensor model versus the actual recorded inclination (top) and the error in the prediction (bottom). (Baltes et al. [6])

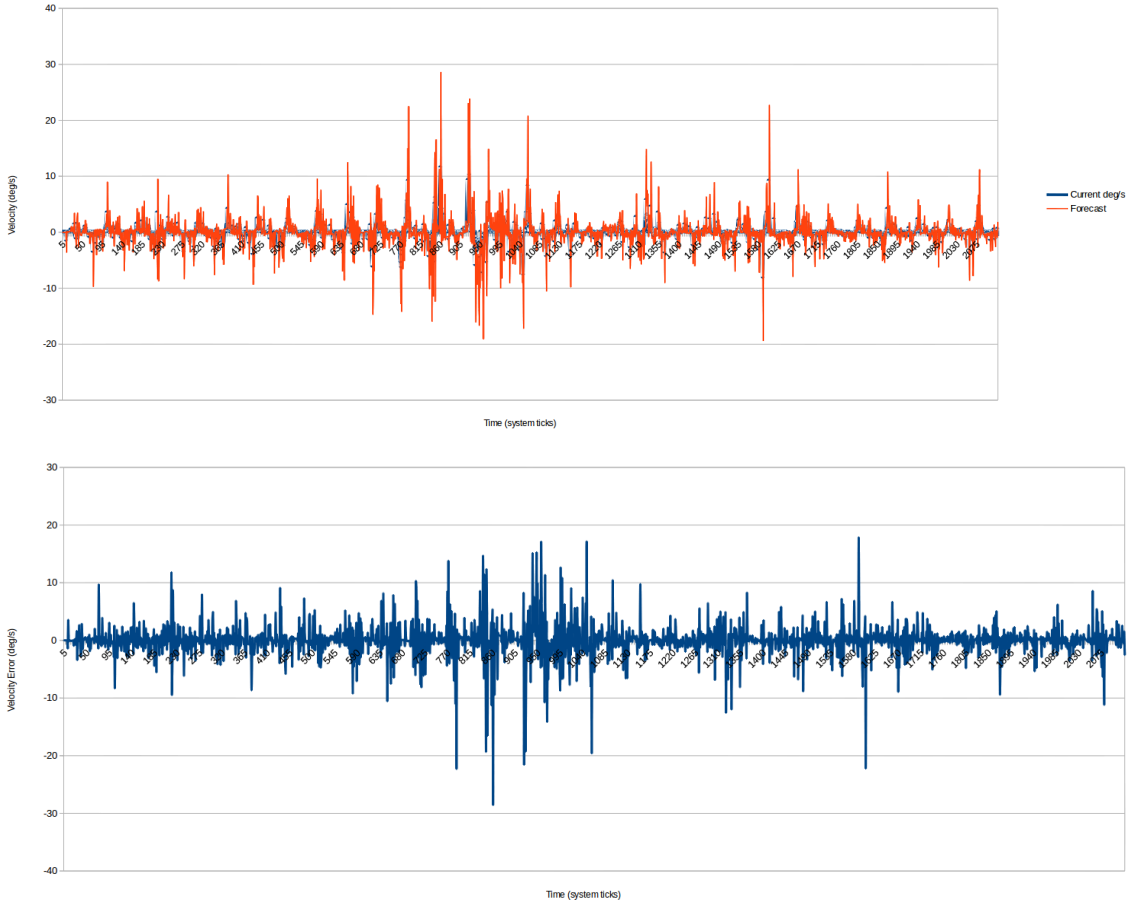


Figure 3.18: The angular velocity predicted by the sensor model versus the actual recorded angular velocity (top) and the error in the prediction (bottom). (Baltes et al. [6])

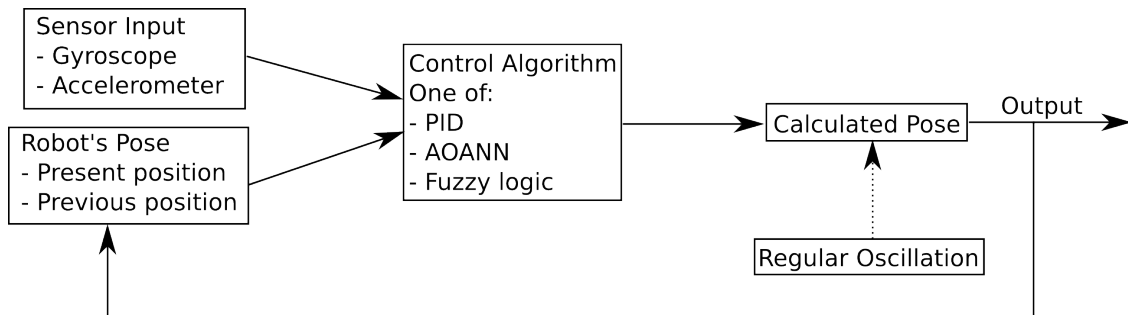


Figure 4.1: A flowchart showing an overall view of how the bongo board control software is written. The regular oscillation is introduced only when using the Let’s Sway control policy.

4 Experimental Design

Several algorithms have been shown to be effective at balancing a cart-and-rod inverted pendulum system (see Section 2.2), but prior to this thesis only PID controllers have been used on the bongo board problem [6]. I implemented two additional algorithms to replace the PID control rules in the Let’s Sway and Do the Shake control policies discussed in Section 3.4:

- an Always-On Artificial Neural Network (AOANN); and
- a fuzzy logic controller.

The Do the Shake policy was chosen because it most closely mimics how humans are observed balancing on the bongo board within the limited range of motion of the robot. The Let’s Sway policy was chosen because it exhibited the highest stability during the preliminary research. The Stiff Upper Lip policy was omitted because of its relative lack of success compared to the other two policies.

Figure 4.1 shows how the software is set up. The basic form of the software is the same as that used in the preliminary research with the exception that the control algorithm is replaced. The implementation of each of the replacement algorithms is described in more

detail below. Regardless of what control algorithm is used the sensor input will always be put through the predictive model developed for the Do the Shake policy, described in Section 3.4.

Each algorithm was tested under identical conditions using an untethered¹⁰ DARwIn-OP robot running on battery power. To ensure consistency a fresh battery is used for each trial. During each trial the robot continually logs its sensor and pose data. The robot is left to balance on its own for a period of 30 seconds at a time.

During each 30-second trial the number of times the robot fails to keep balance is recorded; a failure is indicated by either side of the deck of the bongo board striking the ground and the robot being unable to autonomously right the board. Whenever a failure occurs the robot is manually placed back in the starting position with the wheel centred and the deck level. A total of five 30-second trials are conducted for each combination of control algorithm and control policy, resulting in a total of 60 trials:

1. two sets of five with the PID controller,
2. two sets of five with the Fuzzy Logic controller, and
3. eight sets of five with the AOANN controller using four different pre-trained ANNs.

Once the 30-second timer has started it is not stopped during manual resets. However, in a similar fashion as Association Football (“soccer”), “stoppage time” is applied at the end of each trial to approximate the time lost due to manual resets.

4.1 Degrees of Freedom Used

To control the bongo board the rider uses up to five degrees of freedom:

- left and right leg length,
- lateral hip translation, and

¹⁰In this context untethered refers to a lack of AC power adapters and external ethernet cables.

Variable	Equivalent DoFs	Description
θ_L	Left shoulder roll	Left arm angle
θ_R	Right shoulder roll	Right arm angle
x_{torso}	Lateral hip translation	Lateral position of torso CoM
y_{torso}	Left and right leg lengths	Height of torso CoM
ϕ_r	Left and right leg lengths	Inclination of torso
y_L	Left leg length	Distance from left hip to ground
y_R	Right leg length	Distance from right hip to ground

Table 4.1: Degrees of freedom used by the rider and their corresponding variables from Figure 3.11

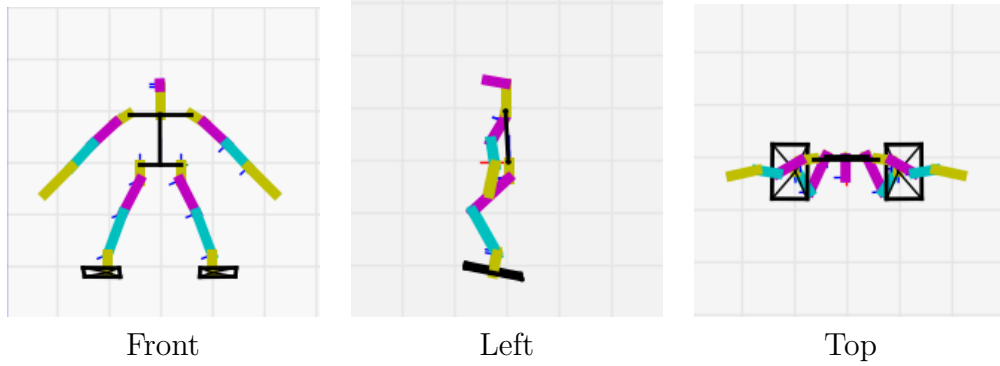


Figure 4.2: Three perspectives of the robot in its starting position. Note that the axes are drawn relative to the robot's origin (located on the waist); when the robot is standing its torso is inclined forward with the feet flat on the ground.

- left and right shoulder roll.

These parameters each contribute to controlling different variables illustrated in Figure 3.5, as shown in Table 4.1.

The robot is provided with an initial or default position when the balancing program begins. This position specifies values for y_L , y_R , θ_L , θ_R , and x_{leg} (the lateral separation between the feet). From these parameters all other variables can be calculated. The default values used are shown in Table 4.2. Figure 3.1 shows the robot standing in the default position and Figure 4.2 shows a three-perspective wireframe model of the robot in the same position.

Variable	Description	Value	Units
x_{torso}	Hip translation	0	mm
x_{leg}	Foot separation	100	mm
y_l, y_r	Hip height	200	mm
ϕ_r	Rider's inclination	0	°
θ_L, θ_R	Shoulder roll angle	45	°
n/a	Front-back foot offset	0	mm
n/a	Shoulder pitch angle	12	°
n/a	Hip pitch angle	10	°
n/a	Hip yaw angle	5	°
n/a	Ankle pitch angle	2	°
n/a	Elbow angle	0	°

Table 4.2: Values for the starting position of the robot and their corresponding variables from Figure 3.11. Note that not all values specified in the initial pose are represented in Figure 3.11; values with “n/a” in the left column are treated as constants during the entire experiment.

The lateral force that can be applied by the arms is zero when the shoulder angle is 90°. Similarly the vertical force that can be applied is zero when the arms are at 0° or 180°. Therefore I programmed the robot to include a natural decay in the arm position back towards the default position. Any calculated arm movement is always reduced by an amount equal to 5% of the distance away from the default position. The leg lengths and lateral hip translation also decay 5% towards the default position at each tick to prevent the robot from ever reaching a position where it is unable to execute corrective motions in any given direction.

As described in Section 3.4.3, the oscillation period of the legs when using the Let’s Sway policy is determined based on the robot’s initial pose, with an experimentally-determined peak amplitude. For all trials using the Let’s Sway policy I use a peak amplitude of 2.5mm and a period of 0.9838s.

4.2 PID Control

For the PID control I expanded the control laws published in Baltes et al. [6] (and described in Section 3.4) to include laws for controlling the height and lateral position of the CoM (y_{torso} and x_{torso} respectively). Equation 4.1 shows the expanded control law for the Do the Shake policy, while Equation 4.2 shows the expanded control law for the Let's Sway policy.

$$\begin{aligned}
\theta'_{torso} &= predicted(\theta_{torso}, \dot{\theta}_{torso}) \\
d_{arms} &= K_{p_{arm}} \cdot \theta'_{torso} + K_{d_{arm}} \cdot \dot{\theta}'_{torso} \\
\theta_{torso} &= K_{p_{torso}} \cdot \theta'_{torso} + K_{d_{torso}} \cdot \dot{\theta}'_{torso} \\
d_x &= K_{p_x} \cdot \theta'_{torso} + K_{d_x} \cdot \dot{\theta}'_{torso} \\
d_y &= K_{p_y} \cdot \theta'_{torso} + K_{d_y} \cdot \dot{\theta}'_{torso}
\end{aligned} \tag{4.1}$$

$$\begin{aligned}
\theta'_{torso} &= predicted(\theta_{torso}, \dot{\theta}_{torso}) \\
\theta_{desired} &= sin(\omega t) \\
d_{arms} &= K_{p_{arm}} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_{arm}} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired}) \\
\theta_{torso} &= K_{p_{torso}} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_{torso}} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired}) \\
d_x &= K_{p_x} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_x} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired}) \\
d_y &= K_{p_y} \cdot (\theta'_{board} - \theta_{desired}) + K_{d_y} \cdot (\dot{\theta}'_{board} - \dot{\theta}_{desired})
\end{aligned} \tag{4.2}$$

To reduce oscillations in the legs rather than calculating the necessary leg lengths according to θ_{torso} I created a two-tiered PID system to control the legs, shown in Figure 4.3. The output from the $(K_{p_{torso}}, K_{d_{torso}})$ PID controller is used as the goal value for a separate PID controller for each leg. These leg PID controllers use a high K_p and small K_d to dampen changes to the leg length.

The final implementation of PID control algorithm used in the trials uses seven separate

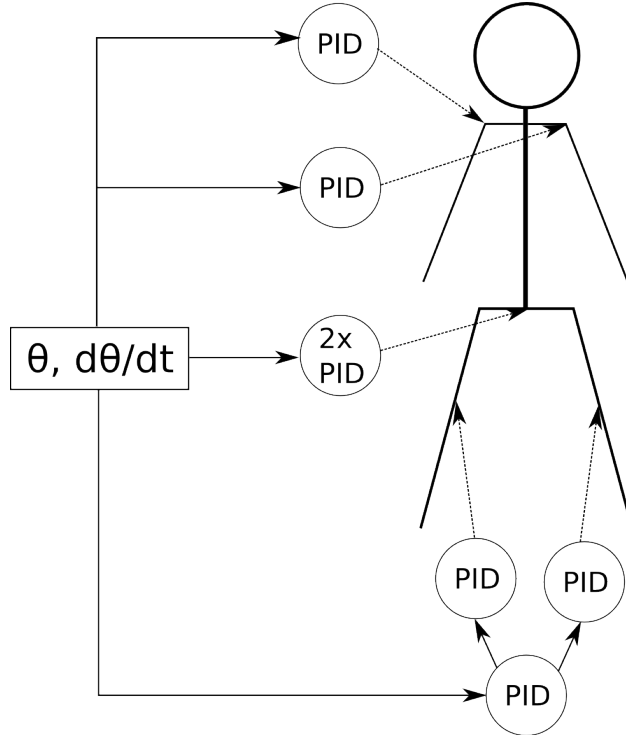


Figure 4.3: The seven PID controllers used in the trials. The arms and hip translation are all controlled independently by their own PID controller. The legs are controlled by a two-tiered set of three PID controllers: one high-level controller determines the robot’s desired inclination while independent controllers control the height of each leg.

PID controllers arranged in a two-tiered manner, as shown in Figure 4.3.

All PID gains used by the system are manually tuned based on trial-and-error experimentation, and use the robot’s current inclination and estimated angular velocity¹¹ as inputs.

4.3 Fuzzy Logic Controller

The Fuzzy Logic controller used to balance the bongo board is based on Yamakawa’s [51] controller, described in Section 2.2.3. I re-use many of Yamakawa’s rules, but add additional cases because of the increased number of degrees of freedom in the bongo board system;

¹¹Preliminary research showed that the robot’s on-board gyroscope was not sensitive enough to accurately detect small changes in the robot’s angular velocity. Therefore the difference between the previous and present inclination and the elapsed time between data recordings is used to estimate the robot’s current angular velocity. [6]

instead of applying voltage to an electric motor to drive the cart forwards or backwards the rider must control their inclination, hip translation, and arm positions. Algorithm 4.1 shows the rules used to balance the bongo board, using the same abbreviations found in Table 2.2. ϕ is the inclination of the bongo board, extrapolated from the rider's pose; one assumes both feet are in contact with the board. Given the robot's torso inclination, θ – measured by the accelerometer – and the positions of the feet one can trivially calculate the board's inclination.

Algorithm 4.1 Linguistic rules for balancing a bongo board.

if ϕ is ZR and $\dot{\phi}$ is ZR then $\dot{\theta}_{arm}$ is ZR $\dot{x}$ is ZR $\dot{y}$ is ZR $\dot{\theta}$ is ZR end if if ϕ is NS and $\dot{\phi}$ is PS then $\dot{\theta}_{arm}$ is ZR $\dot{x}$ is ZR $\dot{y}$ is ZR $\dot{\theta}$ is ZR end if if ϕ is PS and $\dot{\phi}$ is NS then $\dot{\theta}_{arm}$ is ZR $\dot{x}$ is ZR $\dot{y}$ is ZR $\dot{\theta}$ is ZR end if	if ϕ is PS and $\dot{\phi}$ is PS then $\dot{\theta}_{arm}$ is NS $\dot{x}$ is ZR $\dot{y}$ is ZR ϕhi is NS end if if ϕ is NS and $\dot{\phi}$ is NS then $\dot{\theta}_{arm}$ is PS $\dot{x}$ is ZR $\dot{y}$ is ZR $\dot{\theta}$ is PS end if if ϕ is PM and $\dot{\phi}$ is ZR then $\dot{\theta}_{arm}$ is NM $\dot{x}$ is NS $\dot{y}$ is NS $\dot{\theta}$ is NM end if if ϕ is NM and $\dot{\phi}$ is ZR then $\dot{\theta}_{arm}$ is PM $\dot{x}$ is PS $\dot{y}$ is PS $\dot{\theta}$ is PM end if
---	--

The control maps for these rules are shown in Figures 4.4 to 4.7. All input and output sets use overlapping membership functions of the form shown in Figure 2.6. Table 4.3 shows

	ϕ						
$\dot{\phi}$	NL	NM	NS	ZR	PS	PM	PL
PL							
PM							
PS			ZR		NS		
ZR		PM		ZR		NM	
NS			PS		ZR		
NM							
NL							

Figure 4.4: Control rule map for the rider's arms, $\dot{\theta}_{arm}$

	ϕ						
$\dot{\phi}$	NL	NM	NS	ZR	PS	PM	PL
PL							
PM							
PS			ZR		ZR		
ZR		PS		ZR		NS	
NS			ZR		ZR		
NM							
NL							

Figure 4.5: Control rule map for the rider's lateral hip translation, x_{hip}

the thresholds used for NL, NM, NS, ZR, PS, PM, and PL for each input and output set respectively.

These rules emphasise using the arms for fine-scale adjustments. The torso and hips move only if the board's deck is inclined; if the board is allowed to remain in an inclined position it will roll downhill along the wheel, eventually striking the ground.

Set	NL	NM	NS	ZR	PS	PM	PL	Units
ϕ	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00	$^{\circ}$
$\dot{\phi}$	-0.75	-0.50	-0.25	0.00	0.25	0.50	0.75	$^{\circ}/s$
$\dot{\theta}_{arm}$	-15.00	-10.00	-5.00	0.00	5.00	10.00	15.00	$^{\circ}/s$
$\dot{x}$								mm/s
$\dot{y}$								mm/s
$\dot{\theta}$	-3.00	-2.00	-1.00	0.00	1.00	2.00	3.00	$^{\circ}/s$

Table 4.3: Thresholds used for the fuzzy set membership functions for each set in the system

	ϕ						
$\dot{\phi}$	NL	NM	NS	ZR	PS	PM	PL
PL							
PM							
PS			ZR		ZR		
ZR		PS		ZR		NS	
NS			ZR		ZR		
NM							
NL							

Figure 4.6: Control rule map for the rider's vertical hip translation, y_{hip}

	ϕ						
$\dot{\phi}$	NL	NM	NS	ZR	PS	PM	PL
PL							
PM							
PS			ZR		NS		
ZR		PM		ZR		NM	
NS			PS		ZR		
NM							
NL							

Figure 4.7: Control rule map for the rider's torso inclination, $\dot{\theta}$

4.4 Always-On Artificial Neural Network

The Always-On Artificial Neural Network (AOANN), unlike the other algorithms presented in sections 4.2 to 4.3, is not a statically pre-configured algorithm. Instead the AOANN continually monitors the sensor readings and trains itself to balance better as it executes the experiments.

The neural network used in the experiments is a three-layer, fully-connected Artificial Neural Network (ANN): one input layer, one hidden layer, and one output layer, all using symmetric sigmoid activation functions. The inputs are scaled to lie in the range $[-1,1]$. The ANN is implemented using the open-source FANN (Fast Artificial Neural Networks) library [16].

The ANNs are trained on experimental data collected when performing the experiments with the PID and Fuzzy Logic implementations using the Do the Shake method (i.e. no induced rhythmic oscillations). Four separate ANNs are trained in groups of two with different hidden layer sizes; two ANNs are trained from the results of the PID controller and two from the Fuzzy Logic controller. All four ANNs are used for the experiment, using both the Do the Shake and the Let's Sway policies, with their performances recorded separately.

While training based on the data produced by PID and Fuzzy Logic controllers means that whatever biases present in these systems will be learned by the ANNs, there is no other practical source of training data available. Because the neural networks use Always-On training during their execution, any biases that were learned during the initial should eventually be trained away after a sufficiently long execution period.

The following sections describe the input and output layers, and the techniques used to train the neural network. Sections 4.4.1 and 4.4.2 discuss the encoding and scaling of the ANN's input and output layers respectively. Section 4.4.3 discusses the hidden layer and the reasoning that went into choosing the number of hidden neurons.

For simplicity the four ANNs trained for the experiment shall be named according to

their respective training sets and numbers of hidden neurons: PID-28, PID-800, Fuz-28, and Fuz-800 where “PID-” indicates that the ANN was trained based on the output of the PID controller, and “Fuz-” indicates that the ANN was trained based on the output of the Fuzzy Logic controller.

4.4.1 Input Layer

The ANN’s input layer consists of 14 neurons, one for each of the following parameters:

1. current torso inclination,
2. current torso angular velocity,
3. previous torso inclination,
4. previous torso angular velocity,
5. current left leg length,
6. current right leg length,
7. current lateral hip translation,
8. current left shoulder roll,
9. current right shoulder roll,
10. previous left leg length,
11. previous right leg length,
12. previous lateral hip translation,
13. previous left shoulder roll, and
14. previous right shoulder roll.

These inputs are all scaled to lie in the range $[-1,1]$, given the maximum and minimum values given in Table 4.4. The ranges for the inclination (θ_r) and angular velocity ($\dot{\theta}_r$) were chosen based on the data recorded during preliminary research. Under ideal circumstances these values should remain near zero, but a wider range is allowed to ensure that these inputs do not fall outside the $[-1,1]$ range. The allowable ranges for the variables defining the rider’s pose are defined according to the robot’s physical characteristics. For instance, the robot’s leg, when fully extended, measures 218mm. The shoulder rotation is limited to minimum of 20° to reduce the risk of the fingertip striking the leg when the arm is fully contracted.

Variable	Description	Min Value	Max Value
θ_r	Rider's inclination	-45°	45°
$\dot{\theta}_r$	Rider's angular velocity	$-100^\circ/\text{s}$	$100^\circ/\text{s}$
y_L	Curr. left leg length	140mm	218mm
y_R	Curr. right leg length	140mm	218mm
x_{torso}	Curr. lateral hip translation	-100mm	100mm
θ_L	Curr. left shoulder roll	20°	160°
θ_R	Curr. right shoulder roll	20°	160°
y'_L	Prev. left leg length	140mm	218mm
y'_R	Prev. right leg length	140mm	218mm
x'_{torso}	Prev. lateral hip translation	-100mm	100mm
θ'_L	Prev. left shoulder roll	20°	160°
θ'_R	Prev. right shoulder roll	20°	160°

Table 4.4: The maximum and minimum values allowed for each of the neural network's input neurons

4.4.2 Output Layer

As with the PID and Fuzzy Logic controllers, the AOANN controller must output five values: the desired positions for

1. the left arm,
2. the right arm,
3. the left leg length,
4. the right leg length, and
5. the lateral hip translation.

These outputs must lie within the ranges shown in Table 4.4.

The symmetric sigmoid function used by the output neurons only outputs values in the range $(-1,1)$ (see Equation 2.9). Therefore the output values must be converted from this range to a range appropriate for each of the five variables. To facilitate this the ANNs use a single-hot encoding; each output variable is assigned a fixed number of output neurons,

DoF	Minimum	Maximum
Left Shoulder	-22.959	23.993
Right Shoulder	-19.868	22.953
Hip Translation	-5.222	5.948
Left Leg	-3.224	3.231
Right Leg	-2.231	3.224

Table 4.5: The maximum, minimum, and average recorded changes for each of the 5 DoFs used in the PID and Fuzzy Logic experiments

each representing a discrete value with a pre-determined granularity. Only the neuron at the desired output position should be “hot” (i.e. have a value at or near 1.0). All other neurons should read “cold” (i.e. have a value at or near -1.0).

Furthermore, rather than output the absolute position for the DoF, the ANN is trained to output the offset that should be applied to the DoF’s current position. This allows for much finer precision with the single-hot encoding while maintaining a relatively low number of output neurons (and by association a lower number of connection weights to train).

For example, each arm has a range of motion of 160° . To encode the absolute position with a precision of 0.5° each arm is allocated 320 output neurons. To set the arms to 25° and 33° respectively the 50th neuron of one arm’s set of output neurons and the 66th neuron of the other arm’s output set would be hot. All other arm output neurons would be cold.

If the ANN outputs multiple hot neurons for a single output set, I use a weighted average of all hot neurons to decide the final output. If all neurons in a set read cold, no change is made to the variable controlled by those neurons.

Examining the results from the PID and Fuzzy Logic experiments I observed that the five DoFs do not move from one extreme position to the other; for example the arm will not move from 20° immediately to 180° . In fact, the largest change recorded in the arm position was 22.9° . Table 4.5 shows the minimum and maximum recorded changes for each of the 5 DoFs used in the PID and Fuzzy Logic experiments. From these values I concluded that I

Value	Resolution	Number of Neurons	Resolution	Number of Neurons
Δ Arm roll	0.05°	1200 ea.	0.1°	600 ea.
Δ Hip translation	0.05mm	400	0.1mm	200
Δ Leg length	0.05mm	200 ea.	0.1mm	100 ea.
	Total	3200	Total	1600

Table 4.6: The output resolution and the number of neurons used for each of the output variables. Each arm and leg has its own separate output, so their neuron counts are doubled in the final total. To reduce the time required for training the coarser resolutions (shown in the far-right columns) were used when using higher numbers of hidden neurons.

could encode far more precise output values by reducing the range of the outputs; instead of spanning the complete range of motion of the limb, I programmed the ANN to output the offset to be applied to the current position. The current position is known (since it is an input to the ANN).

From the values in Table 4.5 I chose the output ranges as follows, arbitrarily expanding the range to allow for more extreme values should they occur:

- shoulder rotation: $[-30, 30]$
- hip translation: $[-10, 10]$
- leg length: $[-5, 5]$

Table 4.6 shows the resolutions used for each output variable and their corresponding numbers of neurons.

4.4.3 Hidden Layer

No general rule of thumb exists to determine the optimal size of an ANN's hidden layer, but many researchers indicate that the number of hidden neurons should be between the size of the input and output layers [9, 18]. This gives a maximum range of $[14, 3200]$ for the number of neurons given the output resolutions described above. From this range I selected two values:

ANN	Accuracy
PID-28	0.0007854845
PID-800	0.0015347941
Fuz-28	0.0009054163
Fuz-800	0.0016409515

Table 4.7: The accuracy of each of the 4 ANNs after initial training.

- 28, double the number of input neurons; and
- 800, half the number of output neurons when using the coarser resolution.

Initially 1600 hidden neurons were considered (being half the number of output neurons when using the finer resolution). This would have created an ANN with $14 + 1600 + 3200 = 4814$ neurons and a corresponding $14 \cdot 1600 + 1600 \cdot 3200 = 5142400$ connections (using a fully-connected ANN). It was quickly discovered that an ANN of this size could not be trained in a reasonable amount of time on the hardware available¹². Therefore the coarser resolution was created to allow an ANN with an 800-neuron hidden layer.

4.4.4 Training

The five experimental runs with the PID and Fuzzy Logic controllers using the Do the Shake policy resulted in approximately 23,000 rows of training data for each ANN. Each ANN was trained for a minimum of 1000 epochs on the complete data set with learning rate of 0.7 and a learning momentum of 0.2.

Training the 28-hidden neuron ANNs took approximately 48 hours each (training 10000 epochs). The 800-hidden neuron ANNs took approximately 72 hours each to train for 1000 epochs. Table 4.7 shows the final accuracy of the ANNs after training.

All ANNs showed a relatively high initial error (to be expected with the initial random

¹²FANN does not currently support using a GPU for training [14]. Therefore all training had to be done on a CPU. The computer used for training was the author's home server, equipped with a 1.6GHz dual-core, 64-bit AMD Turion processor.

weights), but improved dramatically within the first 50 epochs. The remaining epochs showed steady improvement, but with increasingly small returns. For instance, between epochs 50 and 10000 the error of Fuz-28 improved from 0.0010245015 to 0.0009054163.

While the ANN is running during the experiments it undergoes Always-On (or online) training. This online training compares the robot's previous output with its current state. Given the robot's pose in the previous window and the robot's current sensor data one can determine if the previous output pose was correct or not. Using the equations of motion derived in Section 3.2.2 the program calculates the motions necessary to cancel out the robot's current angular velocity and return to a (or maintain the current) stable position.

This calculated pose is used as the goal output for the ANN, with the inputs being identical to those provided in the previous window. The ANN is then trained on this set of input data and desired outputs for ten epochs before calculating the output for the current window.

5 Results and Evaluation

To evaluate the effectiveness of each control algorithm and control policy we must consider the failure rate of each according to the experimental procedure outlined in Section 4 as well as the sensor data recorded during each trial. An ideal system should have a failure rate of zero, as well as near-constant inclination and angular velocity consistently near zero. Note that the torso inclination does not necessarily need to be near zero; provided the deck is stable the rider’s body may be in any configuration necessary to keep the CoM positioned over the wheel. While it may be aesthetically preferable for the torso inclination to be near-zero, this is not a strict requirement for evaluation.

Because of the size of the bongo board used in the experiment the robot has a minimal amount of time to react before the edge of the deck strikes the ground. When level, the underside of the deck is only 3cm above the ground. If the extreme end of the deck is falling downward at $1g$ (approximately 9.81m/s^2) the robot has only approximately 78ms before the deck strikes the ground. This corresponds to just under 10 system ticks (9.776 ticks), given the robot’s internal 8ms timer. This narrow decision window means that a certain number of failures per trial is likely inevitable. Therefore I evaluate the combinations of control algorithms and policies based primarily on their ability to minimise the robot’s angular velocity and maintain a consistent torso inclination. The failure rate is used as a secondary metric.

Table 5.1 shows the number of failures recorded during each trial with every combination of control algorithm and control policy. Figure 5.1 shows these data as a bar graph.

Control Algorithm	Run No.	No. Fails (Do The Shake)	No. Fails (Let’s Sway)
PID	1	11	8
PID	2	8	11
PID	3	7	8

PID	4	8	9
PID	5	9	10
	Average	8.6	9.2
	Std Deviation	1.517	1.304
Fuzzy Logic	1	9	12
Fuzzy Logic	2	10	12
Fuzzy Logic	3	8	11
Fuzzy Logic	4	9	11
Fuzzy Logic	5	7	12
	Average	8.6	11.6
	Std Deviation	1.140	0.548
AOANN PID-28	1	11	5*
AOANN PID-28	2	10	13
AOANN PID-28	3	10	15
AOANN PID-28	4	11	13
AOANN PID-28	5	10	10
	Average	10.4	12.75
	Std Deviation	0.548	2.062
AOANN PID-800	1	*	*
AOANN PID-800	2	*	*
AOANN PID-800	3	*	*
AOANN PID-800	4	*	*
AOANN PID-800	5	*	*
	Average	n/a	n/a
	Std Deviation	n/a	n/a

AOANN Fuz-28	1	17	14
AOANN Fuz-28	2	13	10
AOANN Fuz-28	3	15	11
AOANN Fuz-28	4	10*	12
AOANN Fuz-28	5	9*	11
	Average	15.0	11.6
	Std Deviation	2.0	1.57
AOANN Fuz-800	1	*	*
AOANN Fuz-800	2	*	*
AOANN Fuz-800	3	*	*
AOANN Fuz-800	4	*	*
AOANN Fuz-800	5	*	*
	Average	n/a	n/a
	Std Deviation	n/a	n/a

Table 5.1: The results of each experimental run using the Do The Shake and Let’s Sway policies. Trials marked with a * character were not successfully completed and not counted toward the average and standard deviation.

5.1 Observations, Self-Recovery, and Catapulting

Subjectively the robot appeared able to balance for the longest continuous periods of time when using the Fuzzy Logic controller with the Do The Shake policy. This can be supported statistically by lower average deck inclination shown in Table 5.2 and in figures 5.2 to 5.5.

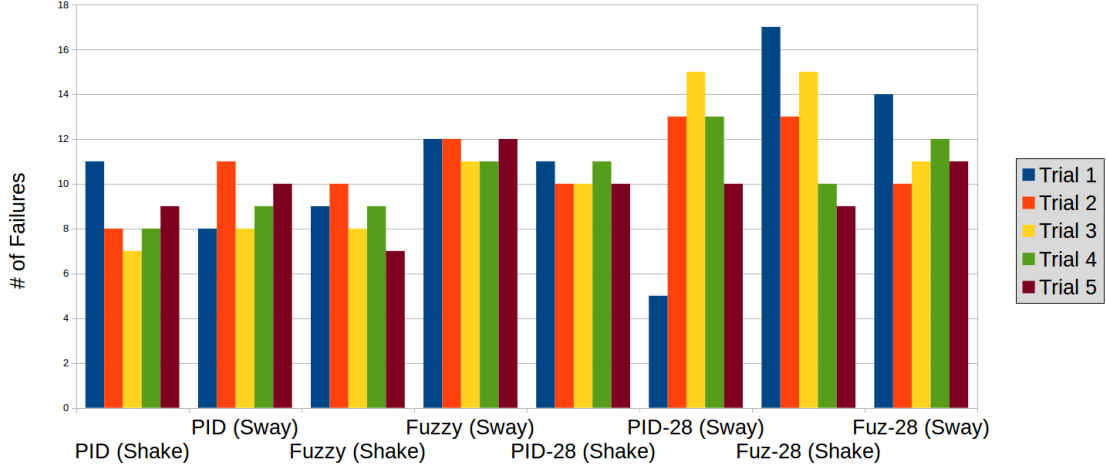


Figure 5.1: The number of failures recorded during each trial of the experiment. Note that the first trial using PID-28 and the Let's Sway control policy was stopped early due to a mechanical failure during the trial.

5.2 Sensor Analysis

The performance of each combination of control algorithm and control policy is mainly determined by the ability to minimise the bongo board's angular velocity and keep the ends of the deck from striking the ground. Under ideal conditions a successful combination of control algorithm and control policy should produce very little motion in the deck of the bongo board. In order for the board to stay stable, the deck must remain relatively level (i.e. deck inclination $\phi \approx 0$). The torso inclination may deviate from zero; it is possible for the wheel to be in such a position that the rider must incline their torso to one side in order to keep the deck of the board level.

Table 5.2 shows the minimum, maximum, and average deck inclinations and the difference in inclination between ticks ($\Delta\phi$, approximating $\frac{d\phi}{dt}$ ¹³) calculated by the robot while balancing. Figures 5.2 to 5.5 show the recorded deck inclination and $\Delta\phi$ during each trial. These graphs show the data for all trials in a single graph. For larger graphs of each individual trial see appendix A.

¹³Because the robot used operates on a constant 8ms cycle for reading the sensor values we can say $\Delta\phi \propto \dot{\phi}$

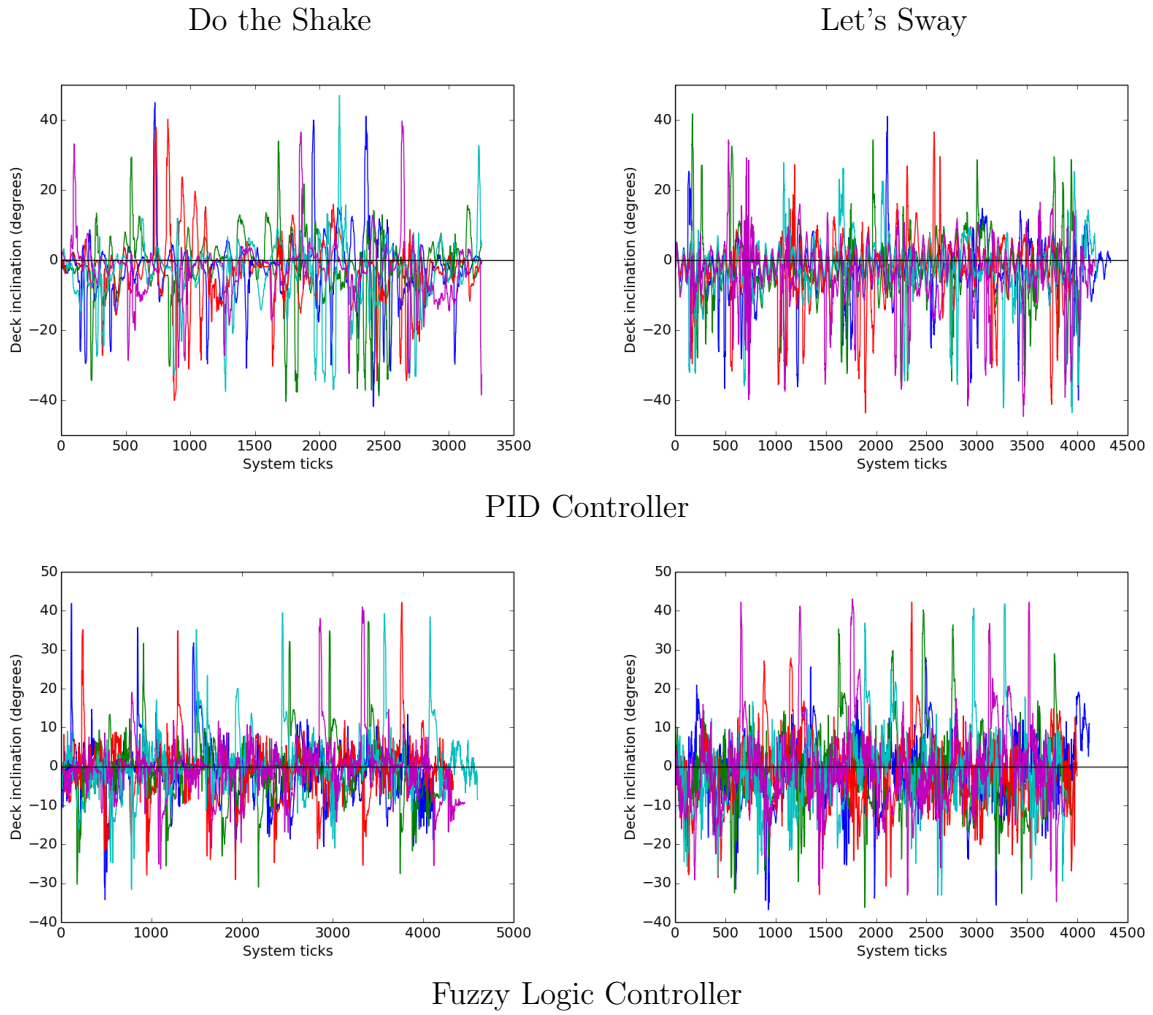


Figure 5.2: The inclination of the bongo board's deck during all trials using the PID and Fuzzy Logic controllers. For graphs of each individual trial see appendix A.

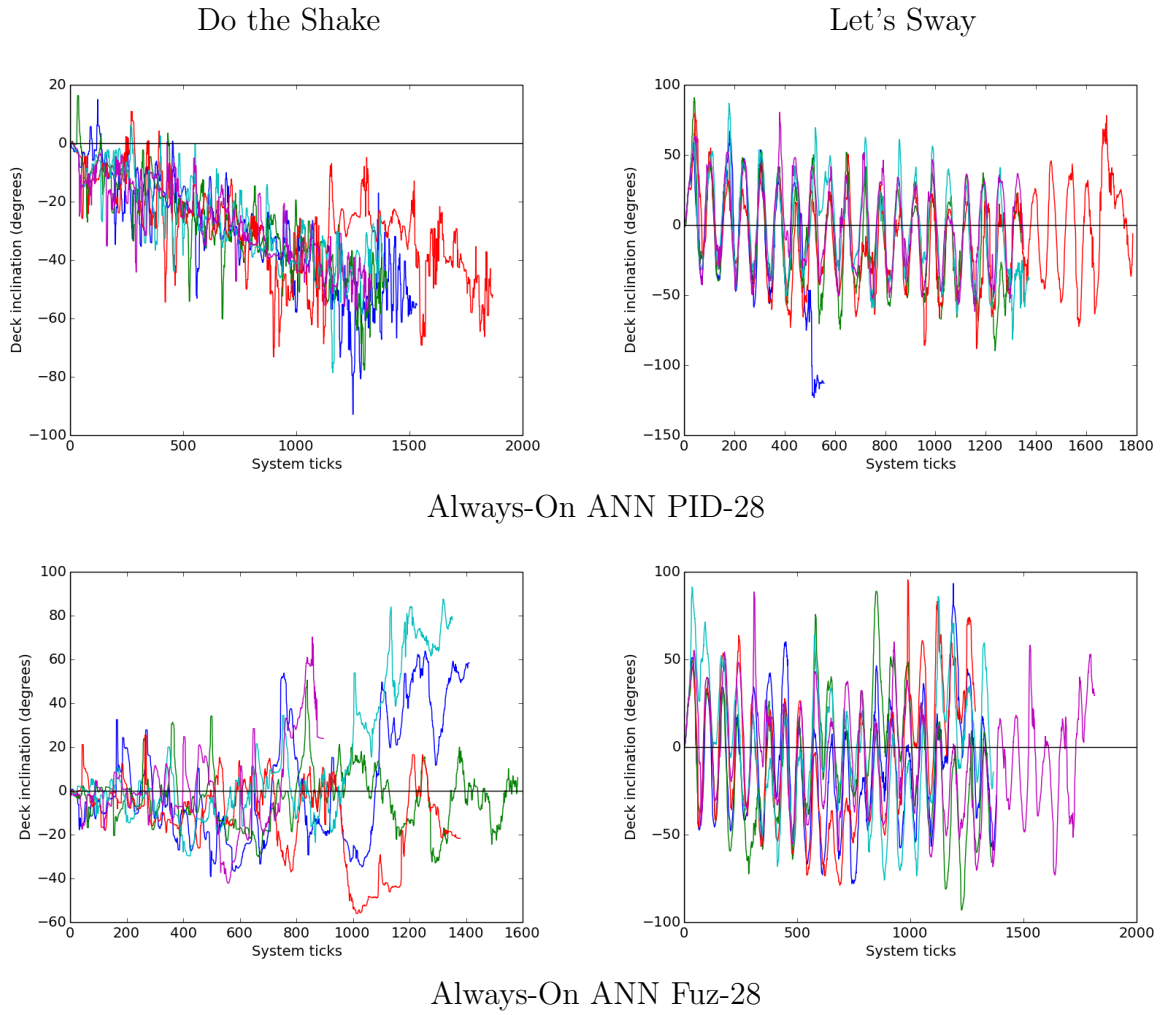


Figure 5.3: The inclination of the bongo board's deck during all trials using the AOANN controllers. For graphs of each individual trial see appendix A.

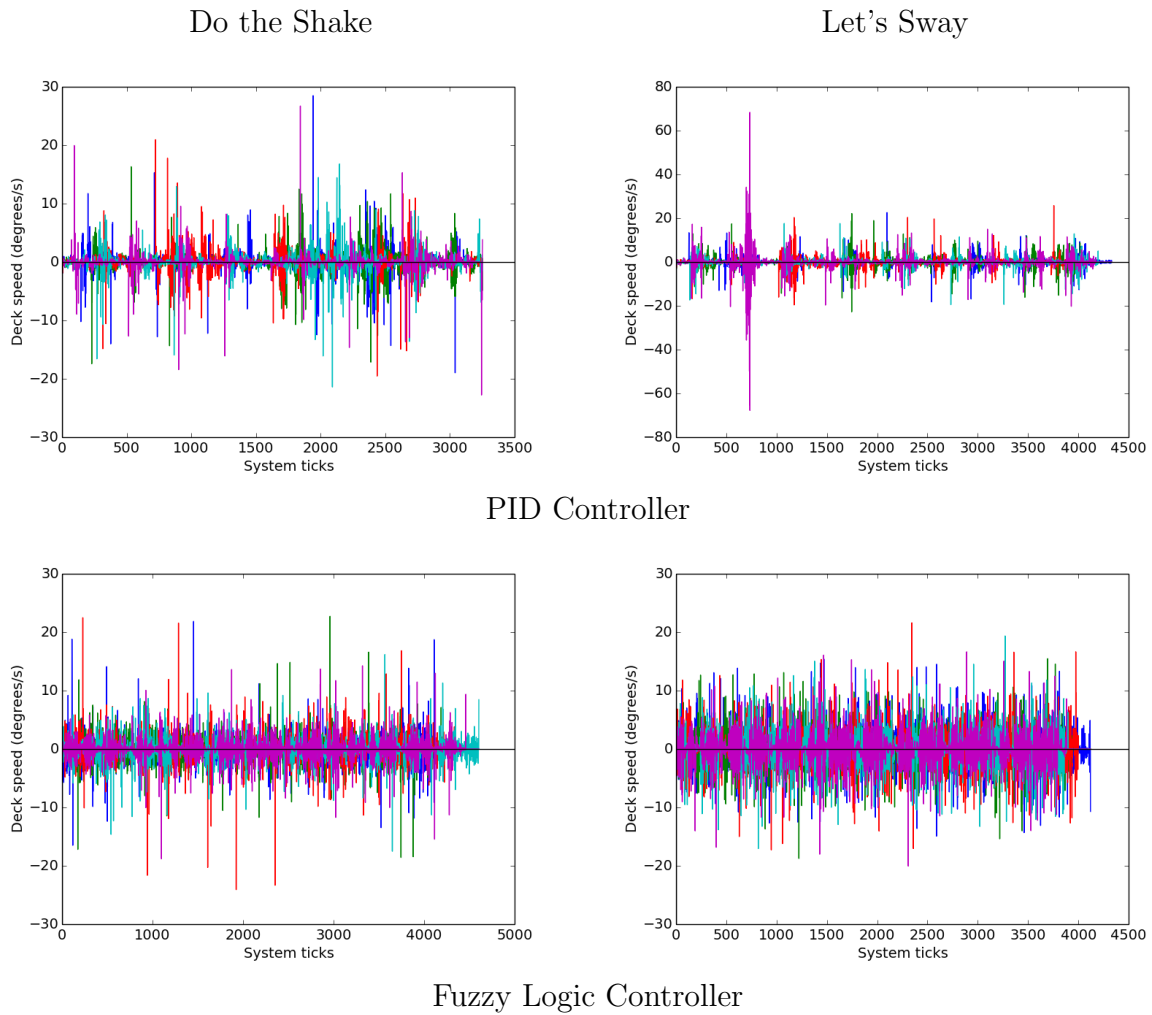


Figure 5.4: The difference in inclination of the bongo board's deck from the previous tick during all trials. For graphs of each individual trial see appendix A.

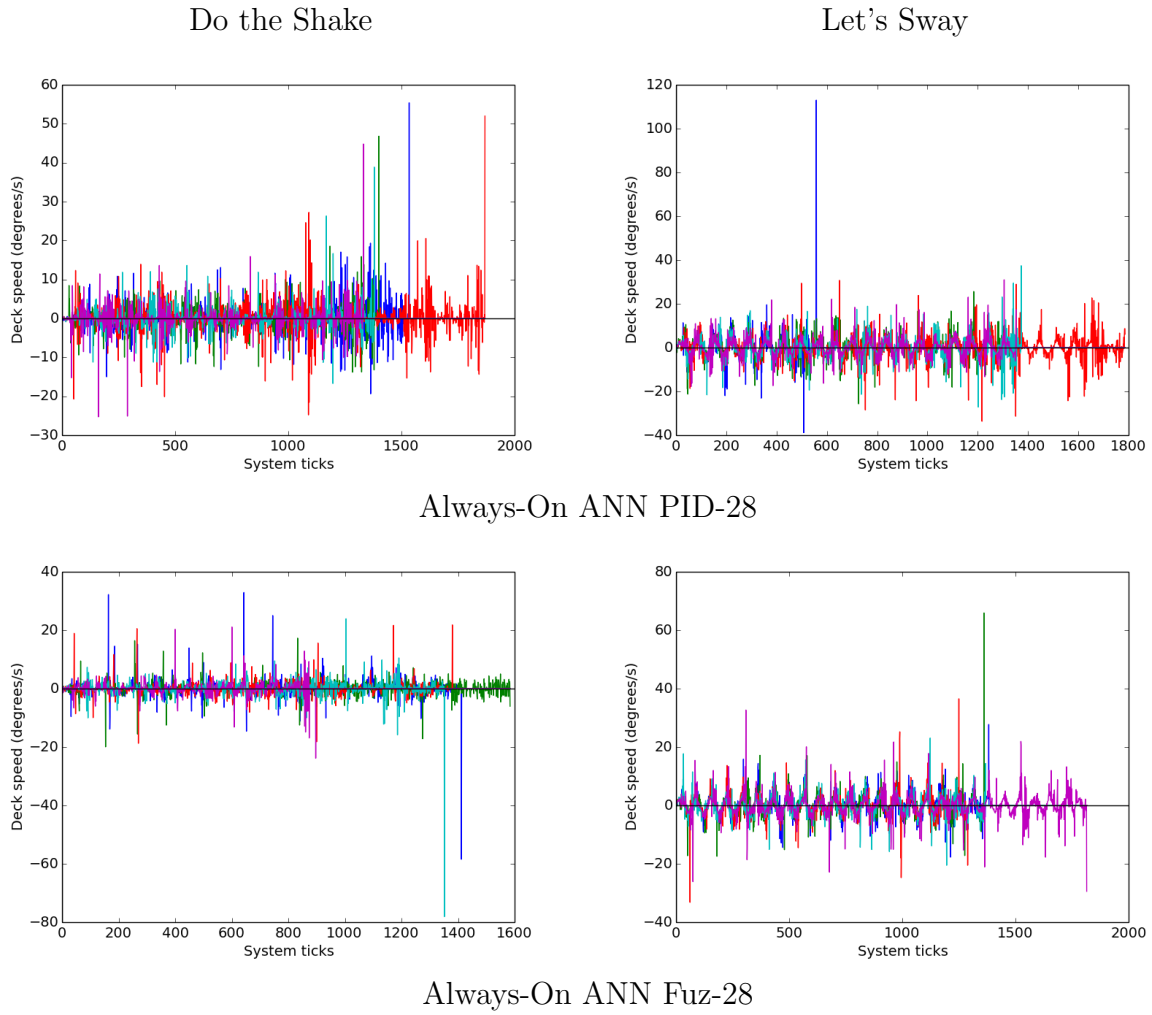


Figure 5.5: The difference in inclination of the bongo board's deck from the previous tick during all trials. For graphs of each individual trial see appendix A.

Do the Shake Policy

Control Algorithm	Min ϕ	Max ϕ	Average ϕ	σ	Min $\Delta\phi$	Max $\Delta\phi$	Average $\Delta\phi$	σ
PID	-41.7816	46.9645	-2.6833	9.1060	-22.7700	28.4904	1.917×10^{-3}	1.5709
Fuzzy Logic	-34.2622	42.1050	-0.7879	7.4964	-24.0847	22.7207	1.629×10^{-15}	1.6288
AOANN PID-28	-92.8650	16.3640	-28.7031	16.0895	-25.2980	55.3826	2.605×10^{-14}	2.7285
AOANN Fuz-28	-56.0985	87.4931	0.0966	23.8531	-78.0872	32.9003	1.343×10^{-14}	2.2722

Let's Sway Policy

Control Algorithm	Min ϕ	Max ϕ	Average ϕ	σ	Min $\Delta\phi$	Max $\Delta\phi$	Average $\Delta\phi$	σ
PID	-44.6387	41.6949	-2.8597	8.9129	-67.7595	68.3437	4.125×10^{-15}	2.1396
Fuzzy Logic	-36.7589	43.0499	-1.2893	9.4946	-20.0433	21.6206	-3.332×10^{-14}	2.5342
AOANN PID-28	-123.1170	90.7680	-6.1620	33.5484	-38.8416	112.9890	2.049×10^{-14}	4.0935
AOANN Fuz-28	-92.9956	95.4600	-6.0435	33.9956	-33.1983	65.9003	-1.386×10^{-14}	3.1352

Table 5.2: Minimum, maximum, and average deck inclination and difference in inclination from the previous tick. Measurements are in $^{\circ}$.

From Table 5.2 and figures 5.2 to 5.5 some general observations can be made:

- the oscillation of the Let's Sway policy appears to reduce the variability in the deck's velocity when used with the PID controller, but increases the variability of the deck's velocity when used with the Fuzzy Logic controller;
- there is little difference in the average deck inclination when using the PID controller with either the Let's Sway or the Do the Shake policies;
- PID-28 shows a consistent bias toward a right (negative) deck inclination, worsening as the trial goes on when using the Do the Shake policy;
- Fuz-28 shows severe degradation in the latter half of each trial; and
- when using the Let's Sway policy both AOANNs show little control over the deck inclination, with the inclination being dictated almost entirely by the legs' oscillation.

These observations are further supported by examining the torso's inclination (θ) and angular velocity ($\dot{\theta}$) as recorded by the robot's on-board sensors. These data are shown in Table 5.3 and in figures 5.6 to 5.9. These graphs show the data for all trials together. For graphs showing the data for each individual trial see appendix B.

From Table 5.3 and figures 5.6 to 5.9 we can make further observations:

- the Let's Sway policy appears to reduce both the inclination the angular velocity of the torso when using the PID controller;
- the Fuzzy Logic controller shows lower torso inclination and angular velocity when using the Do the Shake policy;
- while PID-28 shows a consistent bias toward a right (negative) torso inclination when using the Do the Shake policy, the slope of the trend is flatter than the deck inclination noted above; and

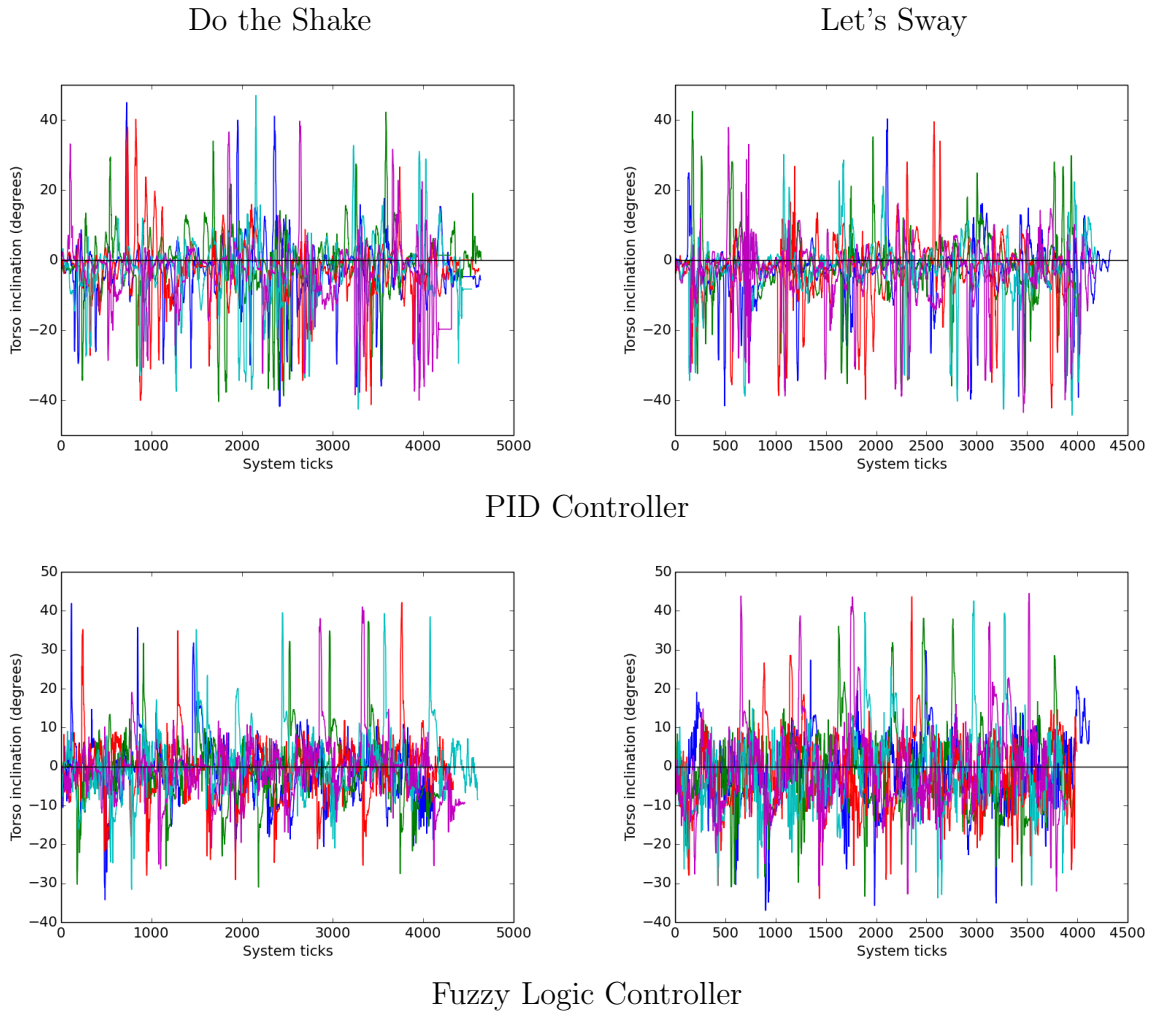


Figure 5.6: The robot's torso inclination during all trials with the PID and Fuzzy Logic controllers. For graphs of each individual trial see appendix B.

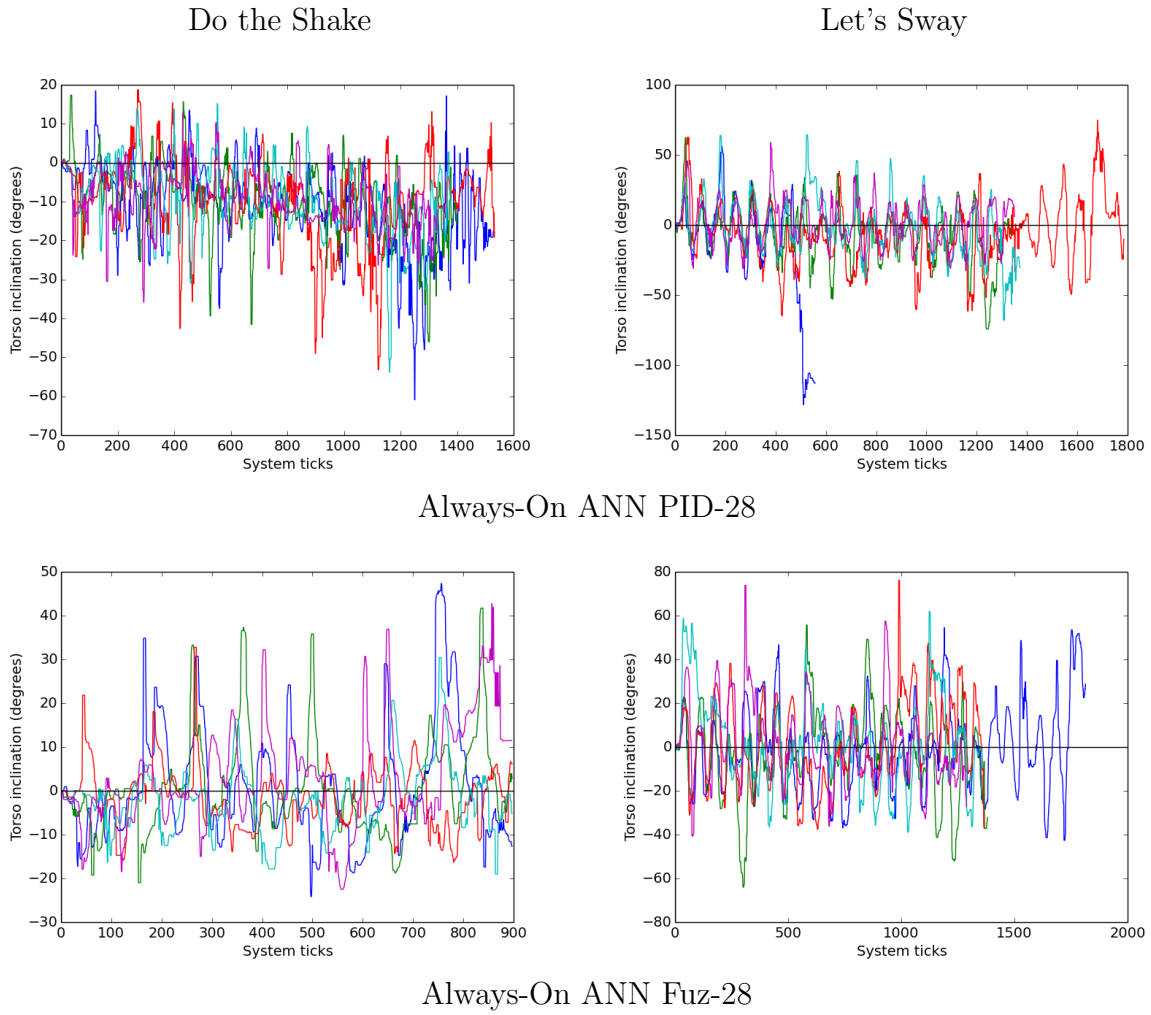


Figure 5.7: The robot's torso inclination recorded during all trials with the AOANN controllers. For graphs of each individual trial see appendix B.

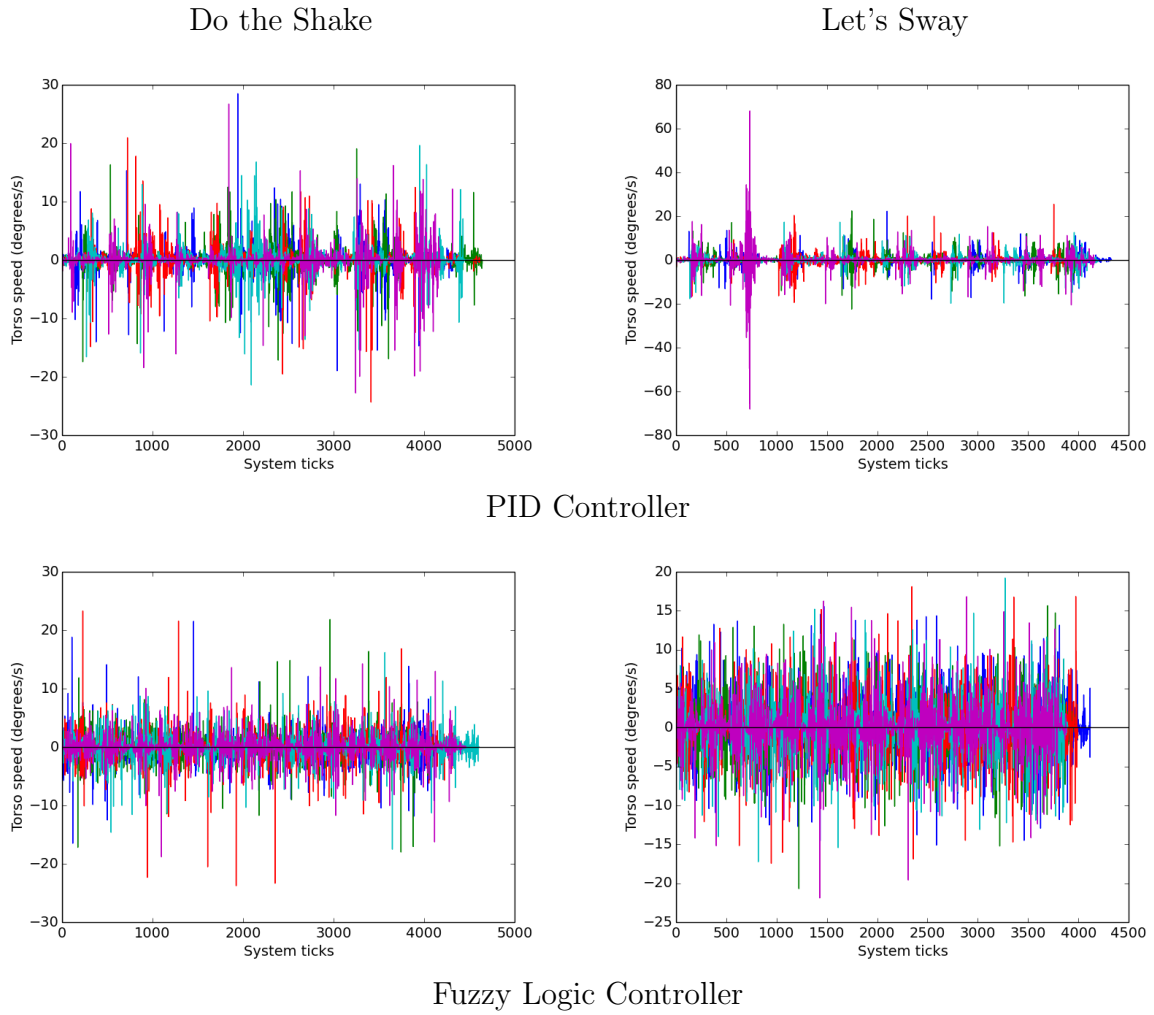


Figure 5.8: The robot's torso velocity during all trials with the PID and Fuzzy Logic controllers. For graphs of each individual trial see appendix B.

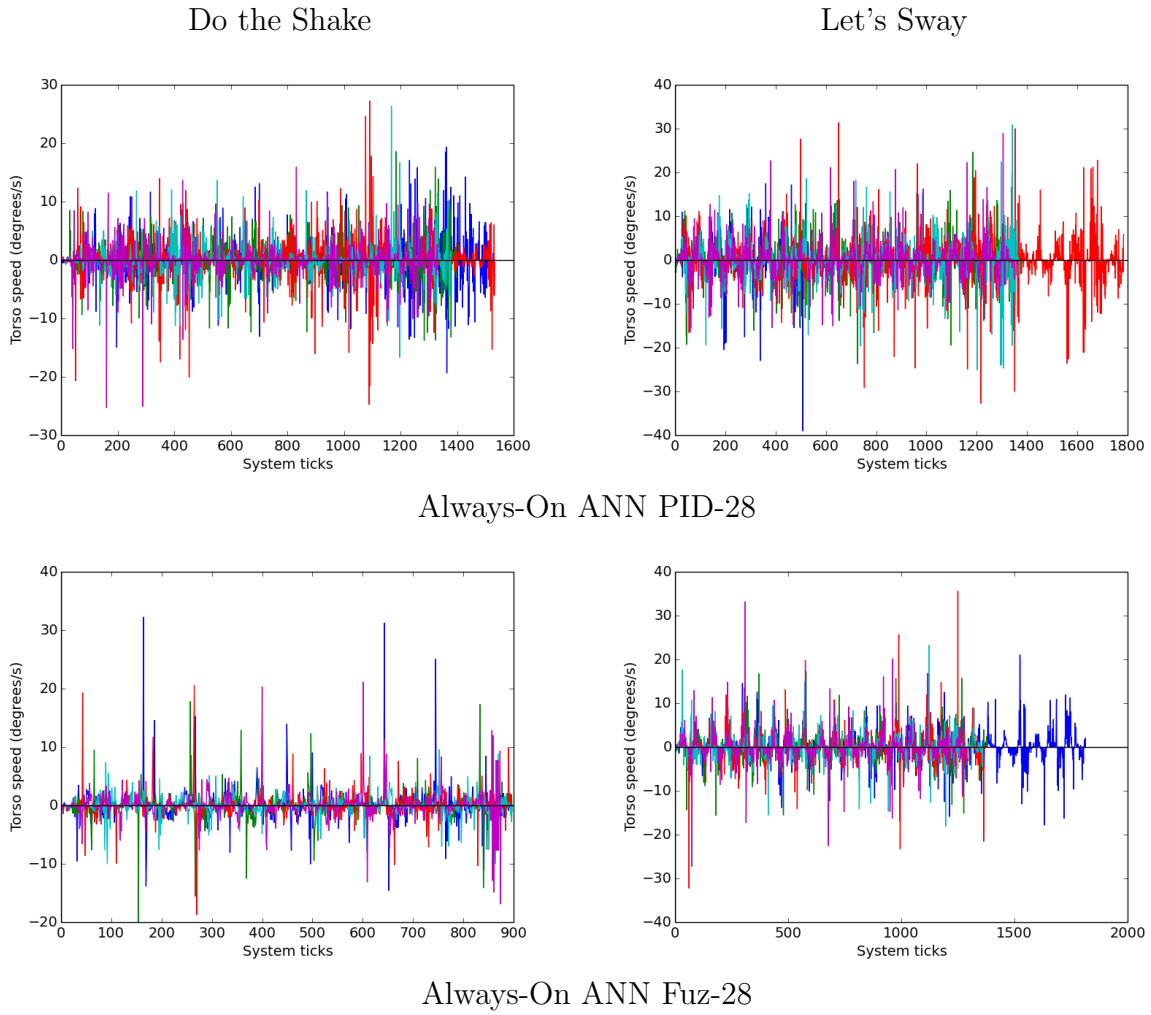


Figure 5.9: The robot's torso's angular velocity recorded during all trials. For graphs of each individual trial see appendix B.

Do the Shake Policy

Control Algorithm	Min θ	Max θ	Average θ	σ	Min $\dot{\theta}$	Max $\dot{\theta}$	Average $\dot{\theta}$	σ
PID	-42.5811	46.9645	-2.6620	9.1182	-24.3399	28.4904	-0.0010	1.6141
Fuzzy Logic	-34.2668	42.1050	-0.8253	7.6646	-23.7593	23.2903	-0.0022	1.5821
AOANN PID-28	-60.8165	18.7145	-10.0961	9.4050	-25.2690	27.2493	-0.0102	2.7148
AOANN Fuz-28	-24.1576	47.3043	0.0536	10.4766	-19.9364	32.2344	0.0006	2.0542

Let's Sway Policy

Control Algorithm	Min θ	Max θ	Average θ	σ	Min $\dot{\theta}$	Max $\dot{\theta}$	Average $\dot{\theta}$	σ
PID	-44.3100	42.4191	-2.8706	8.8555	-68.0614	68.0614	-0.0010	2.1837
Fuzzy Logic	-36.9610	44.4448	-1.4046	10.3874	-21.8892	19.1989	-0.0004	2.4143
AOANN PID-28	-128.266 ^a	74.9237	-4.3194	22.1599	-39.0171	31.3606	-0.0241	4.0501
AOANN Fuz-28	-64.0530	76.1414	-0.0217	19.3642	-32.2248	35.6079	-0.0009	2.7261

Table 5.3: Minimum, maximum, and average inclination and angular velocity of the robot's torso relative to the ground during all trials. Measurements are in $^{\circ}$ and $^{\circ}/s$.

^aWhen the robot catapulted the board the robot also fell on its side, producing a large outlier

- when using the Let’s Sway policy both AOANNs show a tight coupling with the leg oscillation, indicating that the torso and deck are closely synchronised.

5.2.1 Comparison of Control Algorithms

Both the PID and Fuzzy Logic controllers were able to successfully keep the deck in a stable position for short periods of time during the trials. Overall the Fuzzy Logic controller’s performance was better than the PID controller, but the difference is marginal. The AOANNs generally performed much more poorly than the other control algorithms, being unable to keep the deck in a stable position for any appreciable period of time during the trials.

Looking at Figures 5.2 and 5.3 we can see that the AOANNs exhibit no strong tendency to keep the deck inclination near-zero; PID-28 shows a consistent bias to keeping the deck inclined to the right, and Fuz-28 shows severely degrading, random performance as the trials went on. Both the PID and Fuzzy Logic controllers however show a tendency to keep the deck oscillating back-and-forth about the zero position, interspersed with peaks and valleys indicative of deck-strikes.

In terms of deck inclination the Fuzzy Logic controller used in conjunction with the Do the Shake policy had the best performance, with an average inclination of -0.7879° ($\sigma = 7.4964$). The PID controller on the other hand averaged -2.8633° ($\sigma = 9.1060$) with the Do the Shake policy and a nearly-identical -2.8597° ($\sigma = 8.9129$) when using the Let’s Sway policy.

Looking at the torso inclination, shown in Figure 5.6, one can see that the PID and Fuzzy Logic controllers both keep the torso oscillating around zero. As with the deck inclination, the Fuzzy Logic controller with the Do the Shake policy keeps the most stable torso inclination, followed closely by the PID controller using either control policy.

The fact that the torso inclination shows similar trends as the deck inclination shows that the control algorithms do not naturally mimic Stiff Upper Lip policy described in Section 3.4.

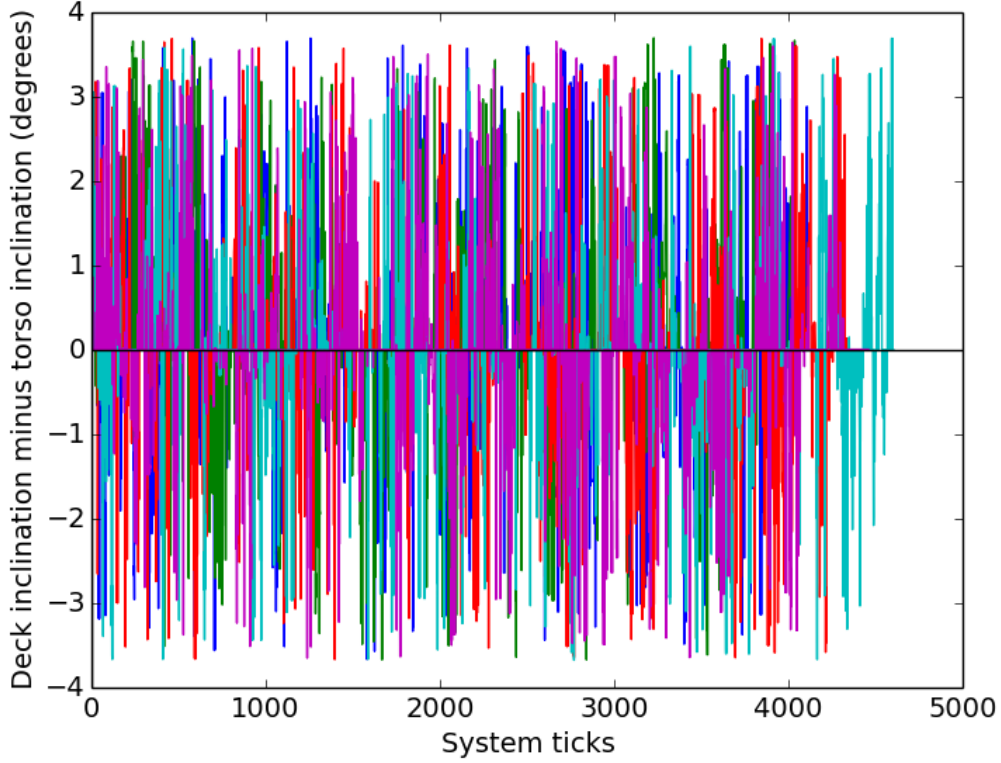


Figure 5.10: Difference between deck inclination and torso inclination recorded while using the Fuzzy Logic controller and the Do the Shake policy

The Stiff Upper Lip policy, observed in humans, effectively de-couples the deck inclination from the torso, allowing the deck to dramatically change inclination while the torso remains relatively stationary. Figure 5.10 shows the difference between the deck inclination and torso inclination recorded when using the Fuzzy Logic controller and the Do the Shake policy. As can be seen from this figure the deck inclination and torso inclination never vary more than 4° .

All control algorithms, regardless of control policy used, maintained an average difference in deck inclination between ticks close to zero. The PID controller used with the Do the Shake policy had the highest average $\Delta\phi$, but was still less than 0.01° . However, the standard deviation in $\Delta\phi$ did change noticeably between different algorithms. Both the PID and Fuzzy

Logic controllers, used with the Do the Shake policy, had the lowest standard deviations (1.5701 and 1.6288 respectively), reinforcing the conclusion that both these algorithms are suitable for the bongo board task.

5.2.2 Comparison of Control Policies

Both the Let's Sway and Do the Shake policies were able to balance the bongo board for short periods of time with the Do the Shake policy generally performing slightly better. Both control policies performed best when used with the Fuzzy Logic controller and performed worst when used with the AOANN controllers. There was negligible difference between policies when used with the PID controller.

The Let's Sway policy showed a tendency to overpower the AOANN controllers' outputs. Looking at figures 5.2 to 5.5 for the AOANN controllers using the Let's Sway policy one can see that both PID-28 and Fuz-28 exhibit very strong, cyclic patterns in deck inclination and $\Delta\phi$. This is due to the ANNs producing weak output and the motion of the robot being dictated almost entirely by the oscillation in the legs.

When used with the PID controller, the Let's Sway policy exhibited a slightly worse average deck inclination but a lower standard deviation than the Do the Shake policy under the same conditions. Overall the Do the Shake policy performed marginally better than the Let's Sway policy when used with the PID controller, but not to any statistically significant extent.

The Let's Sway policy did successfully maintain a lower average angular velocity in the robot's torso, as reported by Baltes et al. [6]. However, the difference was marginal: -0.0010086192°/s versus -0.0010486789°/s for a total difference of 4.00597×10^{-5} °/s. This indicates that, while not detrimental to balancing with a PID controller, the Let's Sway policy does not offer any significant benefits either.

The Do the Shake policy used with the Fuzzy Logic controller maintained the lowest av-

erage deck inclination with the smallest standard deviation of all experiments. Additionally, this combination of control algorithm and control policy had the lowest recorded average $\Delta\phi$. These findings are mirrored in the average torso inclination and angular velocity.

Overall the Do the Shake policy appears to be the better choice of control policy. The rhythmic oscillation introduced by the Let’s Sway policy does not significantly improve the robot’s ability to balance and introduces additional strain to the hip, ankle, and knee motors by keeping them in constant motion.

Unlike the AOANN and PID controllers, the Fuzzy Logic controller was unable to reliably recover from the deck striking the ground. When using the PID or AOANN control algorithm, the rapid decelerations that occurs when the deck strikes the ground regularly triggered an extreme reaction in the opposite direction. This reaction caused the robot to automatically bounce the board back toward the stable position. Occasionally this reaction would be too strong; the board would strike at the opposite end, and led to a cyclic “ping-pong” motion, eventually causing the robot’s feet to slip out of position and forcing a manual reset. The Fuzzy Logic controller occasionally exhibited this self-recovery property, but it was less frequent than with the other control algorithms when using the Do The Shake policy.

When using the Let’s Sway policy, all control algorithms, including the Fuzzy Logic controller, exhibited the self-recovery technique described above. The Let’s Sway policy appeared to have an amplifying effect on the ping-pong motion described above. This became especially apparent with the PID-28 and Fuz-28 AOANNs; both AOANNs exhibited “catapulting” behaviour, where the force of the reaction to a deck-strike was so strong that the entire deck would lift off the ground, causing the wheel to spin out and escape from below the deck.

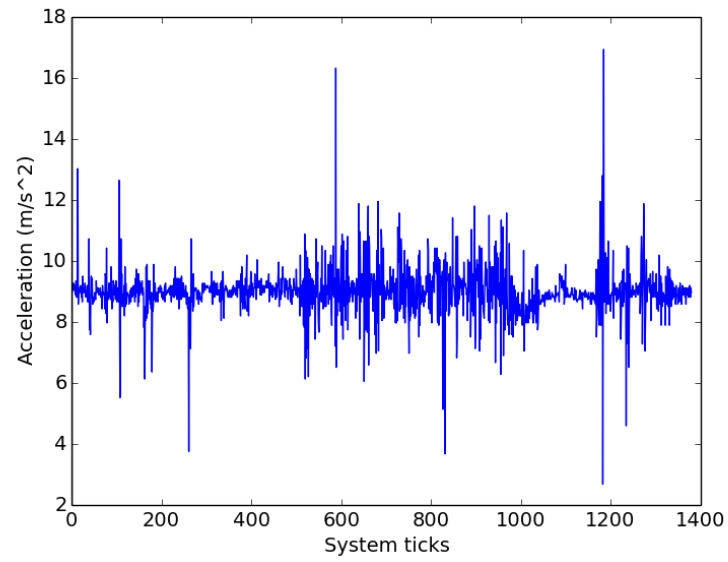
Catapulting was only exhibited during the trials using the Let’s Sway control policy. This is due to the higher forces experienced when the robot’s leg oscillations were timed such that a deck strike occurred while the leg was still extending. In these instances the deck would

strike the ground at higher speeds than normally experienced when using the Do the Shake policy. This in turn would cause a higher-than-normal spike in the accelerometer data, due to the speed at which the high-speed deck strike decelerates the robot. This spike causes an extreme self-recovery action, resulting in catapulting.

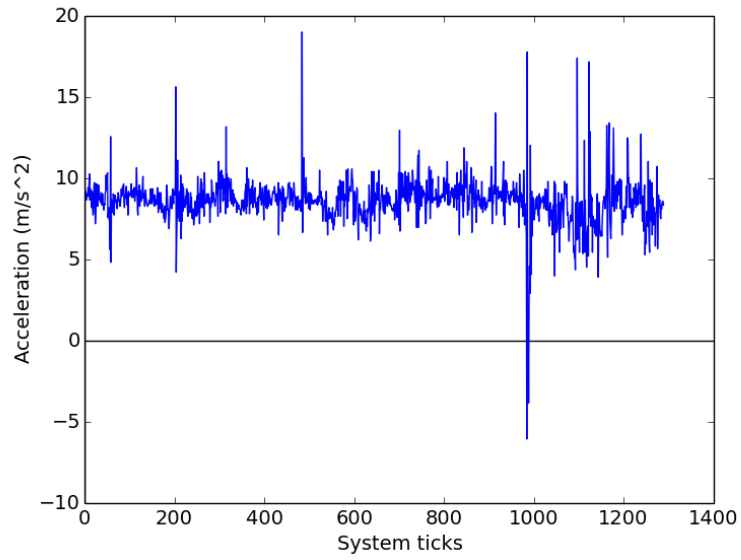
When a deck-strike occurs, a large spike in the robot’s acceleration is detected. This is especially visible in the robot’s z-axis (the axis running vertically through the robot’s head and torso). Figure 5.11 shows the data recorded from the robot’s z-axis accelerometer during the first trial using the Fuz-28 AOANN with both the Let’s Sway and Do the Shake policies. One can see from this figure that the vertical acceleration frequently exceeds g (approximately 9.81m/s^2). These spikes correspond to deck-strikes.

These spikes in the accelerometer data in turn produce large changes in the robot’s perceived inclination and angular velocity. When exposed to a large inclination or angular velocity, the PID controller uses these large values directly as inputs to the control formulae, shown in equations 3.23 and 3.24. These formulae produce large outputs, resulting in the strong reaction to deck-strikes observable in the experiments.

The fact that the Fuzzy Logic controller did not exhibit strong self-recovery nor a tendency to catapult is attributable directly to the way the Fuzzy Logic controller handles large inclinations or angular velocities. The Fuzzy Logic controller treats all values greater than or equal to its Positive-Large threshold as having a membership value of 1.0, and similarly for values less than or equal to its Negative-Large threshold. This means that the large spikes resulting from deck-strikes are treated the same as any other situation in which the robot’s acceleration or angular velocities are deemed “large” by the fuzzifier. Because the control rules described in Section 4.3 are designed to keep the board in a stable position there are no conditions that specifically address deck-strikes. Therefore the Fuzzy Logic controller never exhibits a strong reaction a deck-strike, and cannot self-recover to the same extent as the PID controller.



Do the Shake



Let's Sway

Figure 5.11: Z-axis accelerometer data during experimental trials with the Fuz-28 AOANN using both control policies.

Given that the self-recovery behaviour is a natural result of deck-strikes, the catapulting behaviour observed is a natural extension of this same principle. The cyclic motion of the robot’s legs tend to amplify the accelerometer spikes, as can be seen in Figure 5.11. This increased magnitude in the accelerometer data increases the strength of the self-recovery behaviour, resulting in catapulting.

This catapulting effect occurred once early in the first trial of PID-28 using the Let’s Sway policy (noted by the * in Table 5.1). The wheel rolled off the table and onto the ground, preventing a manual reset in any reasonable amount of time as the robot would have needed to be set down or held while the wheel was recovered. Therefore the balancing program was shut down immediately and the recorded number of failures not counted toward the summary statistics. Because the AOANN was training up until the time the program was cancelled, the inclination and angular velocity recorded by the robot’s sensors up until that point are included in the analysis below.

The catapulting effect was even more pronounced with the Fuz-28 AOANN. At least once during each trial, and sometimes multiple times, the wheel would escape from under the deck. In most instances the wheel was recovered quickly enough to allow a manual reset.

5.3 Failure Rate

After conducting all trials, two general results became immediately apparent: both the PID-800 and Fuz-800 ANNs were incapable of adequately controlling the robot’s motions to balance the bongo board, and broadly-speaking the Do the Shake policy performed better than the Let’s Sway policy. This latter finding contradicts the preliminary research published in Baltes et al. [6] which indicated that the addition of a rhythmic oscillation to promote dynamic stability improved the performance of the robot on the bongo board. As can be seen in Table 5.1, in all cases except the Fuz-28 AOANN, the Do the Shake policy had fewer failures.

From Table 5.1 we can see that the AOANN control algorithms either failed to function entirely (as was the case with PID-800 and Fuz-800), or were occasionally unable to complete all trials. For a discussion on the nature of these failures see Section 5.4.

The higher failure rates experienced by the Let’s Sway control policy are mainly due to the more violent self-recovery behaviour and catapulting that control algorithms experienced during these trials. The Fuz-28 ANN is the lone exception to this trend; while it exhibited the most catapulting of any algorithm under this control policy, it was also able to self-recover and ping-pong back-and-forth for relatively long periods of time before eventually catapulting.

5.4 AOANN Failure Analysis

All four AOANNs showed much poorer performance than the PID and Fuzzy Logic controllers. The failures exhibited by the AOANNs can be largely divided into two categories: dynamic failures, where the motions output by the ANN were incorrect or insufficient to balance the robot and systematic failures, where the ANN failed to output anything or caused the robot to stay stuck in one position. Both 800-hidden-neuron ANNs suffered from the latter type of failure, while PID-28 and Fuz-28 suffered from the former.

Figure 5.12 shows an example of the pose that was typical of the systematic failures; the robot would shift its CoM to one side as far as it could go while simultaneously inclining the torso in that direction. Once this pose was reached, the robot’s position would become stuck, with the ANN outputting only minor changes to the positions of the arms.

5.4.1 PID-800 and Fuz-800

Both 800-hidden-neuron ANNs failed to output any meaningful motions to the robot’s motors. The robot’s motions, when the robot moved at all, were jittery and irregular. This is attributable directly to the size of the ANN in two ways: over-fitting and low throughput.

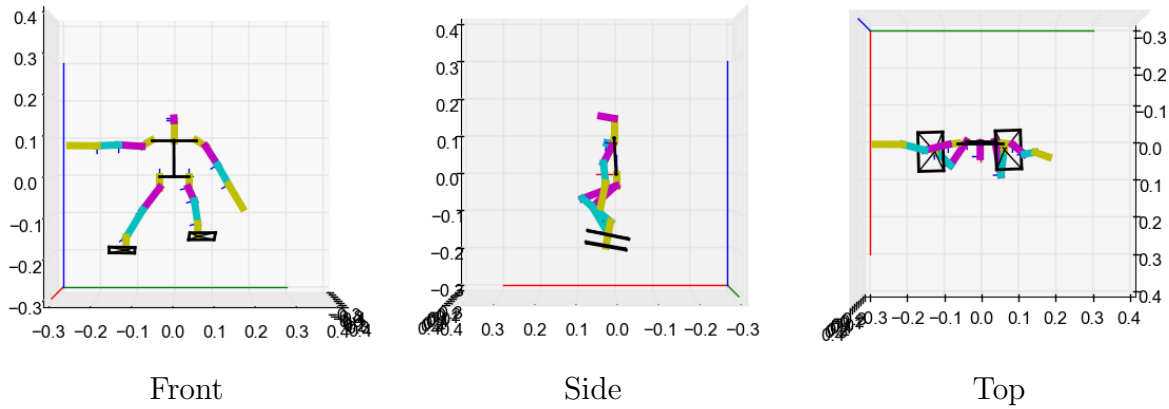


Figure 5.12: During trials with the AOANN the robot would frequently get stuck in an extreme leaning pose like the one pictured here

Because of the relatively high number of hidden neurons, combined with a coarser resolution for the single-hot-encoding, the ANN was likely over-trained, effectively memorising the training data. This over-training meant that the ANN could not generalise, becoming confused when the input data did not precisely match any of the training examples. This resulted in frequent instances where the ANN's output was seemingly random. If this random output produced no hot neurons (i.e. neurons with a positive output value) within the range for any given DoF, the control software would interpret this as a no-change command. If the random output produced multiple hot neurons, the positions represented by these neurons would be averaged using the value of the output neuron as a weight. Depending on the exact values and positions of these hot neurons one would expect irregular and jittery output to be produced.

Secondly, and more crucially, because of the size of the ANN and the number of connections ($14 \cdot 800 + 800 \cdot 1600 = 1,291,200$ total connections) the large ANNs are naturally slow to produce output. This inherent slowness is compounded when the online training is considered.

Assuming each connection in the ANN requires at least 5 clock cycles¹⁴ per epoch to train this implies a minimum of 64,560,000 clock cycles. Operating at 1.6GHz this means that the online training takes slightly more than 4ms per epoch. Given that we use 10 training epochs per system tick, the online training alone uses 40ms, far exceeding the 8ms window. Even reducing to a single training epoch per system tick, we are still left with approximately 4ms during which the software must receive the current sensor data from the sub-controller, calculate the output from the ANN, and send the new positions to the motors over a 1Mb serial connection.

The problems of low throughput could be partially mitigated by a multi-core CPU (such as the one present in the upcoming second-generation DARwIn-OP robot [24]) or by modifying the ANN library to run in parallel on a graphics processor. However, given the size and power requirements of a small humanoid robot, the PID-800 and Fuz-800 ANNs may simply be too large to be practical as a high-speed control system.

5.4.2 PID-28 and Fuz-28

Despite their relative success when compared with PID-800 and Fuz-800, the PID-28 and Fuz-28 ANNs were still less successful than the PID and Fuzzy Logic controllers. Both 28-hidden-neuron ANNs exhibited more instances of catapulting than the PID or Fuzzy Logic controllers (with Fuz-28 exhibiting the largest tendency to catapult).

The smaller ANNs did not suffer from the same throughput problems as the 800-hidden-neuron ANNs; the smaller number of connections ($14 \cdot 28 + 28 \cdot 3200 = 89,992$ total connections, approximately 7% the number of connections as the larger ANNs) allows both output calculation and online training to occur within the 8ms timing window allowed by the framework. Therefore the most likely cause of their failure when compared with the PID and Fuzzy Logic

¹⁴5 clock cycles is a very coarse approximation, but allows for two read/write cycles (at 2 clock cycles each) to load and store the weight, and one additional clock cycle to perform a single mathematical operation to change the current weight.

controllers is a combination of over-training and an insufficiently broad set of training data.

Both ANNs were trained using the Do the Shake policy. Therefore neither was exposed to the exceptionally high accelerometer spikes that resulted in the catapulting behaviour observed during trials with the Let’s Sway policy. These outliers were completely untrained, resulting in erroneous output from the ANN. Indeed, after any deck-strike the robot would move in an erratic fashion, regardless of the control policy used. The most common result of these erratic motions would be for the robot to lock itself in an extreme leaning position, such as the pose shown in Figure 5.12.

This lock-up is most-likely due again to an overly-narrow set of training data. Because the robot’s current and previous poses are used as inputs to the ANN, a combination of an extreme leaning posture combined with the low inclinations and angular velocities experienced during a manual reset were unlikely to have occurred during training. This untrained combination of inputs likely resulted in no hot output neurons, indicating “no-change” in the desired outputs.

5.4.3 Evaluation of Online Learning

Overall there was little observable improvement to the AOANNs’ performances between the first and last runs, indicating that online learning was of negligible benefit. Figure 5.13 shows the deck inclination and $\Delta\phi$ recorded during trials using the Do the Shake policy. As can be seen from this figure, PID-28 shows marginal improvement between the first and last trials; the slope of the negative bias is reduced in the last trial when compared with the first, indicating that some small improvement may have been learned.

Looking at the maximum, minimum, and average deck inclinations experienced by PID-28 during all trials with the Do the Shake policy, one can see that the deck inclination is slowly converging toward zero, as shown in Figure 5.14.

Fuz-28 did not exhibit any appreciable improvement that can be attributed to training.

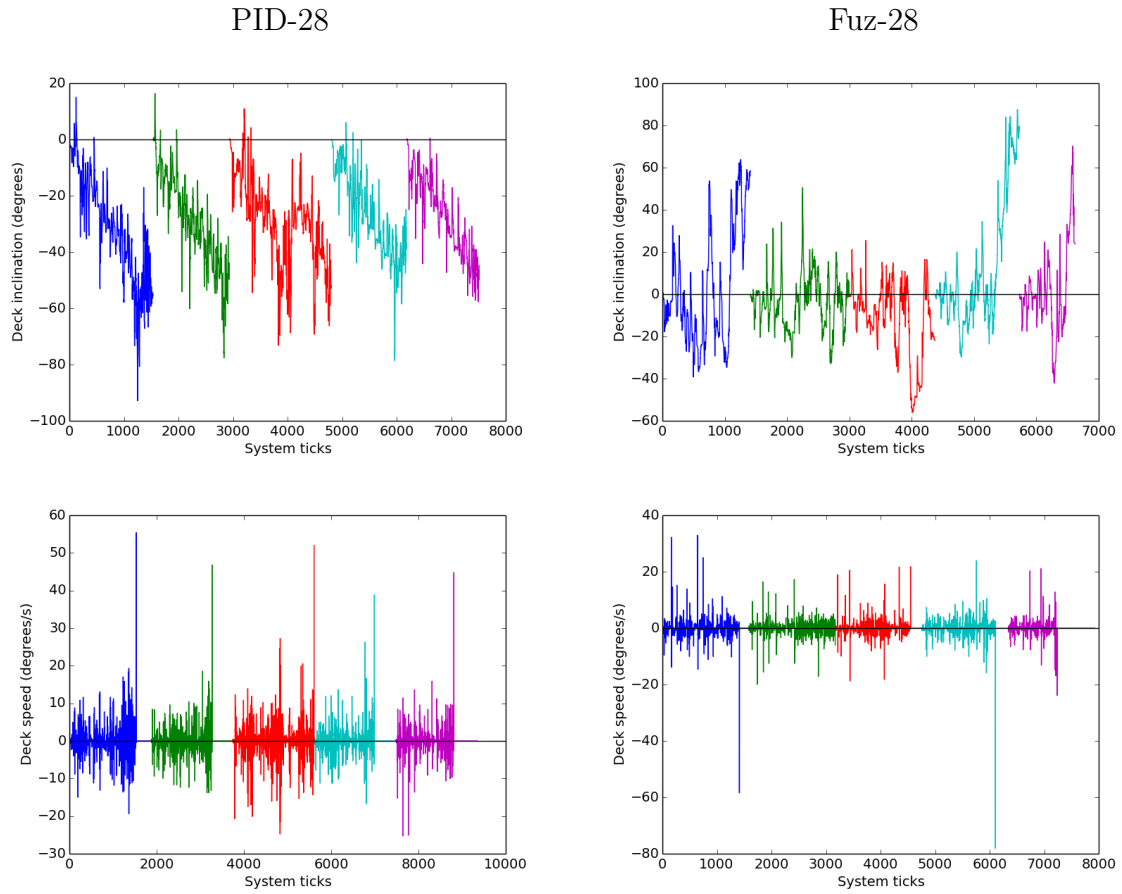


Figure 5.13: The inclination of the deck (top) and difference in inclination between ticks (bottom) across all five trials using both AOANNs and the Do the Shake policy.

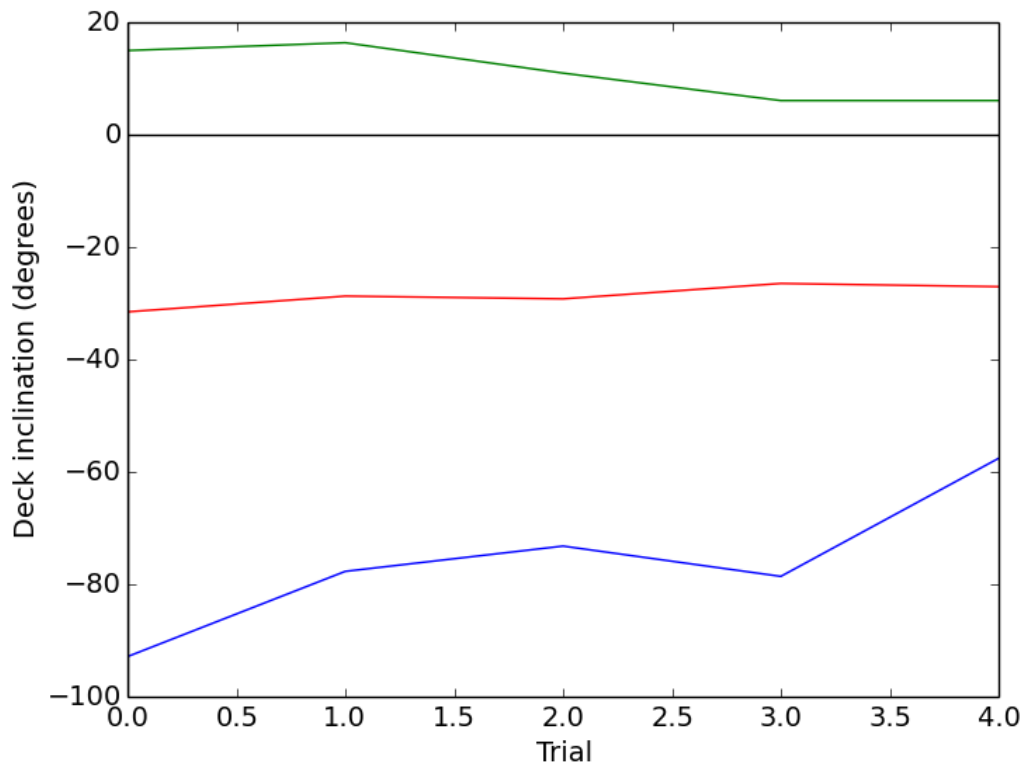


Figure 5.14: Max, min, and average deck inclination for each trial using PID-28 and the Do the Shake policy

This is likely due to the overall poor performance of Fuz-28. Because the training data used by Fuz-28 covered a narrower range of inputs, many output neurons were possibly trained to always output zero, regardless of the input. Overcoming this training bias would require many more epochs than the online training allowed.

PID-28 on the other hand had a broader range of training data, meaning that fewer output neurons suffered from an always-off training bias. This in turn allowed PID-28's performance to improve marginally as a result of the online training.

6 Applications, Future Work and Conclusions

This thesis has presented an evaluation of three control algorithms and two control policies intended to allow a humanoid robot to balance on a bongo board. Given the mechanical similarities between the cart-and-rod inverted pendulum and the bongo board, two well-known solutions to the cart-and-rod inverted pendulum problem – PID control and Fuzzy Logic – and one more novel solution – AOANNs – were modified to control a small humanoid robot’s arm positions, vertical and horizontal hip position, and torso inclination. Based on a mathematical analysis of the bongo board system it was determined that these five degrees of freedom are sufficient to counteract the effects of gravity on the system.

Each of the three control algorithms was evaluated when used with two different control policies: Do the Shake, which attempts to keep the deck of the bongo board level; and Let’s Sway, which induces a rhythmic oscillation to the deck in an attempt to keep the system in a state of dynamic stability. While the Let’s Sway policy was able to keep the robot’s torso marginally more stable, the improvement was not statistically significant.

Of the three control algorithms, both PID control and Fuzzy Logic were able to keep the bongo board system stable for short periods of time. The differences between the two algorithms was not statistically significant, but Fuzzy Logic was better-able to keep the bongo board’s deck level for longer periods of time.

The AOANNs, in contrast with PID control and Fuzzy Logic, were unable to control the bongo board system. However, the AOANNs did demonstrate improved performance as the trials continued, suggesting that with better initial training data and more time to learn while balancing on the bongo board their performance can be improved.

The research presented in this thesis offers several possible avenues of future work. Because balancing is central to many humanoid robotic tasks, the ability to balance on unstable terrain without relying on foot placement strategies may improve the usefulness of humanoid

robots in natural terrain.

6.1 AOANN Improvement

First and foremost, the AOANN-based balancing algorithms have much room for improvement. While the data shows that they AOANNs were successful at improving their performance while balancing, their overall performance was undeniably poor when compared to the PID and Fuzzy Logic controllers. Further experimentation to determine the appropriate number of hidden neurons and experimenting with different output encodings (e.g. binary encoding of the output layer) is warranted.

The AOANNs also suffered from relatively narrow training data. The creation of a complete physics model of the robot and bongo board that can be used in simulation to create larger data sets may be useful. While no simulation can completely replicate the randomness of the real-world, the ability to create training datasets of any size that are broadly representative of real-world physics would be helpful.

The combination of different numbers of hidden neurons with broader training data sets may reduce the amount of over-fitting presented by the ANNs, resulting in an overall performance improvement.

6.2 Expanded Fuzzy Logic Rules

While Yamakawa's rules for the the inverted pendulum offers a good baseline for a system to control the bongo board, the lack of any rules to specifically address failure conditions prevented the Fuzzy Logic controller from self-recovering from deck-strikes. By adding as few as two additional rules for situations where the deck's inclination is high it may be possible to allow the Fuzzy Logic controller to attempt to recover from a deck-strike. Given the ability of the Fuzzy Logic controller to keep the bongo board's deck level when using the

Do the Shake policy, further research into expanding the fuzzy rules is warranted.

6.3 Other Control Algorithms

While PID control, Fuzzy Logic, and AOANNs have all been shown to be effective at balancing the cart-and-rod inverted pendulum, they are by no means the only solutions that may be adaptable to the bongo board problem.

Reinforcement learning algorithms, such as Q-learning, like AOANNs, are able to improve their performance over time, learning to control the bongo board by trying to actively balance upon it. However, Q-learning is not reliant on a large initial training set, possibly avoiding the problems of over-fitting or insufficient breadth in the training set encountered by the AOANNs implemented for this thesis.

6.4 Hardware and Framework Improvements

The largest hardware-related issue encountered during this research was the inability to process large ANNs within the 8ms timing window allowed by the DARwIn-OP's software framework. The fact that the FANN library used to implement the ANNs does not support parallel processing on the GPU greatly hindered the performance of these algorithms.

Equipping the robot with a more powerful CPU and implementing balancing algorithms to take advantage of a general-purpose GPU (e.g. using a GPU that supports Nvidia's CUDA architecture for general-purpose computing) may improve the robot's reaction time. This faster reaction time may in turn improve the robot's ability to balance.

6.5 Improved Sensor Filtering

Each balancing algorithm takes sensor data accumulated during the previous 8ms window as its primary input. While these sensor data are filtered using an n -tab median filter,

the result is still very noisy. Relying on the accelerometer data to estimate inclination is especially unreliable, as the sum of the robot’s acceleration along all three axes does not always add up to g , and large changes in the robot’s perceived inclination can result.

These issues can be mitigated through an improved sensor filtering system, possibly with the inclusion of additional sensor data. For example, using the robot’s camera to estimate the angle of the horizon may improve the accuracy of the inclination estimation. Equipping the robot with additional accelerometers and gyroscopes to provide additional data points may also help reduce sensor noise by allowing the software to take an average of all readings.

6.6 Alpine and Cross-Country Skiing

In early 2015 the University of Manitoba’s Autonomous Agents Lab – the laboratory in which the research for this thesis was conducted – developed software to allow a small humanoid robot to ski downhill and cross-country [12, 21, 23, 30, 31]. This skiing research was the first application of the balancing algorithms presented in this thesis to a non-bongo-board problem. The skiing research was presented at the 2015 ICRA Humanoids Application Challenge in Seattle, WA, where it won first place in the challenge. [43]

Snow, as a natural material, does not present a uniform surface to traverse. Changes in inclination are common, as hills curve up and down. Snow may also compress unevenly underfoot, producing small, local changes in the inclination underfoot.

To allow the robot to balance on snow of any inclination we used two PID-based systems using the Do the Shake control policy; one controller is responsible for the robot’s balance in the frontal plane, just as with the bongo board, while the second is responsible for the balance in the sagittal plane. [22]

Because the robot’s arms are holding ski poles used to assist in steering the robot’s arms are not used to provide fine corrections; instead the robot’s ankle motors are used to make gross corrections, with the hip motors to provide finer adjustments. By moving the robot’s



Figure 6.1: Jennifer the robot equipped with skis demonstrating alpine skiing. We use the Do the Shake policy to control the robot’s pitch and roll while skiing. This skiing research won the 2015 ICRA Humanoids Application Challenge. (Winnipeg Free Press [30])

ankles the robot’s entire body moves, producing the largest changes in the position of the robot’s CoM. Keeping the legs still while moving the torso with the hip motors produces smaller changes in the CoM’s position.

By using lower P and D gains than in the bongo board research the robot was adequately able to keep its CoM positioned over the skis, allowing it to ski downhill and cross-country across snow of varying inclinations. [21]

6.7 Ice Skating

In 2012 the Autonomous Agents Lab developed Jennifer¹⁵, the world’s first ice skating, hockey playing humanoid robot [26, 27, 36], pictured in Figure 6.2. This research into the mechanics of ice skating and the implementation of a simple ice skating gait won the 2012 ICRA DARwIn-OP Humanoid Application Challenge. [28, 42]

¹⁵Jennifer was named after Jennifer Botterill, a five-time world champion and three-time Olympic gold medallist in women’s ice hockey.



Figure 6.2: Jennifer, the world’s first ice skating and hockey-playing humanoid robot on an outdoor skating rink (left, Iverach-Brereton et al. [26]), and skating indoors with members of the University of Manitoba Bisons hockey team (right, Daily Planet [36]). The research into ice skating won the 2012 ICRA Humanoids Application Challenge.

During the research into ice skating we discovered that the robot was unable to balance on a single skate blade for any significant length of time, preventing the robot from sustaining a glide phase while skating. As such the robot’s gait is more akin to shuffling on skates than a true skating gait.

Ice skates, like the bongo board, provide good stability in the sagittal plane and little support in the frontal plane. The balancing techniques used on the bongo board should be applicable to ice skating, helping to create a true skating gait.

6.8 Wobble Board

Given the success of the alpine skiing’s three-dimensional implementation of the Do the Shake control policy, the control algorithms and policies presented in this thesis may be suitable to other, higher-dimensional balancing problems. The wobble board may serve as a good benchmark balancing problem for such higher-dimensional problems.

The wobble board is similar to the bongo board, but uses a circular deck and a spherical wheel, as shown in Figure 6.3. The use of a spherical wheel instead of a cylindrical one means that the rider must control inclination in both the frontal and sagittal planes, as well

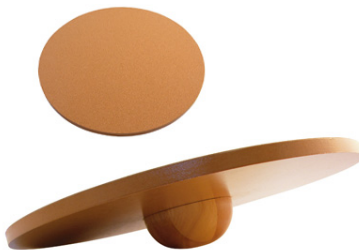


Figure 6.3: A commercially-made wobble board with a fixed spherical wheel intended for use in physiotherapy. (Spinal Products [37])

as yaw about the vertical axis. Simultaneous control in the frontal and sagittal planes has already been demonstrated with our research into alpine and cross-country skiing¹⁶; all that is required is the addition of a system to allow the robot to control its rotation about the vertical axis.

6.9 Combining with Foot Placement Strategies

Foot placement strategies, such as the capture point technique described by Pratt et al. [32] have been shown to be effective at compensating for external forces applied to the robot (e.g. pushes, sudden changes in terrain). However, these strategies are often computationally intensive and therefore slower for the robot to execute. Furthermore they rely entirely on a strong ground reaction force to arrest the robot's movement.

A simple PID or Fuzzy Logic controller, such as those demonstrated in this thesis, may be able to provide a rapid-response balancing subroutine for use while the robot performs the more intense calculations required for foot placement techniques to be effective. A PID or Fuzzy Logic system requires minimal resources to operate, meaning a decision can be made very quickly in dynamic environments.

If a foot placement strategy inadvertently places the foot on unstable ground (e.g. rotting

¹⁶See Section 6.6.

or broken floors in an urban rescue environment, a patch of ice outdoors in winter) the ground reaction force alone may be insufficient to prevent the robot from falling. In such instances a fall-back balancing system that does not rely on moving the feet but instead uses the robot's own body to produce reactionary forces could be all that keeps the robot upright and able to operate normally.

6.10 Conclusion

The research in this thesis shows that well-known solutions to the cart-and-rod inverted pendulum problem – namely Proportional-Integral-Derivative (PID) controllers and Fuzzy Logic – are suitable for the much more dynamic bongo board problem. Artificial neural networks, a more novel solution to the inverted pendulum problem, were also evaluated but did not balance the bongo board as successfully as PID or Fuzzy Logic control.

This thesis also shows that two control policies developed – Do the Shake, which uses a linear predictive sensor model, and Let's Sway, which introduces a rhythmic oscillation to the system – are both suitable for the bongo board problem. The rhythmic oscillation introduced by the Let's Sway control policy reduced the angular velocity of the robot's torso at the cost of increased angular velocity in the deck when compared with the Do the Shake policy. While the improved torso stability when using the Let's Sway policy is notable, the improvement is not statistically significant.

Through an analysis of the forces exerted on the bongo board, both by gravity and by the rider, I have provided equations that describe the motion of the bongo board system. By developing algorithms that monitor the external forces and control the rider's motions to produce opposing forces I have shown that a small humanoid robot is able to independently balance on a bongo board for short periods of time.

Broadly-speaking the Fuzzy Logic controller was better-able to control the motion of the bongo board, but the rules developed did not allow the robot to self-recover reliably when

the deck struck the ground. The PID controller, while suffering a slightly worse performance than the Fuzzy Logic controller, was able to benefit from the rapid deceleration experienced during a deck-strike, and could bounce the bongo board back into a near-stable position.

While many problems remain to be solved in robotics, this thesis has demonstrated new, effective techniques for use in balancing problems for humanoid robots on dynamic terrain. The bongo board, an easy-to-replicate example of unstable, dynamic terrain, is a novel application for humanoid robots, with little existing research dedicated to it. It is however comparable to the well-known cart-and-rod inverted pendulum problem and can benefit from existing research in this area.

Based on an analysis of the bongo board as an inverted pendulum, I have developed control algorithms based on staple solutions to the cart-and-rod problem – PID and Fuzzy Logic control – and shown experimentally that they are able to keep a small humanoid robot stable on highly dynamic terrain. The utility of these balancing algorithms has already been demonstrated with an alpine skiing robot able to balance on natural snow. These algorithms are applicable to a range of additional balancing tasks, advancing the state-of-the-art of humanoid robotic balancing.

References

- [1] C. W. Anderson. Learning to control an inverted pendulum using neural networks. *Control Systems Magazine, IEEE*, 9(3):31–37, 1989.
- [2] S. Anderson and J. Hodgins. Adaptive torque-based control of a humanoid robot on an unstable platform. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 511–517, 2010.
- [3] M. Araki. PID control. *Control Systems, Robotics and Automation*, 2, 2002.
- [4] J. Baltes, C. Iverach-Brereton, and J. Anderson. Sensor filtering for balancing of humanoid robots in highly dynamic environments. In *Automatic Control Conference (CACS), 2013 CACS International*, pages 170–173, Dec 2013.
- [5] J. Baltes. Lecture notes for comp 4360 machine learning: Neural networks. http://www4.cs.umanitoba.ca/~jacky/Teaching/Courses/COMP_4360-MachineLearning/current/Lectures/L04_NeuralNetworks.pdf, February 2012.
- [6] J. Baltes, C. Iverach-Brereton, and J. Anderson. Human inspired control of a small humanoid robot in highly dynamic environments or Jimmy Darwin rocks the bongo board. In R. A. C. Bianchi, H. L. Akin, S. Ramamoorthy, and K. Sugiura, editors, *RoboCup 2014: Robot World Cup XVIII*, volume 8992 of *Lecture Notes in Computer Science*, pages 466–477. Springer International Publishing, 2015.
- [7] J. Baltes, K.-Y. Tu, and S. L. Lip. *HuroCup Competition*. FIRA, Sept. 2013.
- [8] J. Baltes, K.-Y. Tu, and S. L. Lip. *FIRA HuroCup Rules*. FIRA, Sept. 2014.
- [9] A. Blum. *Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems*. John Wiley & Sons Canada, Ltd, 1992.
- [10] M. K. Bugeja. Non-linear swing-up and stabilizing control of an inverted pendulum system. In *The IEEE Region 8 Eurocon 2003. Computer as a Tool*, volume 2, pages 437 – 441, 22–24 September 2003.
- [11] C. E. Clauser, J. T. McConville, and J. Young. Weight, volume, and center of mass of segments of the human body. Aerospace Medical Research Laboratory, Aerospace Medical Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, August 1969.
- [12] D. Da Silva. Trading in skates for skis: Jennifer the skating robot learns a new sport. <http://www.winnipegfreepress.com/our-communities/lance/Trading-in-skates-for-skis-291419281.html>, February 10, 2015.

- [13] L. da Vinci. Vitruvian man. http://commons.wikimedia.org/wiki/File:Da_Vinci_Vitruve_Luc_Viatour.jpg, c. 1492, Downloaded April 22, 2014.
- [14] S. N. et al. Neural networks on the GPU. http://leenissen.dk/fann/html_latest/files2/gpu-txt.html, Accessed December 1, 2014.
- [15] S. N. et al. Fann datatypes. http://leenissen.dk/fann/html/files/fann_data-h.html, Accessed June 5, 2014.
- [16] S. N. et al. Fast artificial neural network library. <http://leenissen.dk/fann/wp/>, Accessed June 5, 2014.
- [17] M. E. Harmon and S. S. Harmon. Reinforcement learning: A tutorial. *WL/AAFC, WPAFB Ohio*, 45433, 1996.
- [18] J. Heaton. *Introduction to Neural Networks for Java, 2nd Edition*. Heaton Research, Inc., 2008.
- [19] M. Hehn and R. D’Andrea. A flying inverted pendulum. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 763–770. IEEE, 2011.
- [20] M. Howell and M. Best. On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata. *Control Engineering Practice*, 8:147–154, 2000.
- [21] C. Iverach-Brereton. Jennifer Darwin carves the slopes. <https://youtu.be/XU17sbItYxI>, Uploaded 28 Maym 2015.
- [22] C. Iverach-Brereton. Active balancing for alpine skiing. https://youtu.be/NxDfWx_4ywc, Uploaded February 6, 2015.
- [23] C. Iverach-Brereton. Jennifer goes cross-country and alpine skiing. <https://youtu.be/3jesBT-xp1A>, Uploaded January 30, 2015.
- [24] C. Iverach-Brereton. Dismantling the DARwIn-OP 2.0. <http://youtu.be/c4H3XU10cJs>, Uploaded January 9, 2015.
- [25] C. Iverach-Brereton. Jimmy Darwin rocking the bongo board. <http://youtu.be/ia2ZYqqF-1w>, Uploaded May 7, 2013.
- [26] C. Iverach-Brereton, J. Baltes, J. Anderson, A. Winton, and D. Carrier. Gait design for an ice skating humanoid robot. *Robotics and Autonomous Systems*, 62(3):306–318, 2012.
- [27] C. Iverach-Brereton, A. Winton, and J. Baltes. Ice skating humanoid robot. In G. Herrmann, M. Studley, M. Pearson, A. Conn, C. Melhuish, M. Witkowski, J.-H. Kim, and P. Vadakkepat, editors, *Advances in Autonomous Robotics*, volume 7429 of *Lecture Notes in Computer Science*, pages 209 – 219. Springer Berlin / Heidelberg, 2012.

- [28] Jenny Ford. Hockey-playing robot scores gold. <http://www.winnipegfreepress.com/local/hockey-playing--robot-scores-gold-159013585.html>, Accessed 12 June, 2015.
- [29] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 239 – 246, Maui, HI, 29 October – 3 November 2001.
- [30] N. Martin. Robot goes from hockey skates to skis. *Winnipeg Free Press*, February 2, 2015 A2.
- [31] Metro News Staff. Video: Not satisfied with just hockey, Jennifer the robot goes cross-country, alpine skiing. <http://metronews.ca/news/winnipeg/1278140/video-not-satisfied-with-just-hockey-jennifer-the-robot-goes-cross-country-alpine-skiing/>, February 3, 2015.
- [32] J. E. Pratt, J. Carff, S. V. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *Proceedings of the 2006 IEEE-RAS International Conference on Humanoid Robotics, Genoa, Italy*, pages 200 – 207, 6–7 December 2006.
- [33] RoboSavvy. RoboSavvy – Experimenting with DARwIn: Part 3 Joint Layout and Kinematics . http://robosavvy.com/site/index.php?option=com_content&task=view&id=221, Accessed April 20, 2012.
- [34] Robotis. DARwIn-OP : Robotis. http://www.robotis.com/xe/darwin_en, Accessed April 19, 2012.
- [35] Robotis. MX-28T/MX-28R. support.robotis.com/en/product/dynamixel/mx_series/mx-28.htm, Accessed January 27, 2012.
- [36] B. Schaub (producer). Daily Planet. Broadcast on Discovery Channel, March 6 2012, March 6, 2012.
- [37] Spinal Products. Wobble board cork. http://www.spinalproducts.co.uk/product.asp?P_ID=228, Accessed 20 March, 2015.
- [38] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modelling and Control*, chapter 3, pages 61–82. Wiley, 2005.
- [39] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modelling and Control*, chapter 4, pages 83–98. Wiley, 2005.
- [40] T. Sugihara, Y. Nakamura, and H. Inoue. Realtime humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*, volume 2, pages 1404 – 1409, Washington, DC, May 2002.

- [41] K. Sultan and A. Mirza. *Inverted Pendulum Analysis, Design and Implementation*. IEEE Visionaries, 2003.
- [42] University of Manitoba. Winner, ICRA 2012 DARwIn-OP humanoid application challenge. <http://aalab.cs.umanitoba.ca/index.php/9-news/11-winner-icra-2012-darwin-op-humanoid-application-challenge>, Accessed 12 June, 2015.
- [43] University of Manitoba. Jennifer wins 2015 DARwIn-OP humanoid application challenge. <http://aalab.cs.umanitoba.ca/index.php/9-news/54-jennifer-wins-2015-darwin-op-humanoid-application-challenge>, Accessed 28 May, 2015.
- [44] University of Manitoba. Autonomous Agents Laboratory. <http://aalab.cs.umanitoba.ca/>, Accessed February 12, 2013.
- [45] J.-J. Wang. Simulation studies of inverted pendulum based on PID controllers. *Simulation Modelling Practice and Theory*, 19(1):440 – 449, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.
- [46] L.-X. Wang. Stable adaptive fuzzy controllers with application to inverted pendulum tracking. *IEEE Transactions of Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(5):677 – 691, October 1996.
- [47] Wikimedia Commons. Bodyplanes.jpg. <http://upload.wikimedia.org/wikipedia/commons/3/34/BodyPlanes.jpg>, Accessed February 26, 2013.
- [48] Wikimedia Commons. PID en updated feedback.svg. http://commons.wikimedia.org/wiki/File:PID_en_updated_feedback.svg, Accessed June 2, 2013.
- [49] Wikimedia Commons. Colored neural network.svg. http://commons.wikimedia.org/wiki/File:Colored_neural_network.svg, Accessed September 10, 2014.
- [50] Q. Wu and A. Pugh. Reinforcement learning control of unknown dynamic systems. *Control Theory and Applications, IEE Proceedings D*, 140:313–322, 1993.
- [51] T. Yamakawa. Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system. *Fuzzy Sets and Systems*, 32(2):161 – 180, 1989.
- [52] J. Yang and V. Honavar. Lecture notes on artificial neural networks. <http://home.sogang.ac.kr/sites/gsinfotech/study/study007/Lists/b6/Attachments/35/neuralnets.pdf>, Accessed November 28, 2014.
- [53] J. Zhong. PID controller tuning: A short tutorial. Lecture Slides, Mechanical Engineering, Purdue University <http://saba.kntu.ac.ir/eedcd/pcl/download/PIDtutorial.pdf>, Accessed September 9, 2013.

- [54] J. Ziegler and N. Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.

Appendices

A Deck Inclination During All Trials

The following graphs show the inclination of the bongo board's deck during all trials. These graphs are stacked in Figures 5.2 to 5.5.

A.1 Do the Shake Policy

This section shows the deck inclination and $\Delta\phi$ during all trials using the Do the Shake policy.

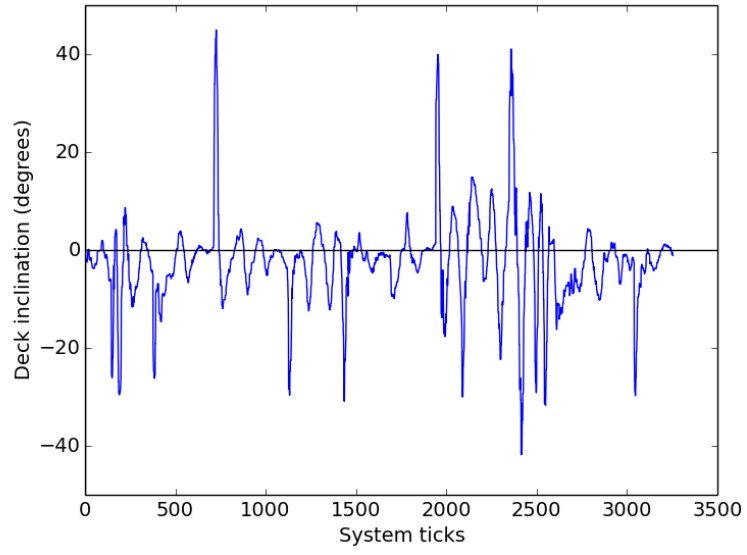


Figure A.1: Deck inclination during trial 1 with PID controller using Do the Shake policy

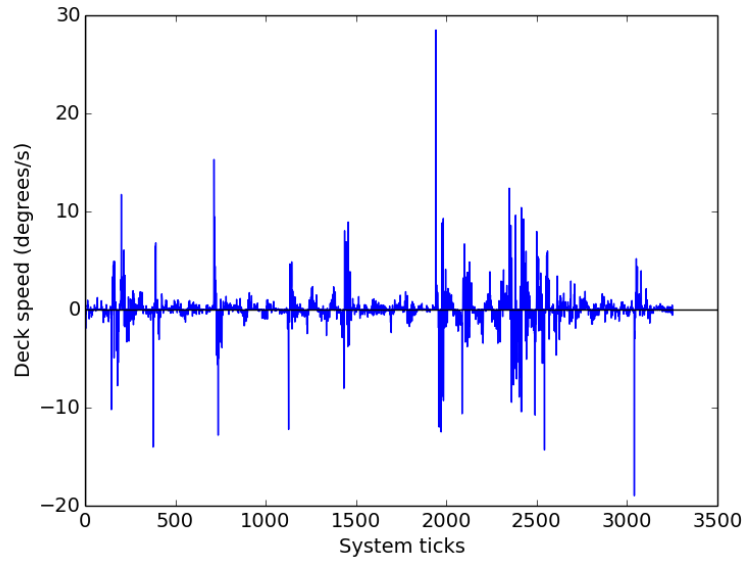


Figure A.2: Difference in deck inclination ($\Delta\phi$) during trial 1 with PID controller using Do the Shake policy

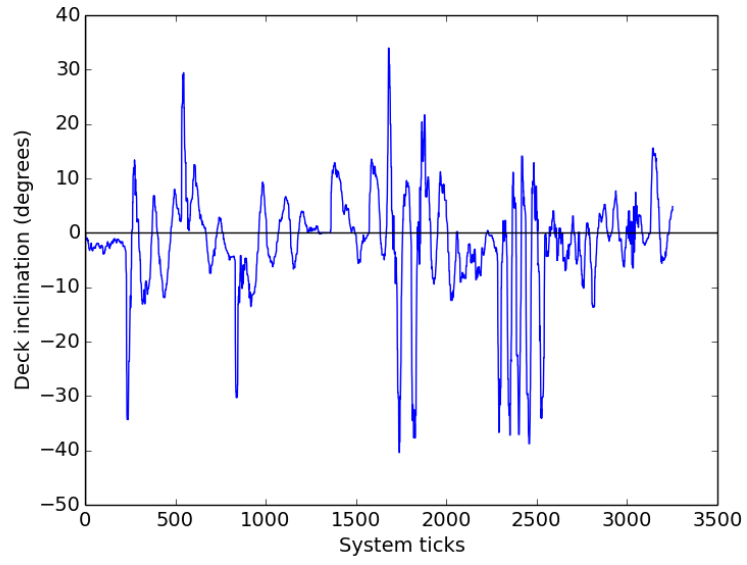


Figure A.3: Deck inclination during trial 2 with PID controller using Do the Shake policy

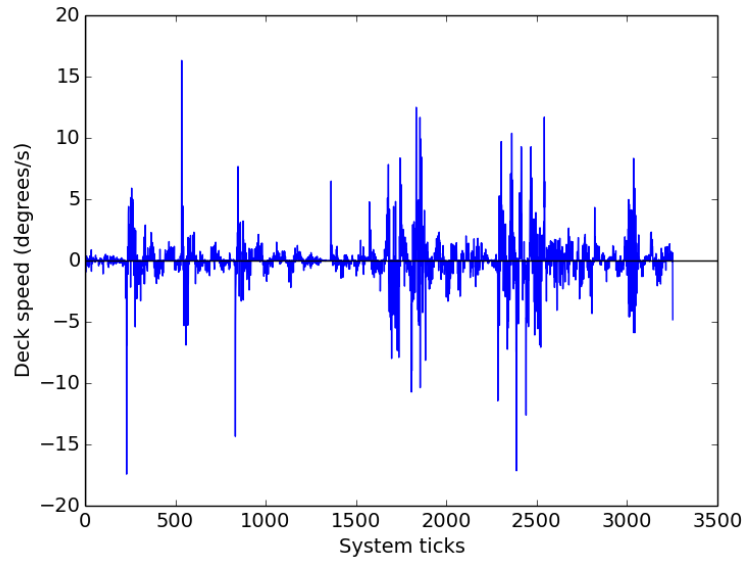


Figure A.4: Difference in deck inclination ($\Delta\phi$) during trial 2 with PID controller using Do the Shake policy

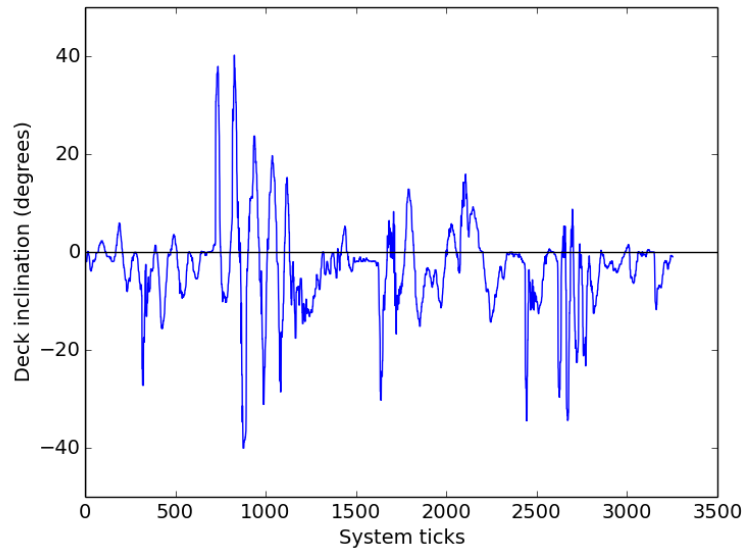


Figure A.5: Deck inclination during trial 3 with PID controller using Do the Shake policy

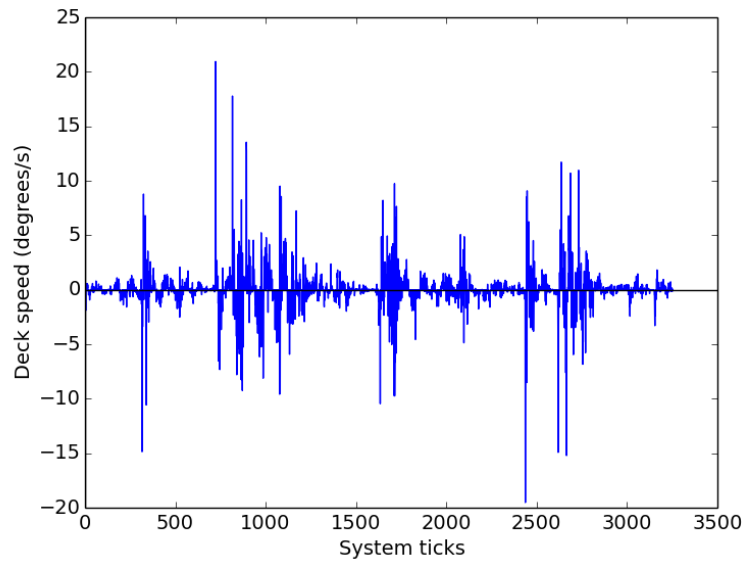


Figure A.6: Difference in deck inclination ($\Delta\phi$) during trial 3 with PID controller using Do the Shake policy

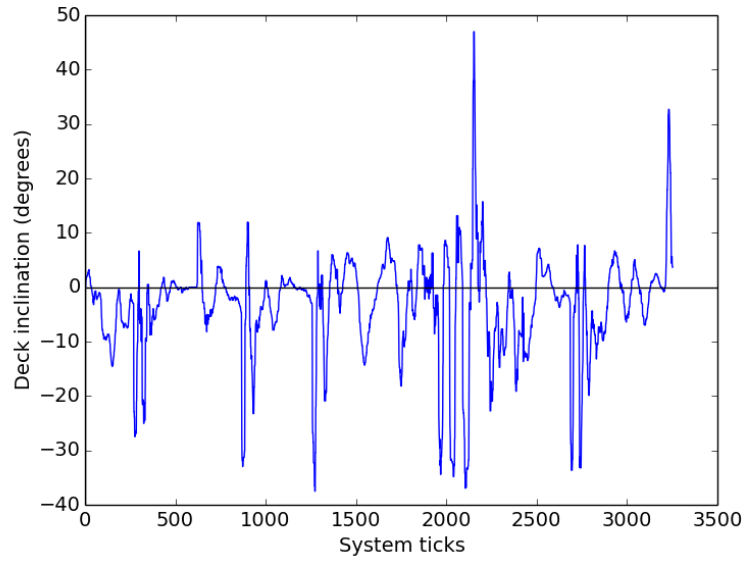


Figure A.7: Deck inclination during trial 4 with PID controller using Do the Shake policy

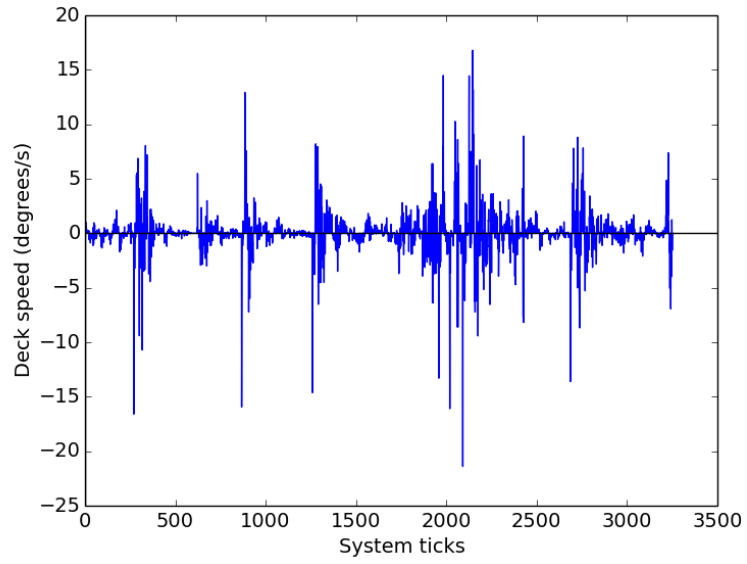


Figure A.8: Difference in deck inclination ($\Delta\phi$) during trial 4 with PID controller using Do the Shake policy

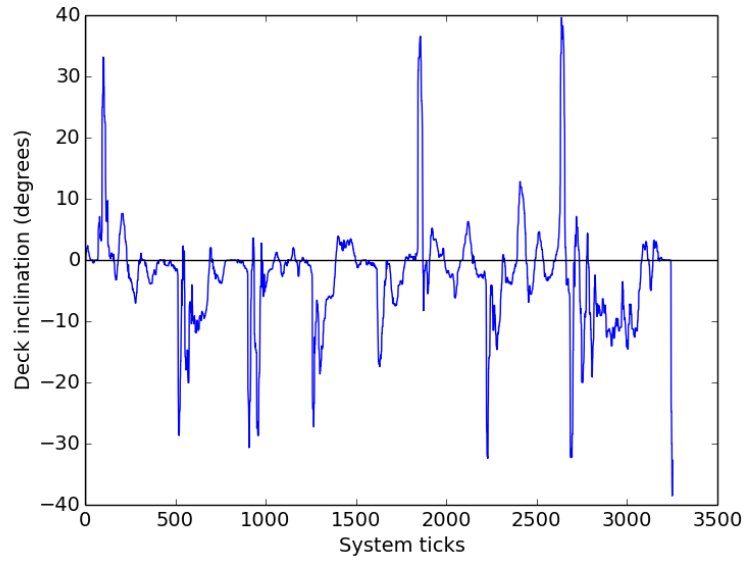


Figure A.9: Deck inclination during trial 5 with PID controller using Do the Shake policy

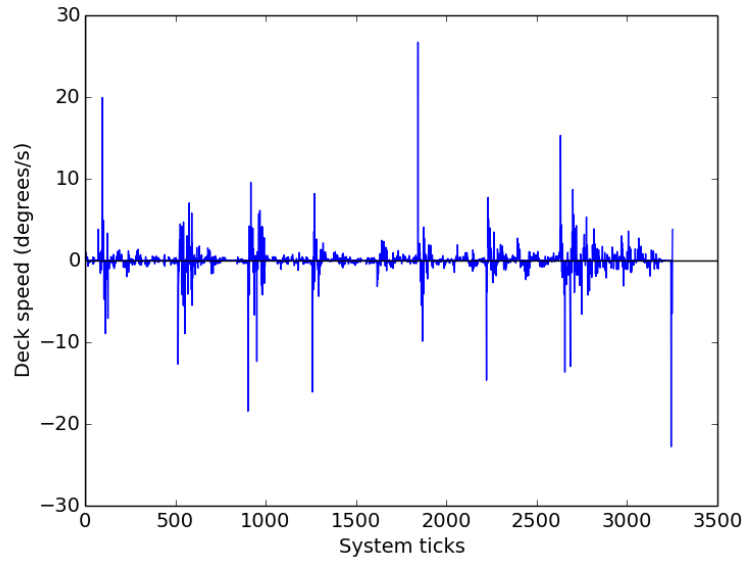


Figure A.10: Difference in deck inclination ($\Delta\phi$) during trial 5 with PID controller using Do the Shake policy

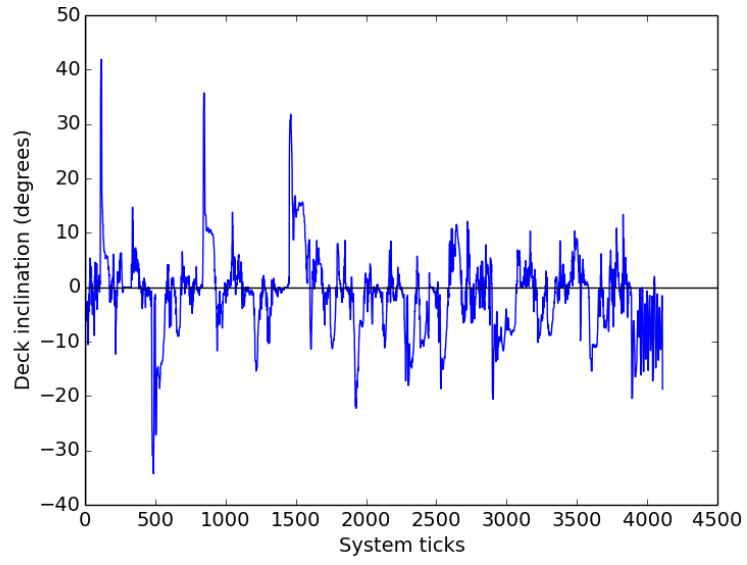


Figure A.11: Deck inclination during trial 1 with Fuzzy Logic controller using Do the Shake policy

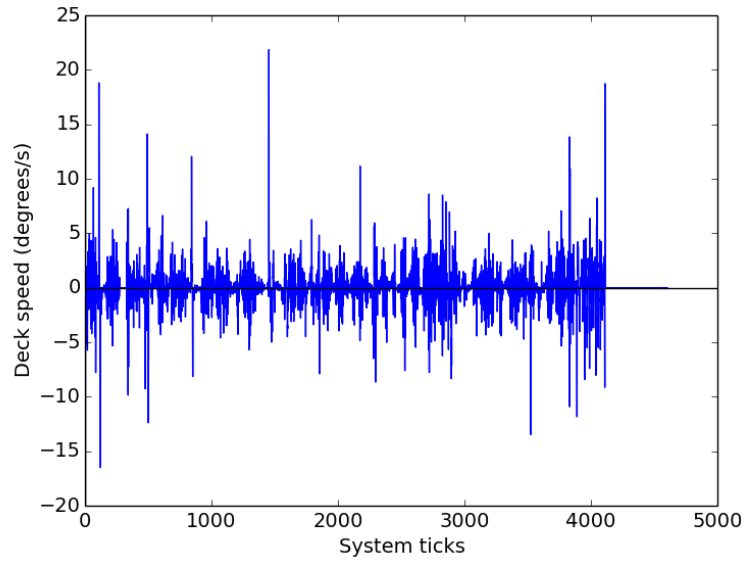


Figure A.12: Difference in deck inclination ($\Delta\phi$) during trial 1 with Fuzzy Logic controller using Do the Shake policy

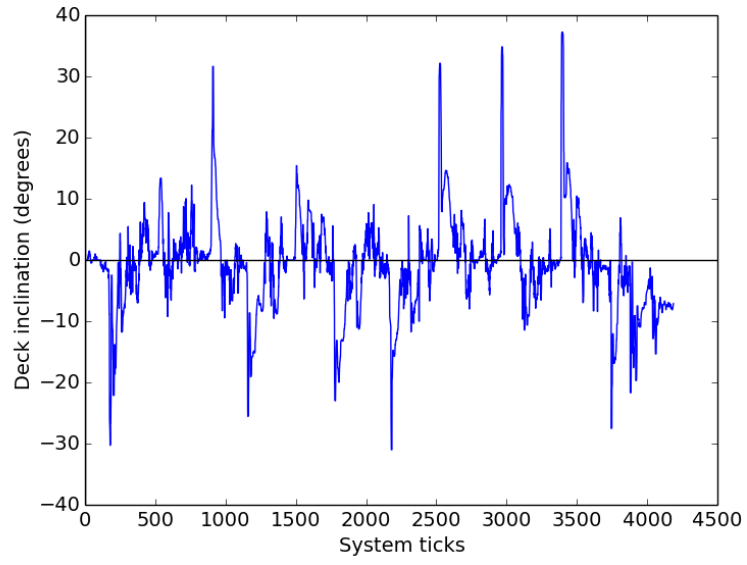


Figure A.13: Deck inclination during trial 2 with Fuzzy Logic controller using Do the Shake policy

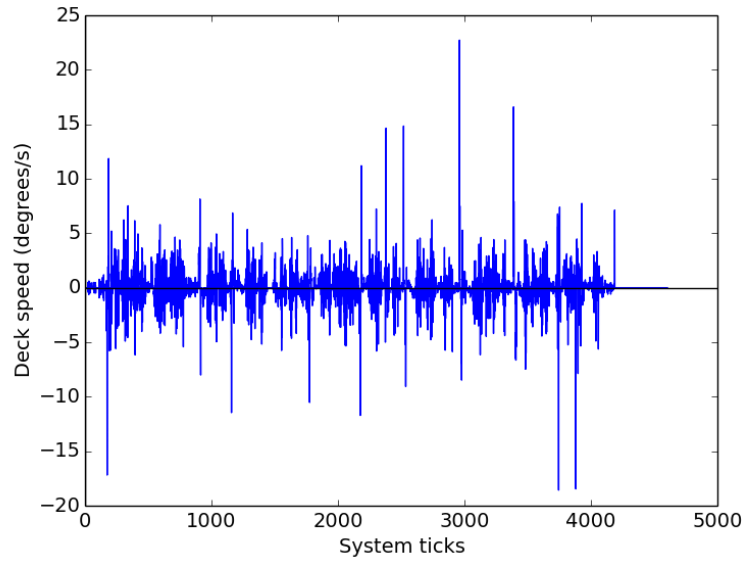


Figure A.14: Difference in deck inclination ($\Delta\phi$) during trial 2 with Fuzzy Logic controller using Do the Shake policy

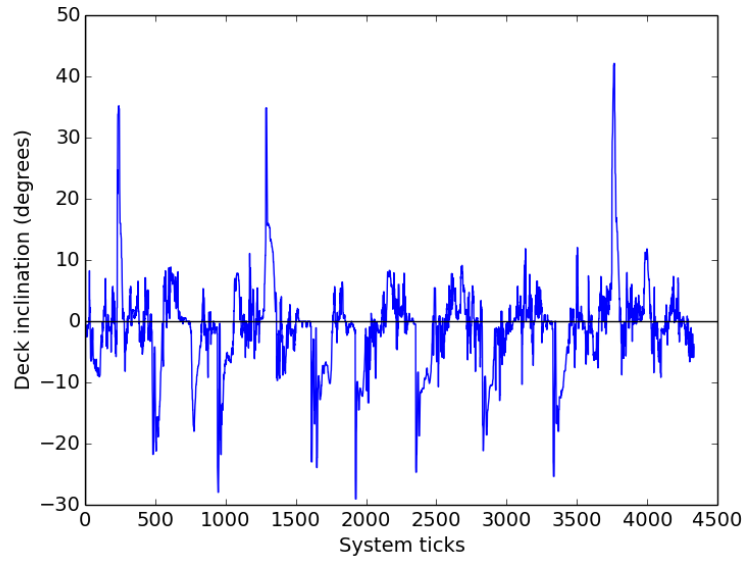


Figure A.15: Deck inclination during trial 3 with Fuzzy Logic controller using Do the Shake policy

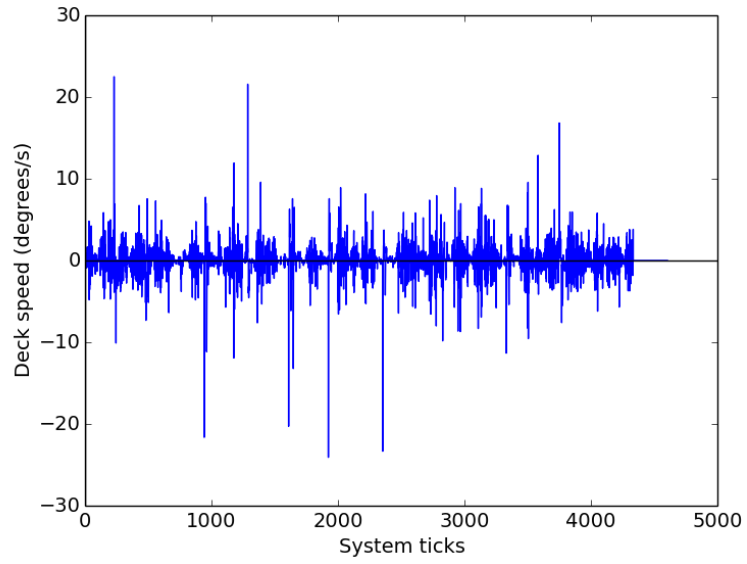


Figure A.16: Difference in deck inclination ($\Delta\phi$) during trial 3 with Fuzzy Logic controller using Do the Shake policy

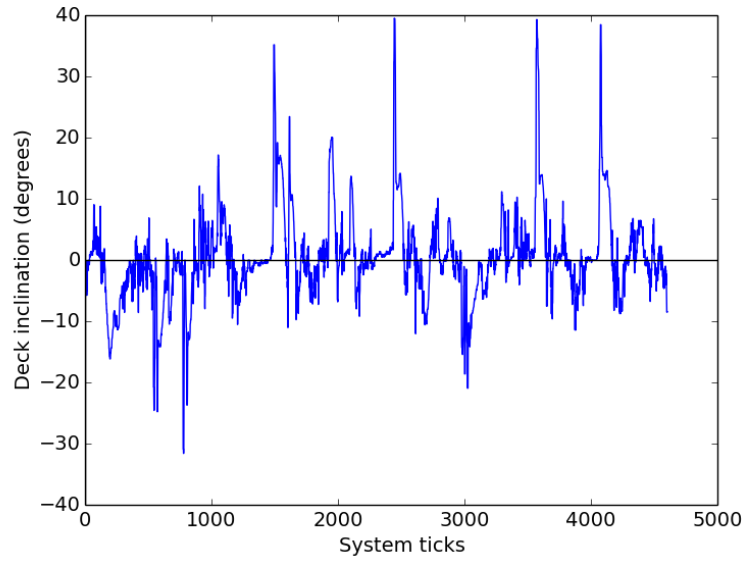


Figure A.17: Deck inclination during trial 4 with Fuzzy Logic controller using Do the Shake policy

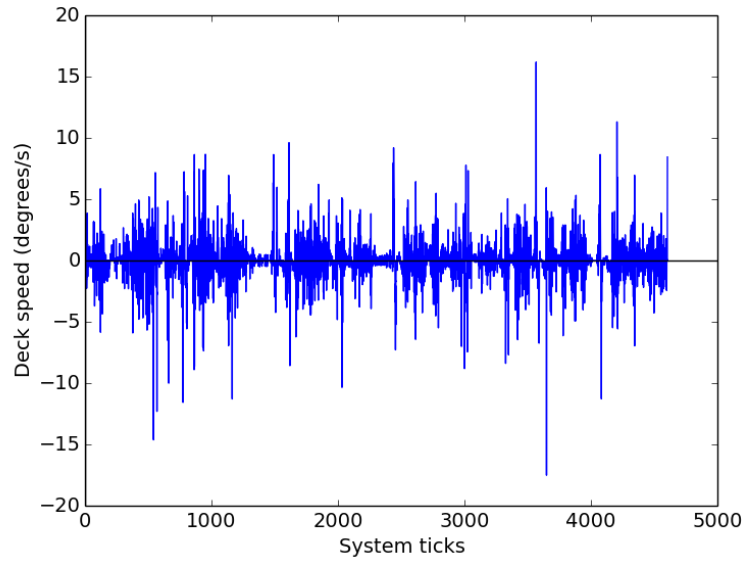


Figure A.18: Difference in deck inclination ($\Delta\phi$) during trial 4 with Fuzzy Logic controller using Do the Shake policy

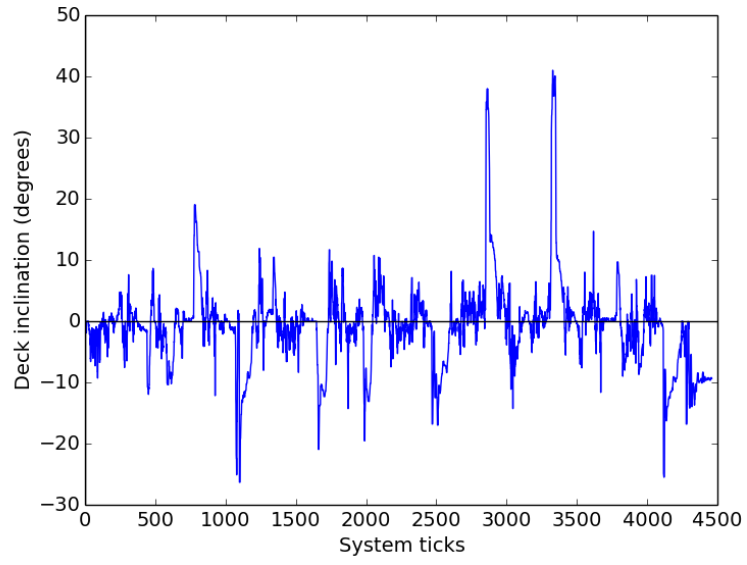


Figure A.19: Deck inclination during trial 5 with Fuzzy Logic controller using Do the Shake policy

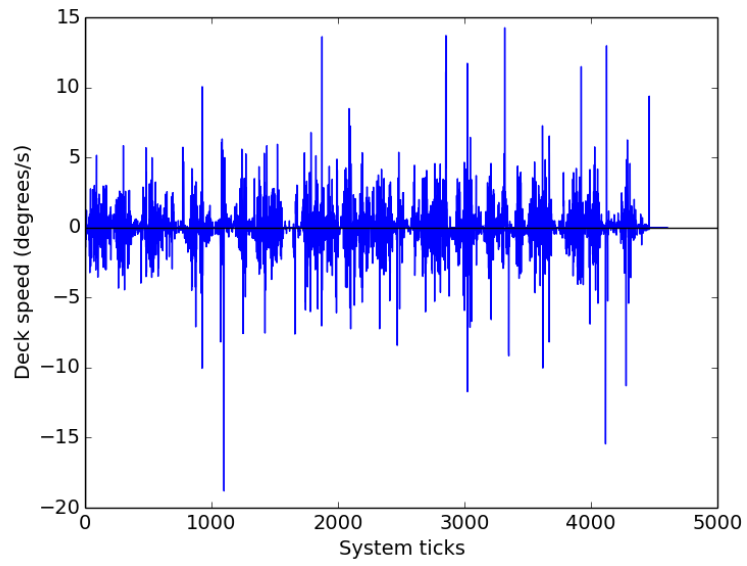


Figure A.20: Difference in deck inclination ($\Delta\phi$) during trial 5 with Fuzzy Logic controller using Do the Shake policy

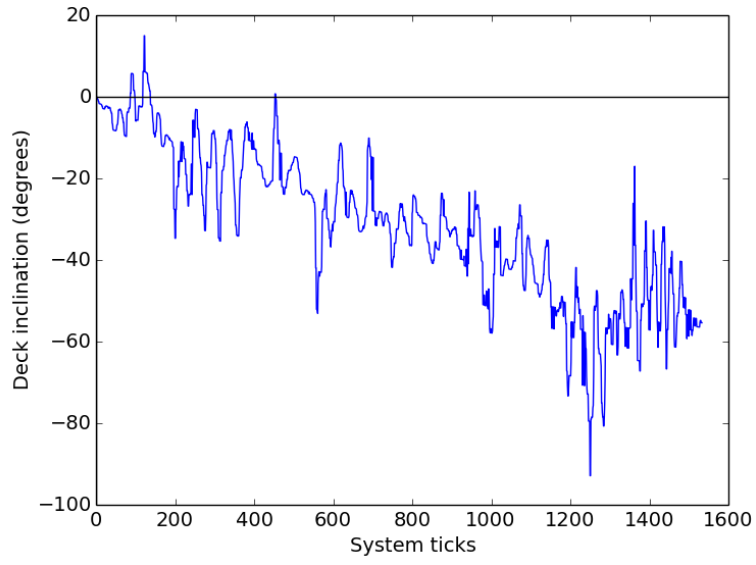


Figure A.21: Deck inclination during trial 1 with AOANN PID-28 controller using Do the Shake policy

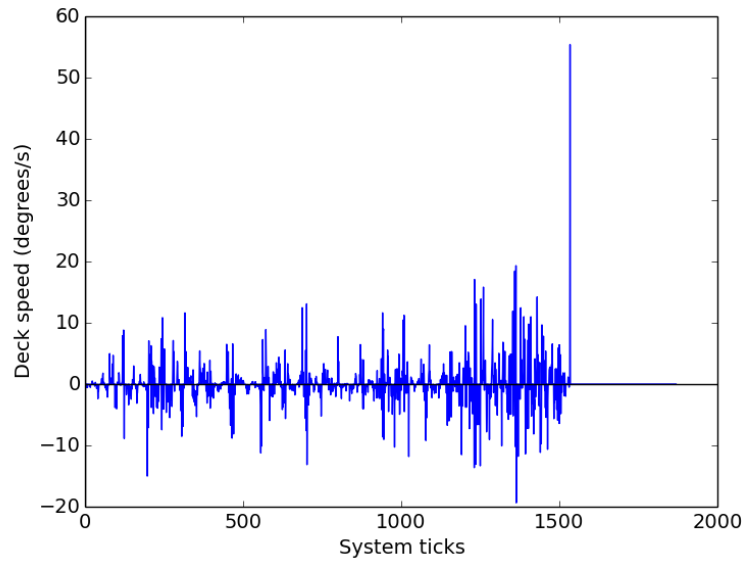


Figure A.22: Difference in deck inclination ($\Delta\phi$) during trial 1 with AOANN PID-28 controller using Do the Shake policy

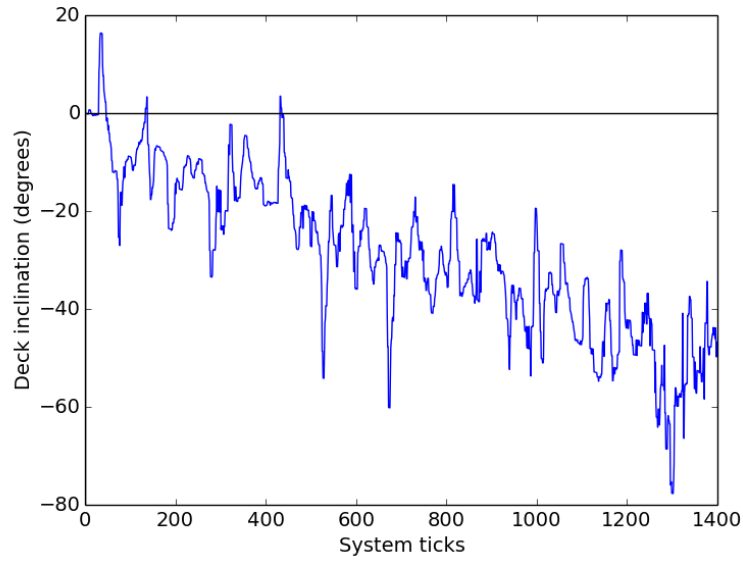


Figure A.23: Deck inclination during trial 2 with AOANN PID-28 controller using Do the Shake policy

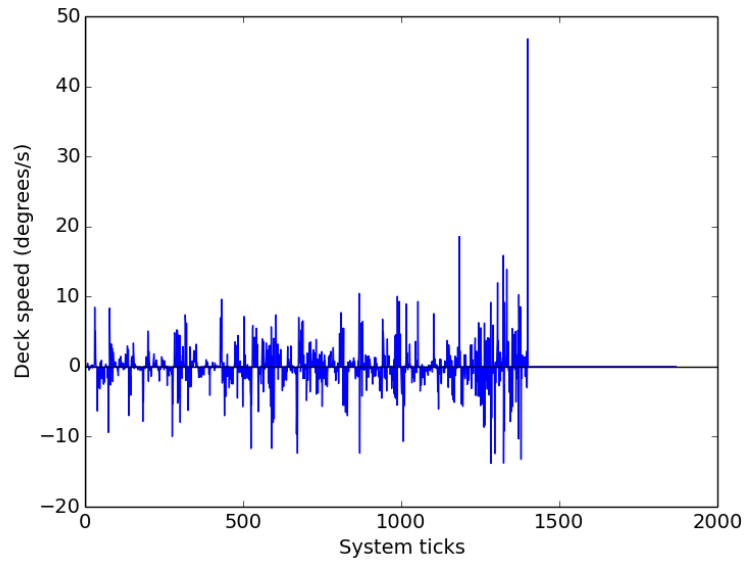


Figure A.24: Difference in deck inclination ($\Delta\phi$) during trial 2 with AOANN PID-28 controller using Do the Shake policy

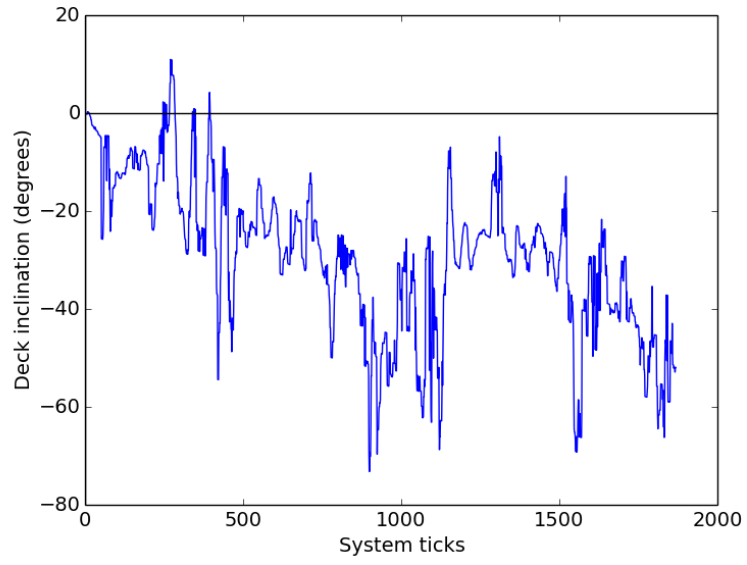


Figure A.25: Deck inclination during trial 3 with AOANN PID-28 controller using Do the Shake policy

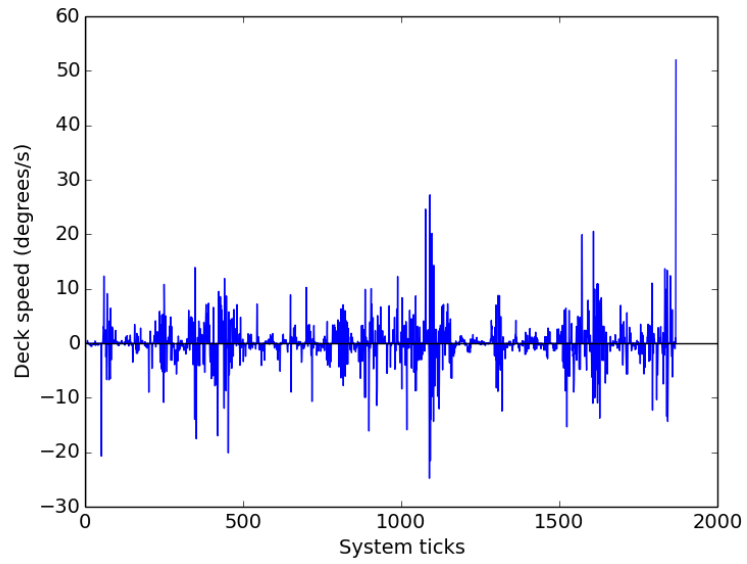


Figure A.26: Difference in deck inclination ($\Delta\phi$) during trial 3 with AOANN PID-28 controller using Do the Shake policy

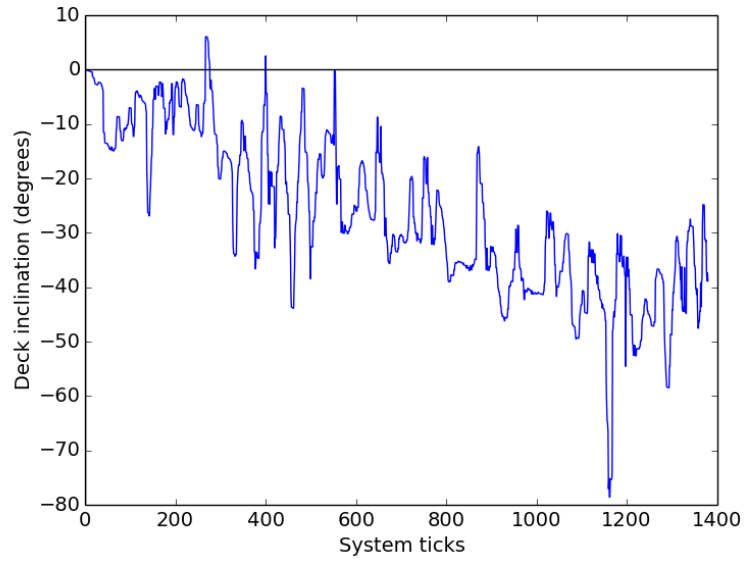


Figure A.27: Deck inclination during trial 4 with AOANN PID-28 controller using Do the Shake policy

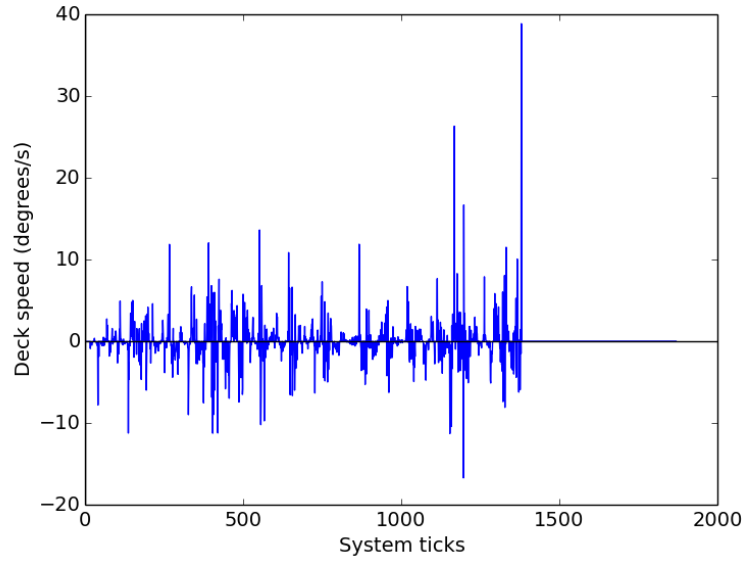


Figure A.28: Difference in deck inclination ($\Delta\phi$) during trial 4 with AOANN PID-28 controller using Do the Shake policy

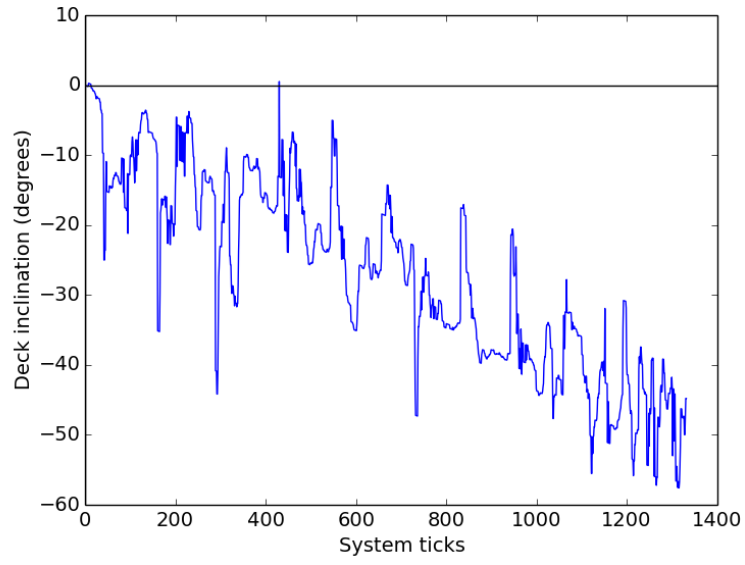


Figure A.29: Deck inclination during trial 5 with AOANN PID-28 controller using Do the Shake policy

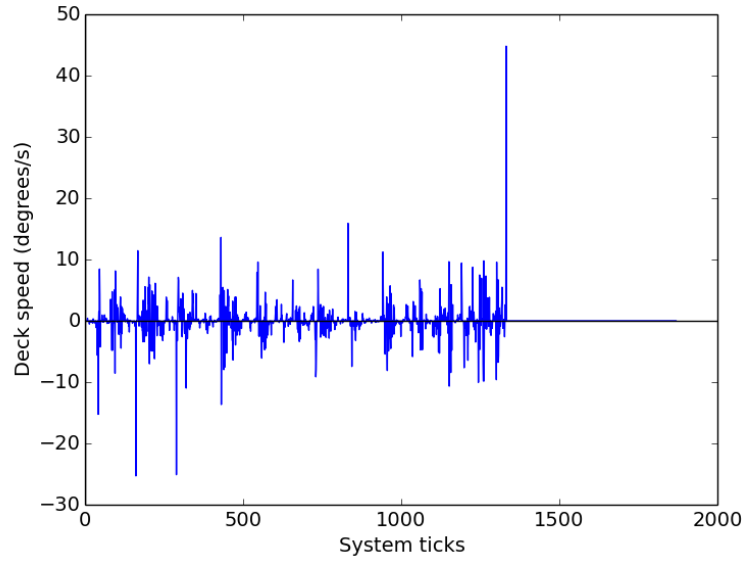


Figure A.30: Difference in deck inclination ($\Delta\phi$) during trial 5 with AOANN PID-28 controller using Do the Shake policy

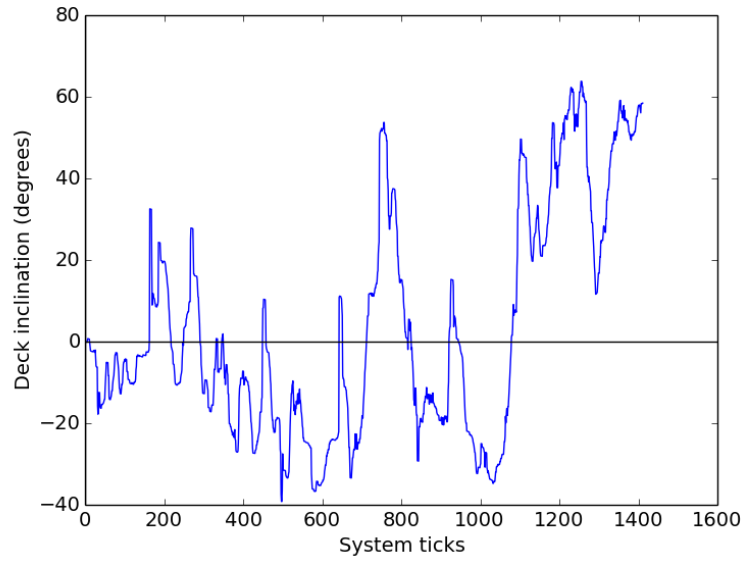


Figure A.31: Deck inclination during trial 1 with AOANN Fuz-28 controller using Do the Shake policy

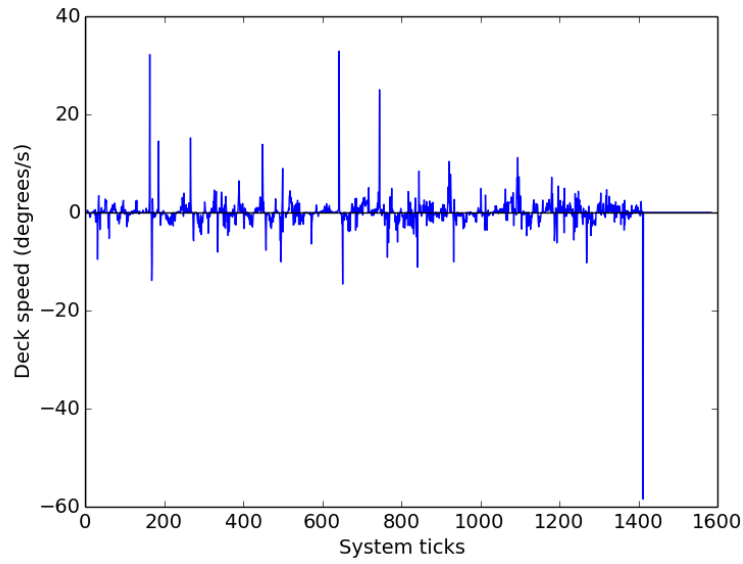


Figure A.32: Difference in deck inclination ($\Delta\phi$) during trial 1 with AOANN Fuz-28 controller using Do the Shake policy

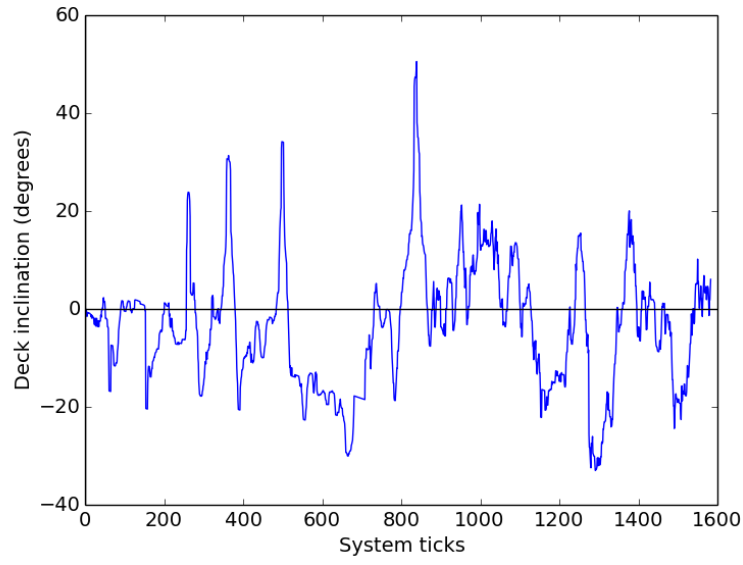


Figure A.33: Deck inclination during trial 2 with AOANN Fuz-28 controller using Do the Shake policy

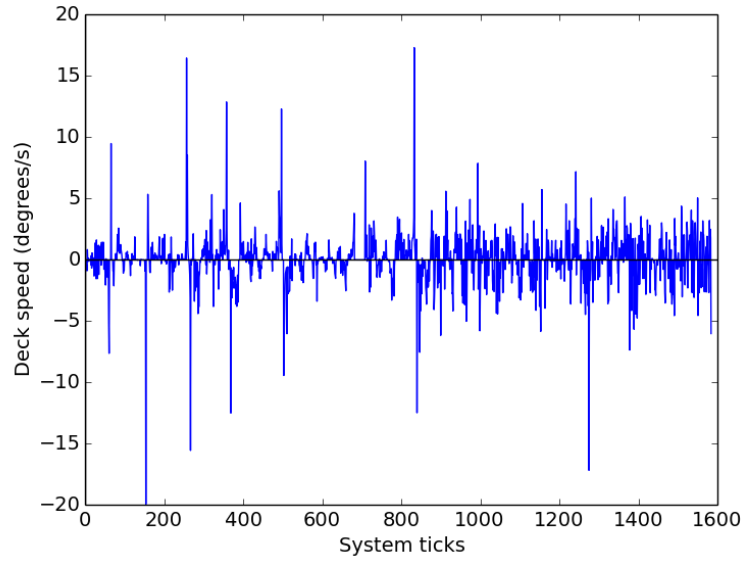


Figure A.34: Difference in deck inclination ($\Delta\phi$) during trial 2 with AOANN Fuz-28 controller using Do the Shake policy

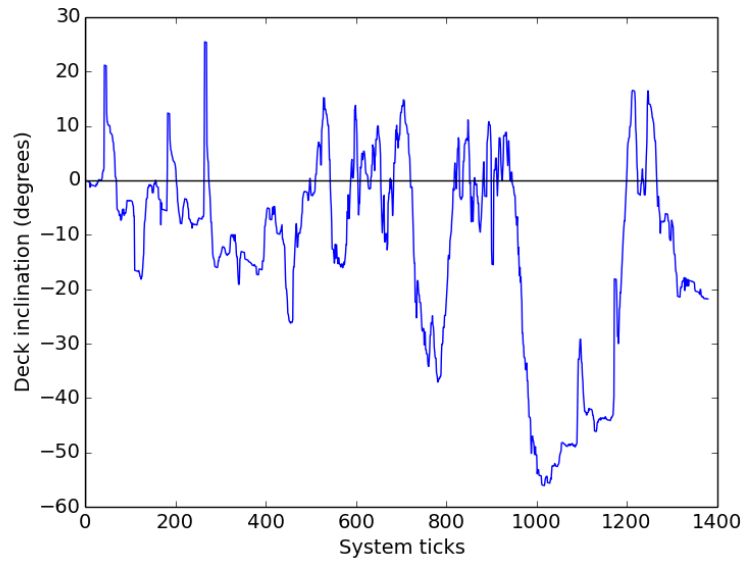


Figure A.35: Deck inclination during trial 3 with AOANN Fuz-28 controller using Do the Shake policy

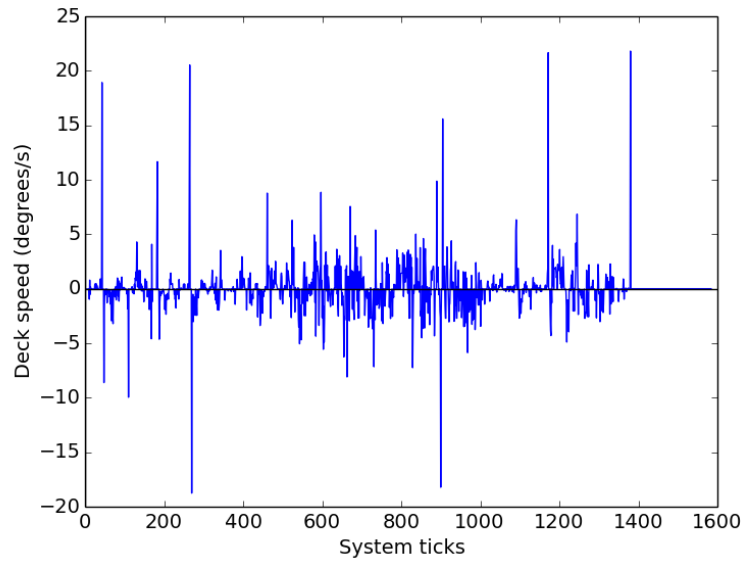


Figure A.36: Difference in deck inclination ($\Delta\phi$) during trial 3 with AOANN Fuz-28 controller using Do the Shake policy

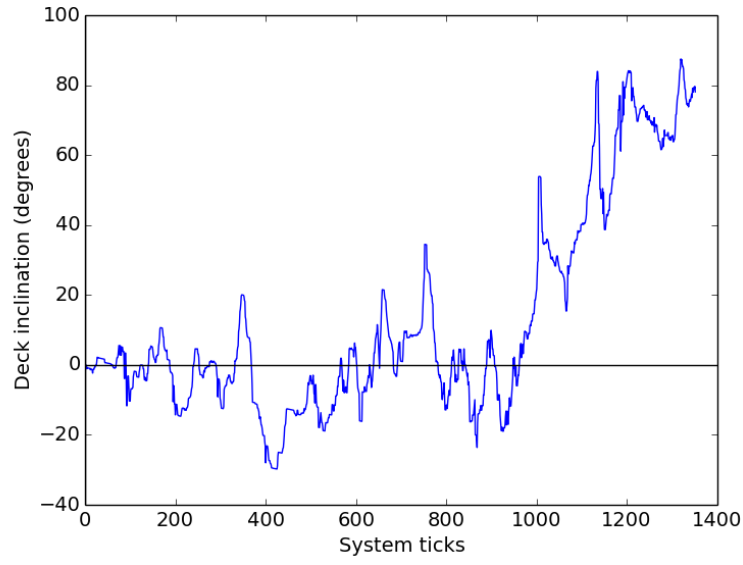


Figure A.37: Deck inclination during trial 4 with AOANN Fuz-28 controller using Do the Shake policy

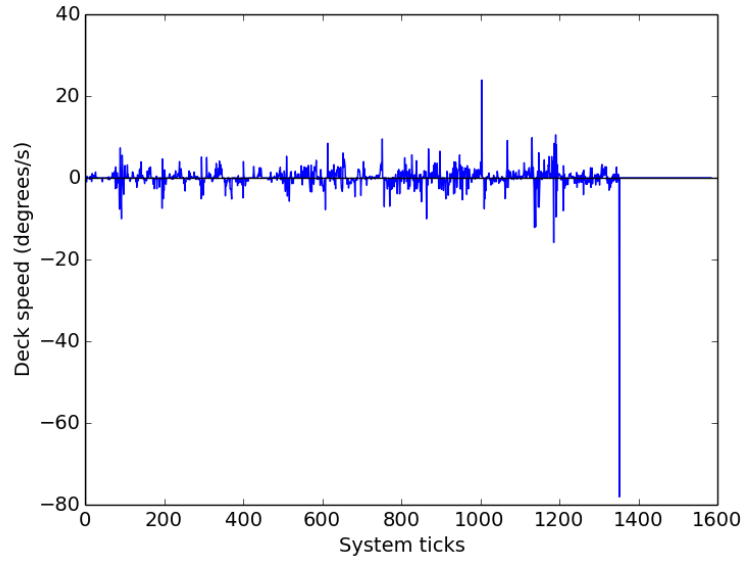


Figure A.38: Difference in deck inclination ($\Delta\phi$) during trial 4 with AOANN Fuz-28 controller using Do the Shake policy

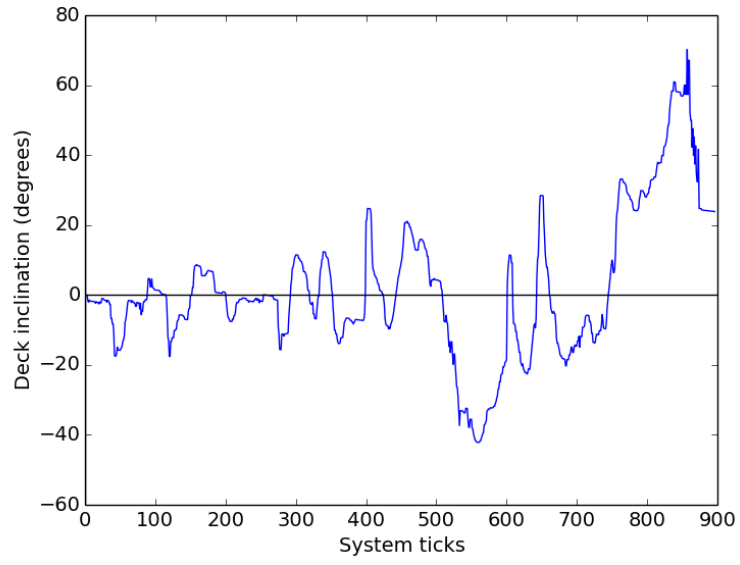


Figure A.39: Deck inclination during trial 5 with AOANN Fuz-28 controller using Do the Shake policy

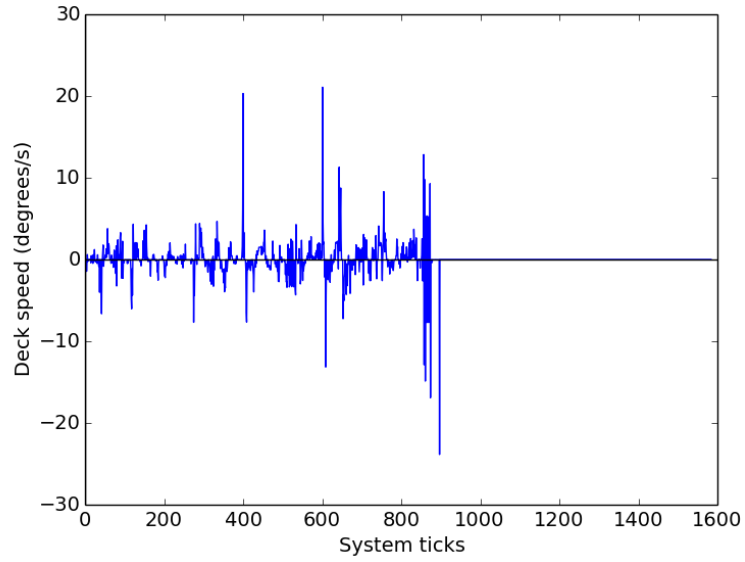


Figure A.40: Difference in deck inclination ($\Delta\phi$) during trial 5 with AOANN Fuz-28 controller using Do the Shake policy

A.2 Let's Sway Policy

This section shows the deck inclination and $\Delta\phi$ during all trials using the Let's Sway policy.

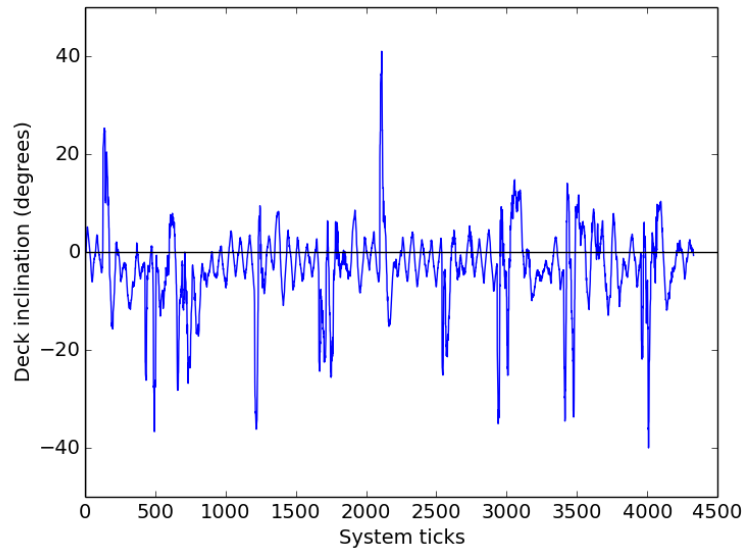


Figure A.41: Deck inclination during trial 1 with PID controller using Let's Sway policy

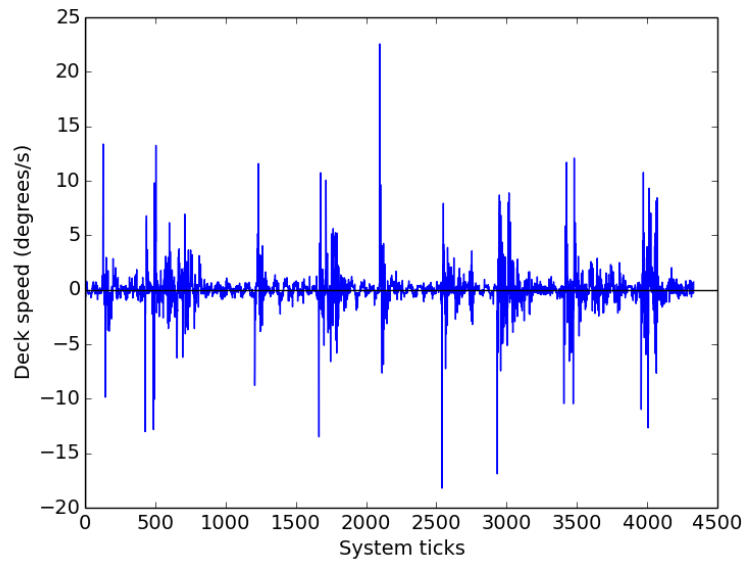


Figure A.42: Difference in deck inclination ($\Delta\phi$) during trial 1 with PID controller using Let's Sway policy

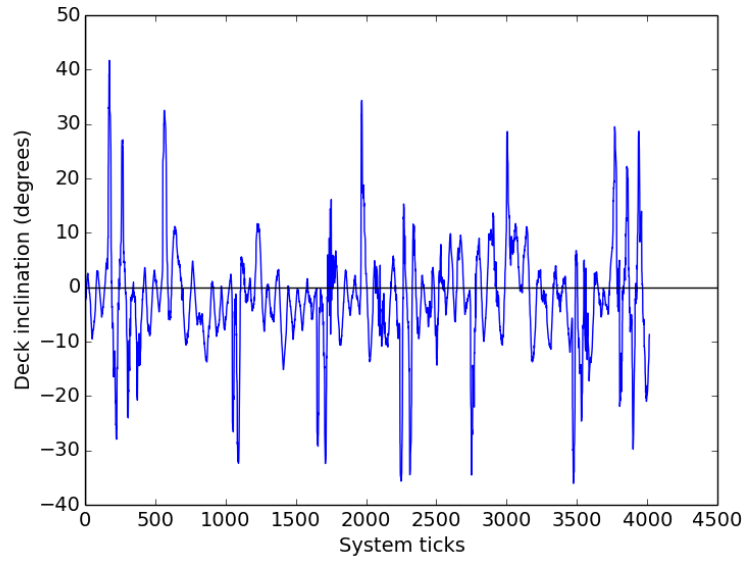


Figure A.43: Deck inclination during trial 2 with PID controller using Let's Sway policy

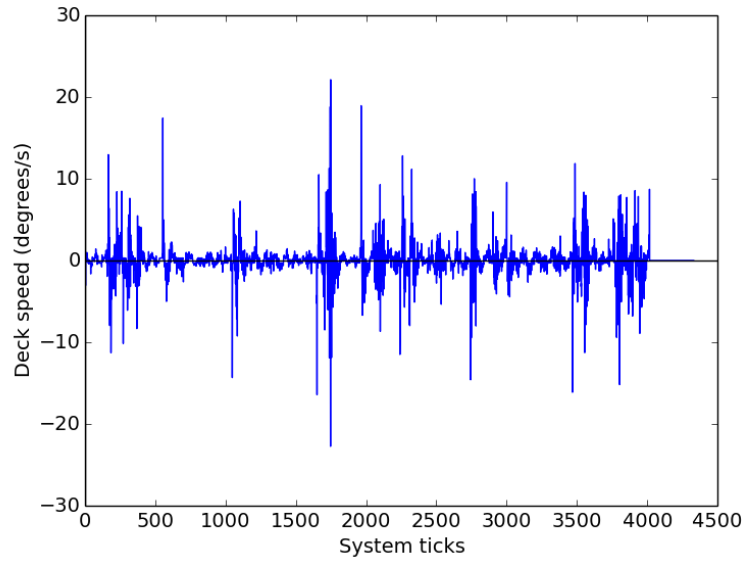


Figure A.44: Difference in deck inclination ($\Delta\phi$) during trial 2 with PID controller using Let's Sway policy

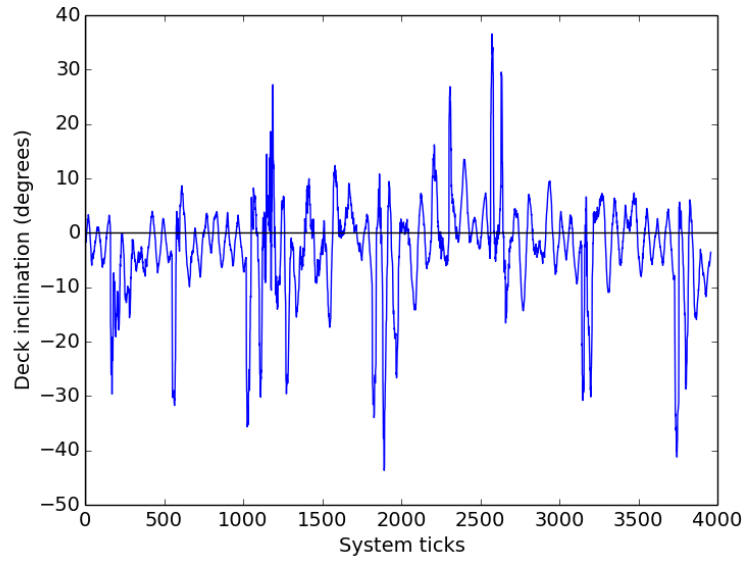


Figure A.45: Deck inclination during trial 3 with PID controller using Let's Sway policy

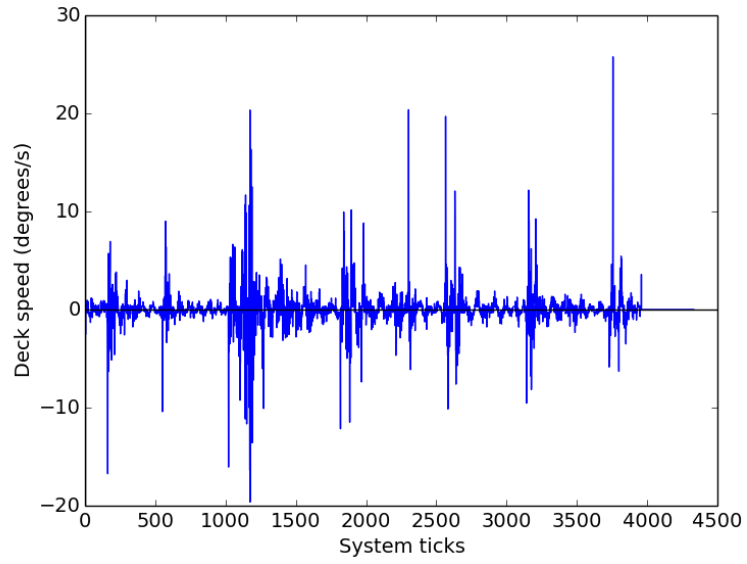


Figure A.46: Difference in deck inclination ($\Delta\phi$) during trial 3 with PID controller using Let's Sway policy

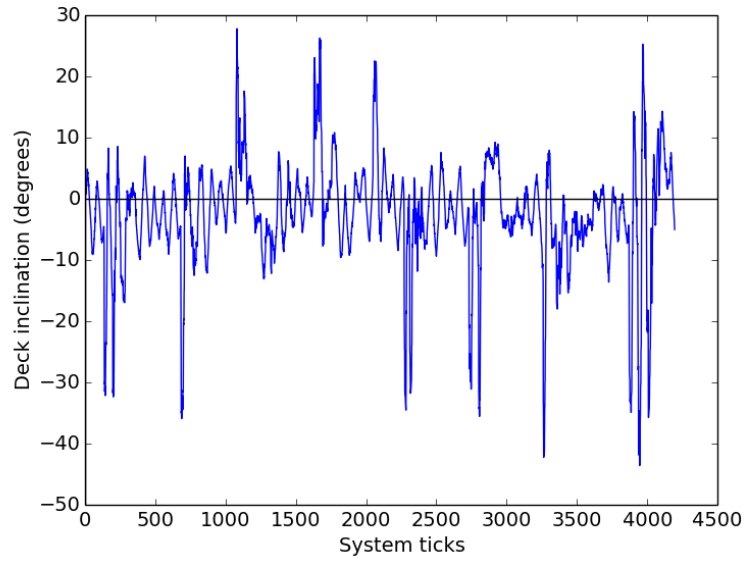


Figure A.47: Deck inclination during trial 4 with PID controller using Let's Sway policy

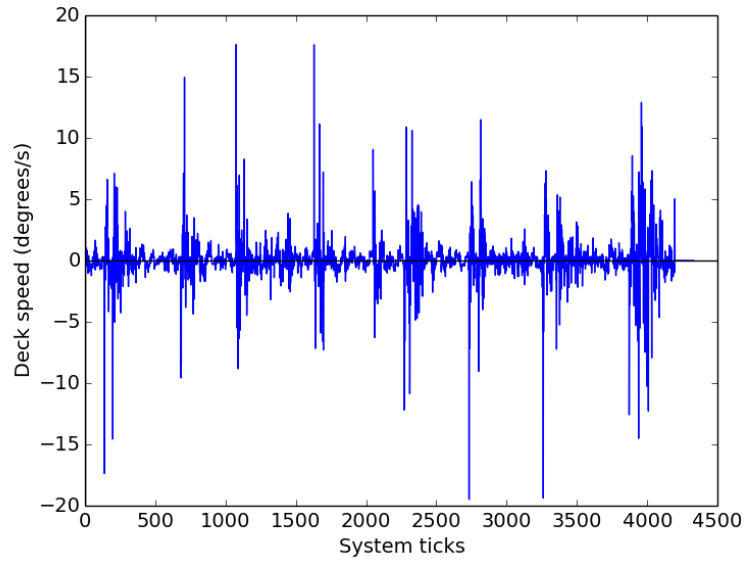


Figure A.48: Difference in deck inclination ($\Delta\phi$) during trial 4 with PID controller using Let's Sway policy

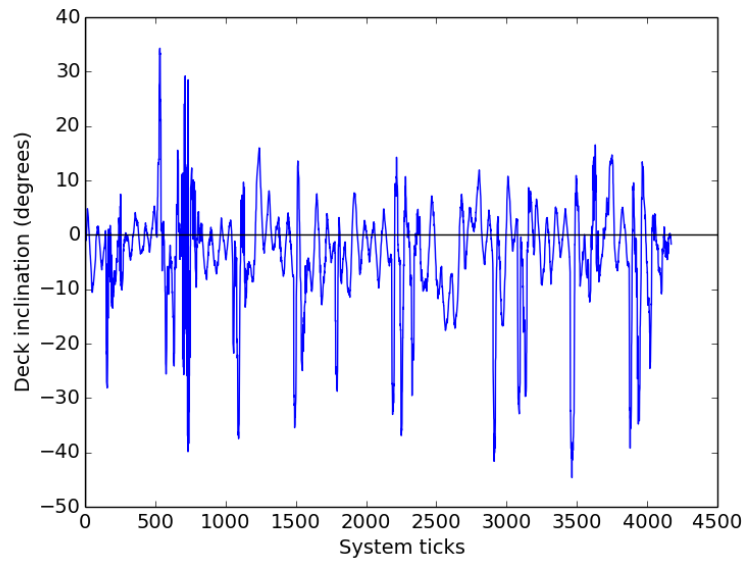


Figure A.49: Deck inclination during trial 5 with PID controller using Let's Sway policy

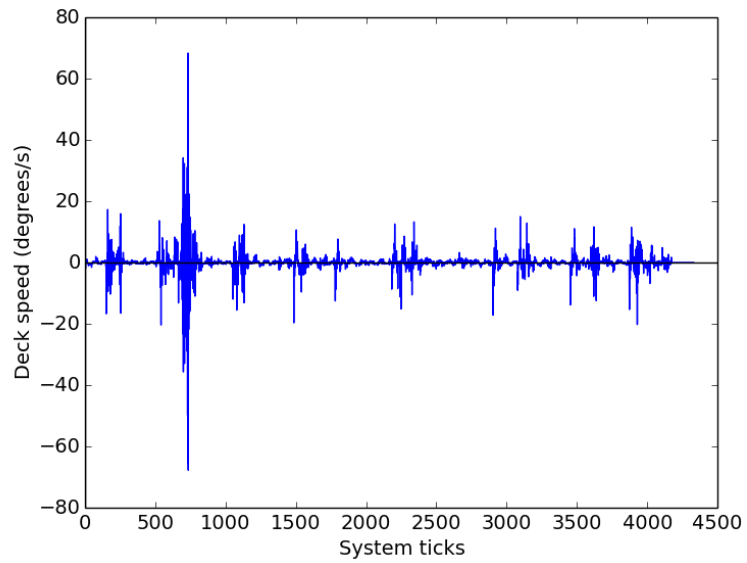


Figure A.50: Difference in deck inclination ($\Delta\phi$) during trial 5 with PID controller using Let's Sway policy

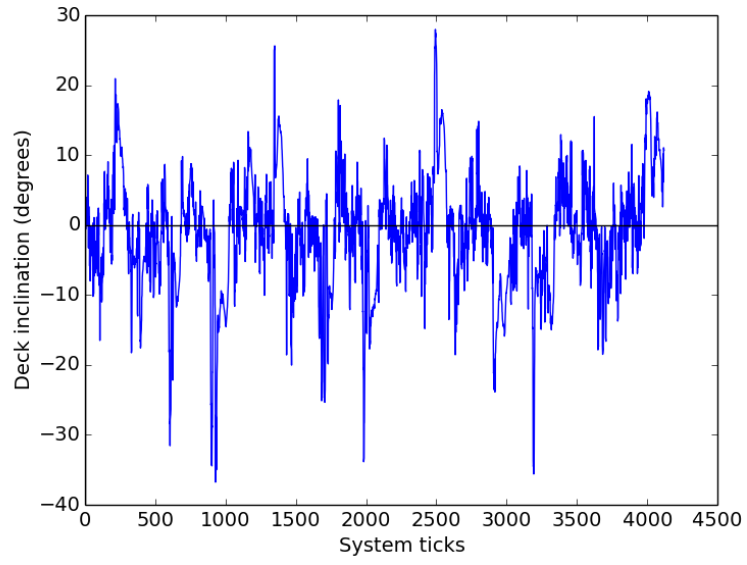


Figure A.51: Deck inclination during trial 1 with Fuzzy Logic controller using Let's Sway policy

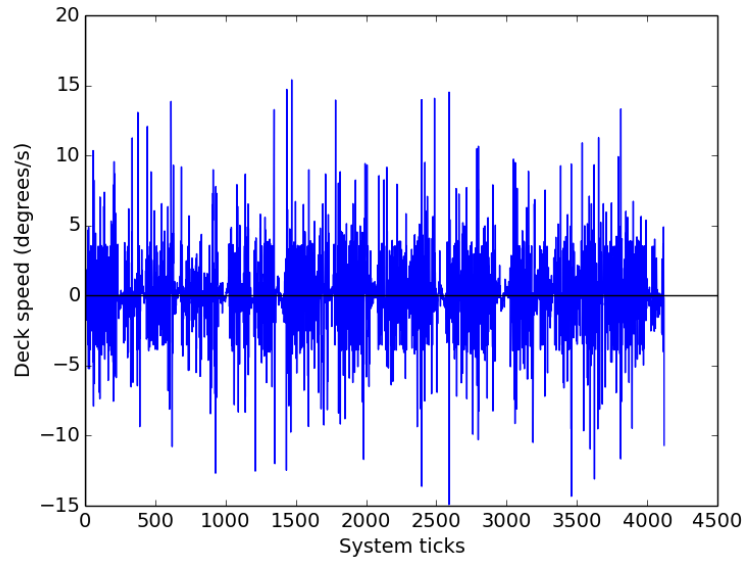


Figure A.52: Difference in deck inclination ($\Delta\phi$) during trial 1 with Fuzzy Logic controller using Let's Sway policy

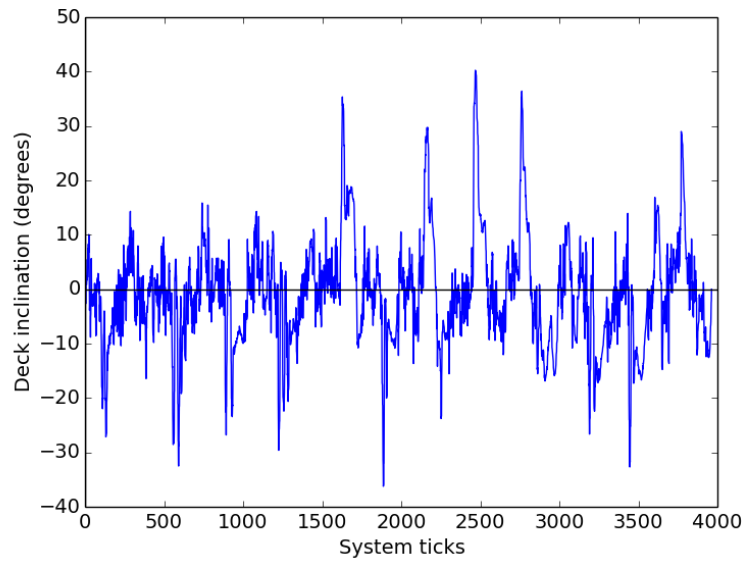


Figure A.53: Deck inclination during trial 2 with Fuzzy Logic controller using Let's Sway policy

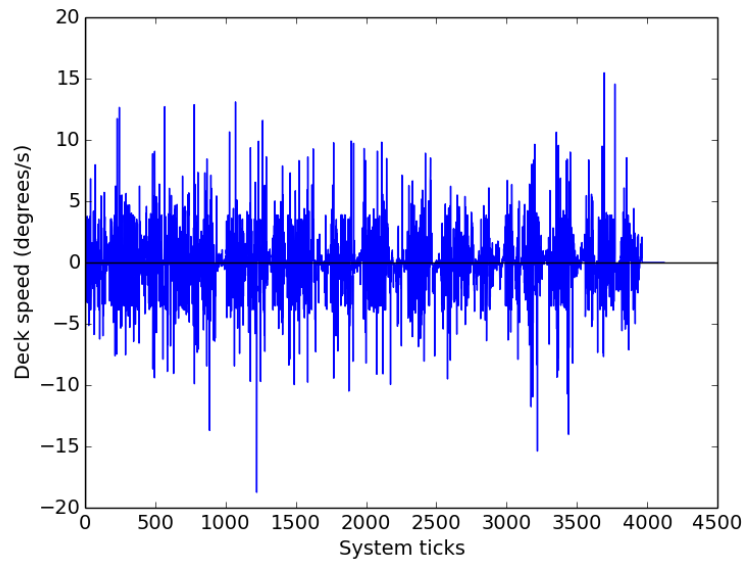


Figure A.54: Difference in deck inclination ($\Delta\phi$) during trial 2 with Fuzzy Logic controller using Let's Sway policy

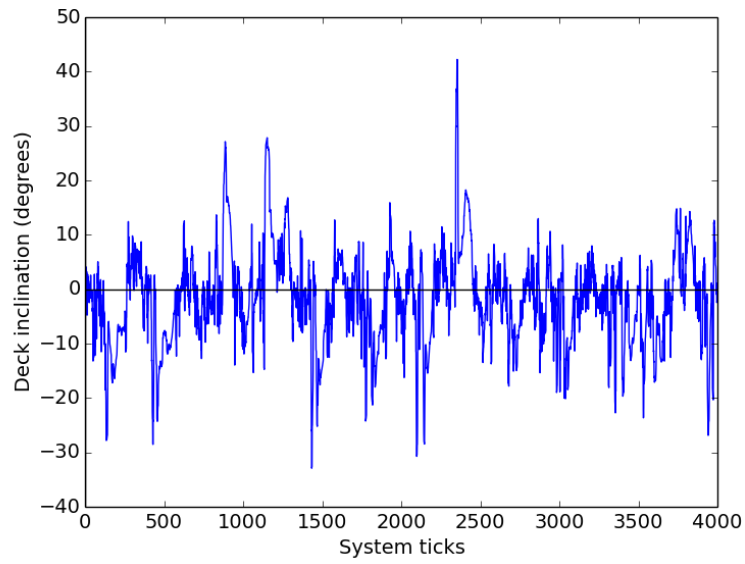


Figure A.55: Deck inclination during trial 3 with Fuzzy Logic controller using Let's Sway policy

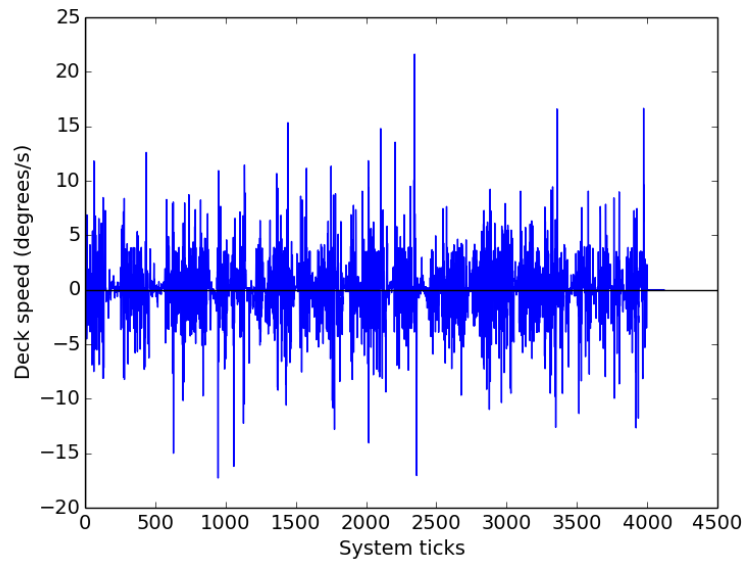


Figure A.56: Difference in deck inclination ($\Delta\phi$) during trial 3 with Fuzzy Logic controller using Let's Sway policy

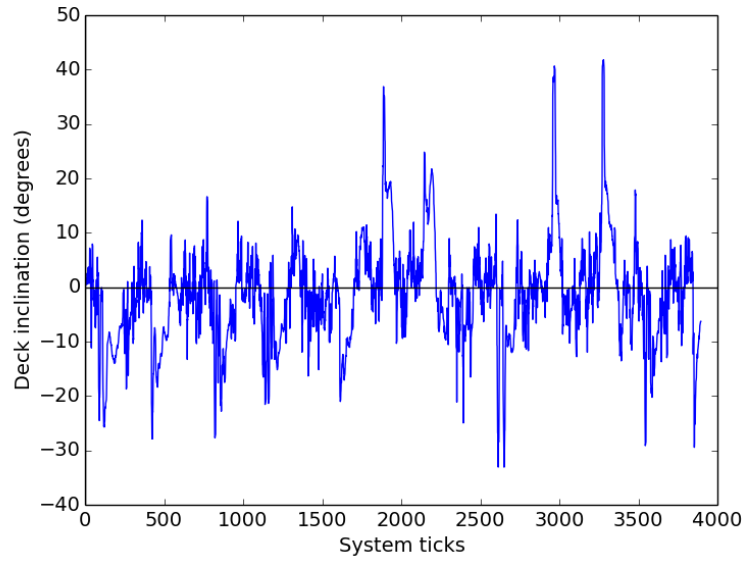


Figure A.57: Deck inclination during trial 4 with Fuzzy Logic controller using Let's Sway policy

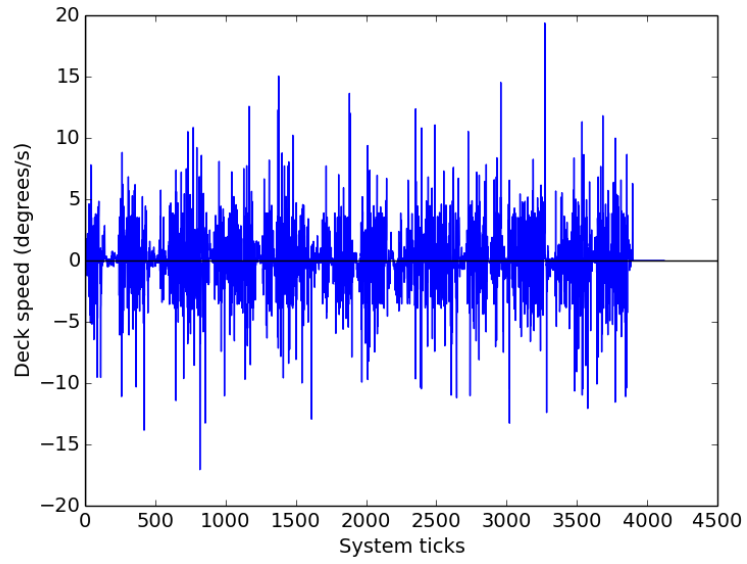


Figure A.58: Difference in deck inclination ($\Delta\phi$) during trial 4 with Fuzzy Logic controller using Let's Sway policy

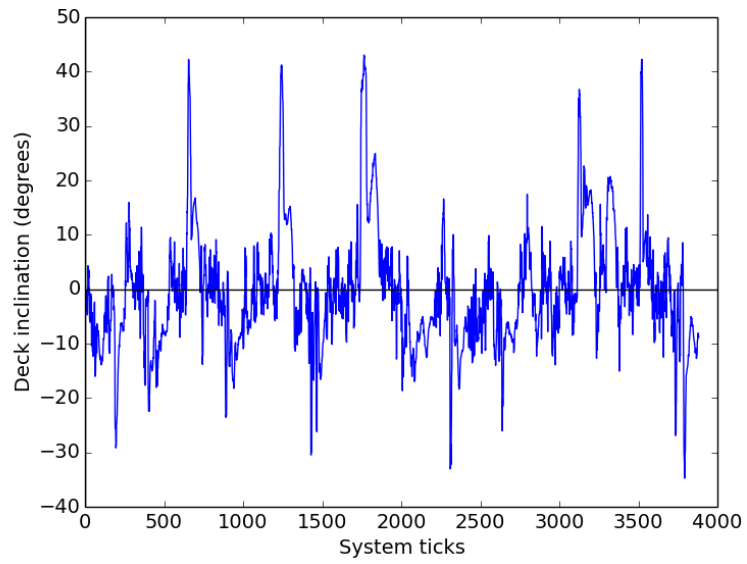


Figure A.59: Deck inclination during trial 5 with Fuzzy Logic controller using Let's Sway policy

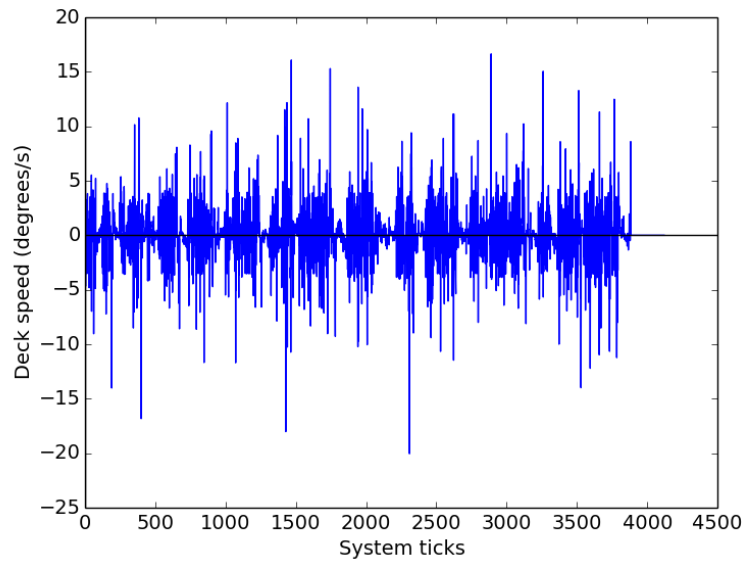


Figure A.60: Difference in deck inclination ($\Delta\phi$) during trial 5 with Fuzzy Logic controller using Let's Sway policy

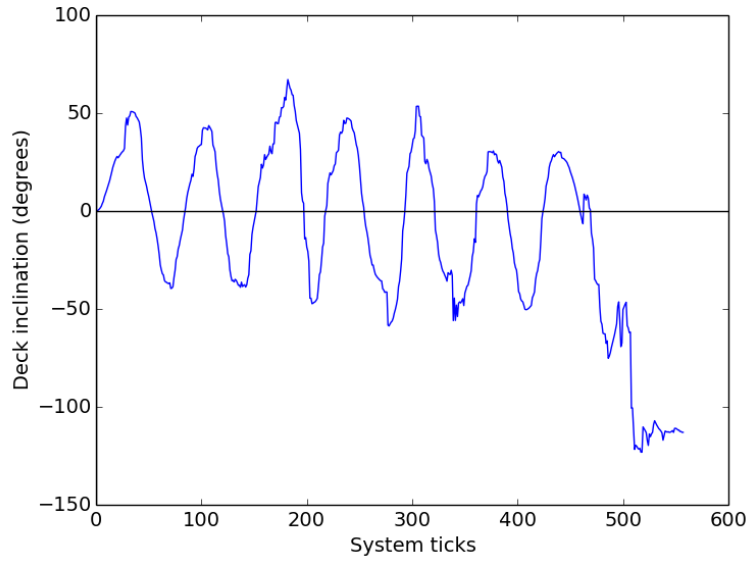


Figure A.61: Deck inclination during trial 1 with AOANN PID-28 controller using Let's Sway policy

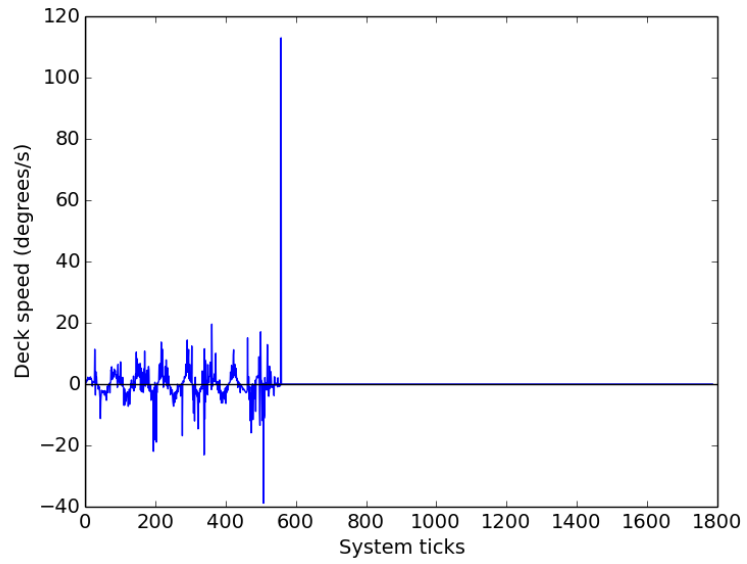


Figure A.62: Difference in deck inclination ($\Delta\phi$) during trial 1 with AOANN PID-28 controller using Let's Sway policy

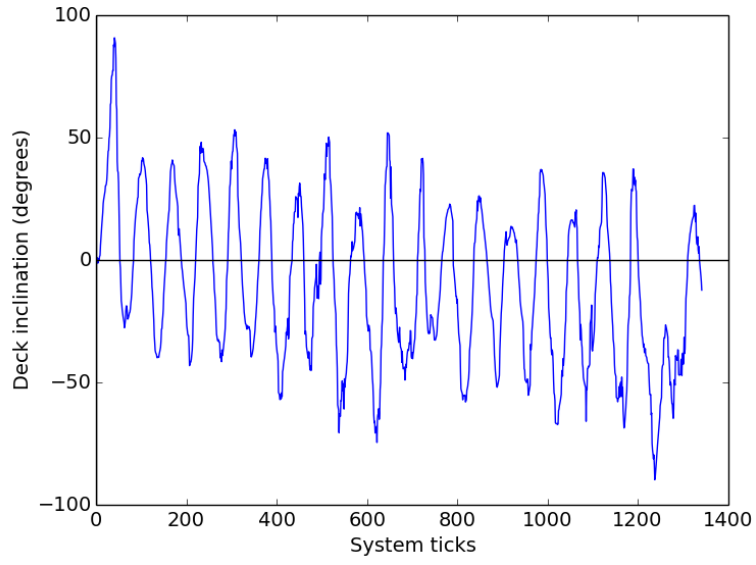


Figure A.63: Deck inclination during trial 2 with AOANN PID-28 controller using Let's Sway policy

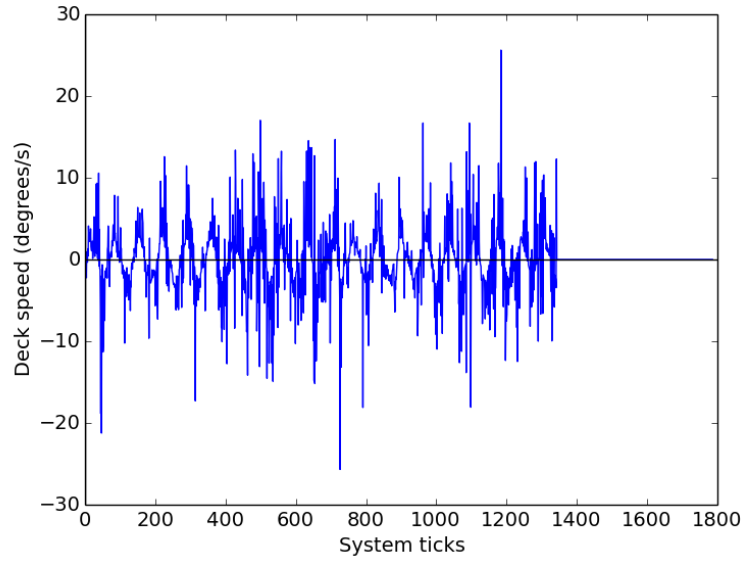


Figure A.64: Difference in deck inclination ($\Delta\phi$) during trial 2 with AOANN PID-28 controller using Let's Sway policy

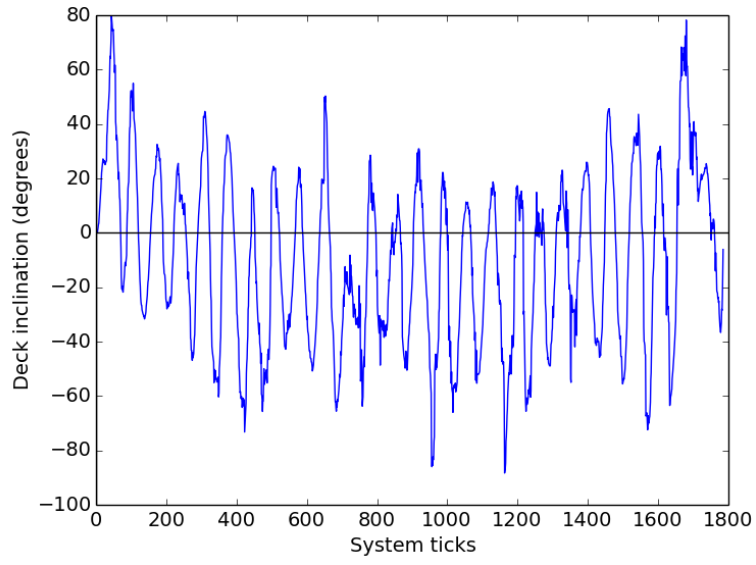


Figure A.65: Deck inclination during trial 3 with AOANN PID-28 controller using Let's Sway policy

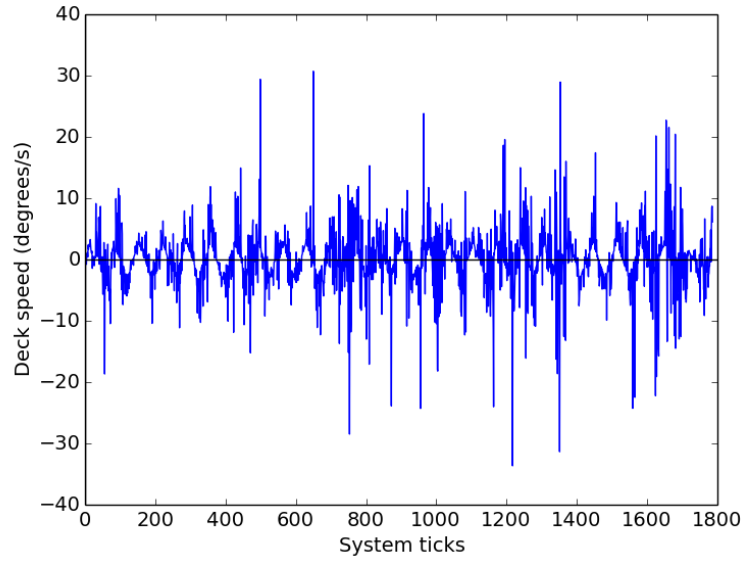


Figure A.66: Difference in deck inclination ($\Delta\phi$) during trial 3 with AOANN PID-28 controller using Let's Sway policy

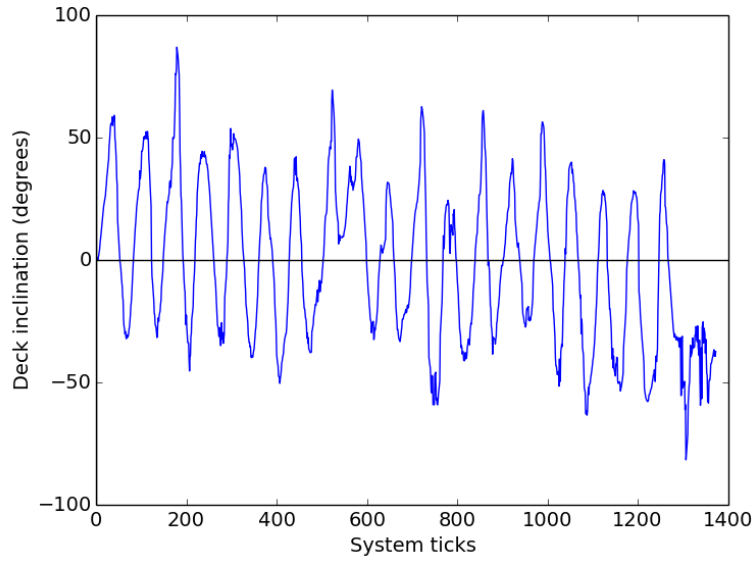


Figure A.67: Deck inclination during trial 4 with AOANN PID-28 controller using Let's Sway policy

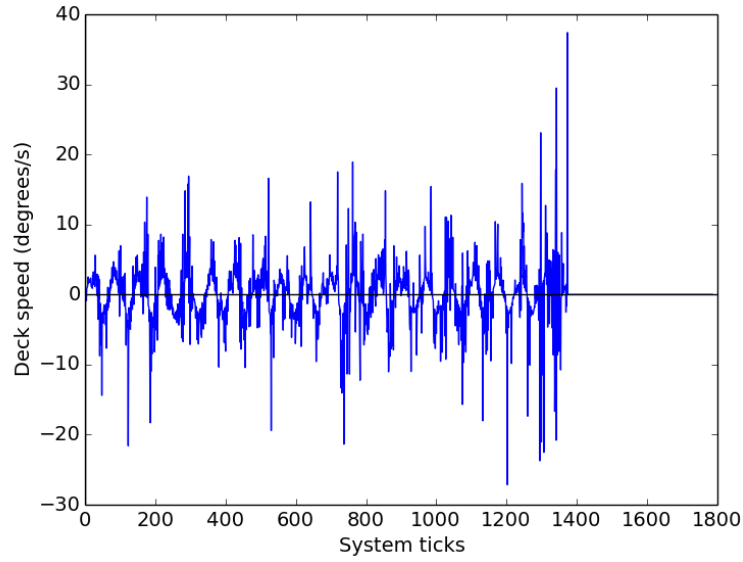


Figure A.68: Difference in deck inclination ($\Delta\phi$) during trial 4 with AOANN PID-28 controller using Let's Sway policy

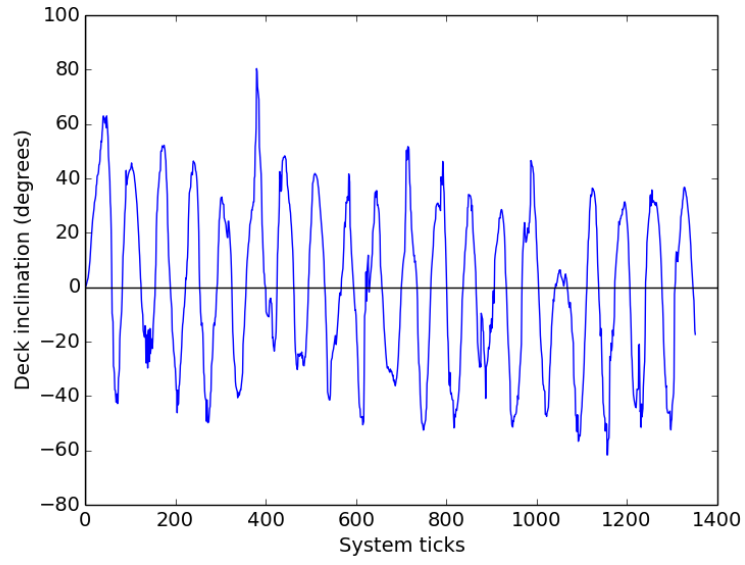


Figure A.69: Deck inclination during trial 5 with AOANN PID-28 controller using Let's Sway policy

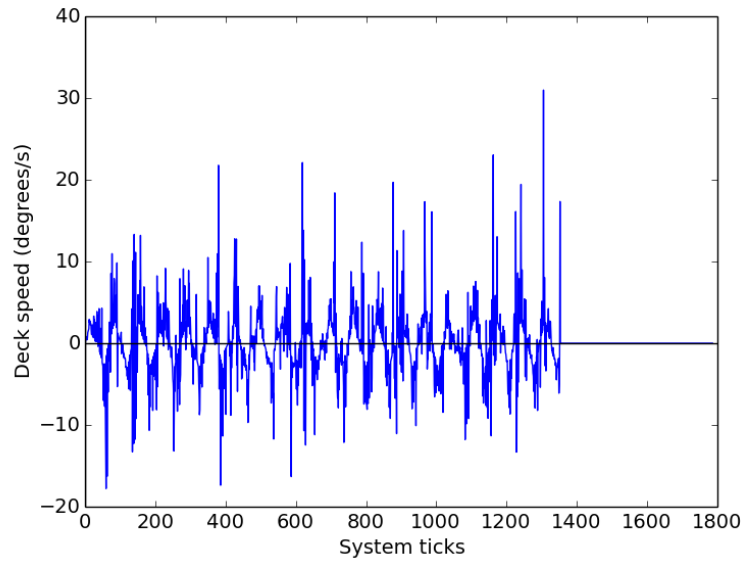


Figure A.70: Difference in deck inclination ($\Delta\phi$) during trial 5 with AOANN PID-28 controller using Let's Sway policy

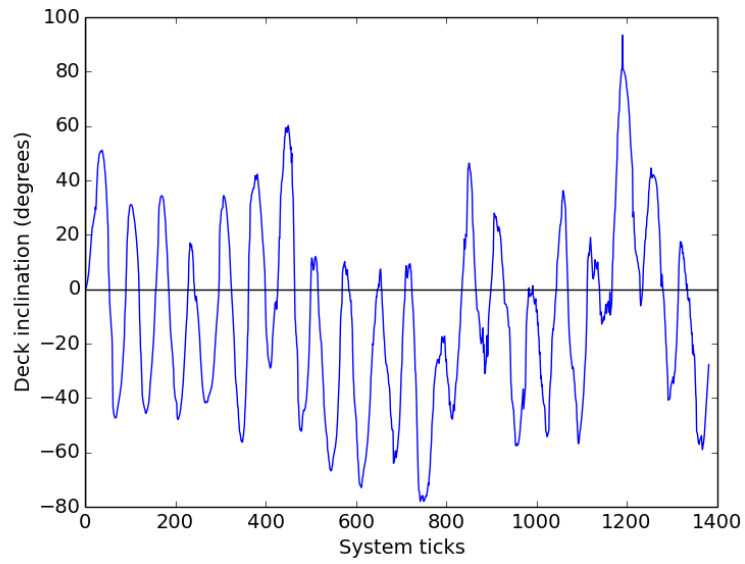


Figure A.71: Deck inclination during trial 1 with AOANN Fuz-28 controller using Let's Sway policy

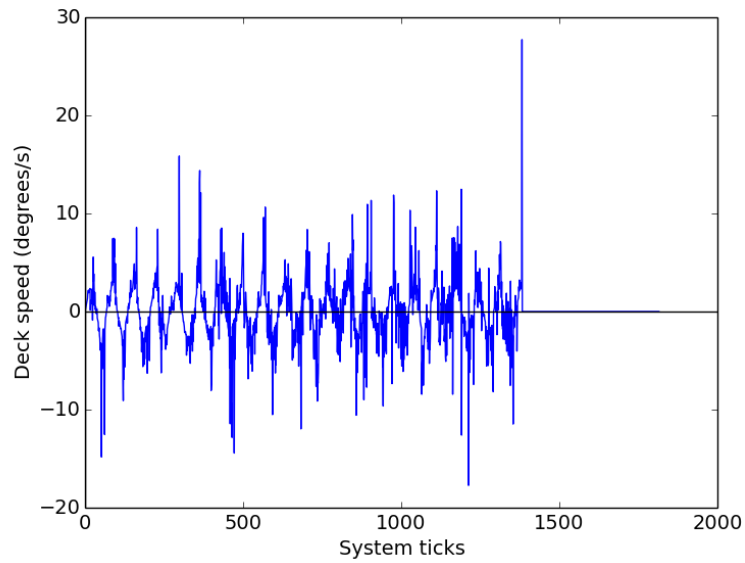


Figure A.72: Difference in deck inclination ($\Delta\phi$) during trial 1 with AOANN Fuz-28 controller using Let's Sway policy

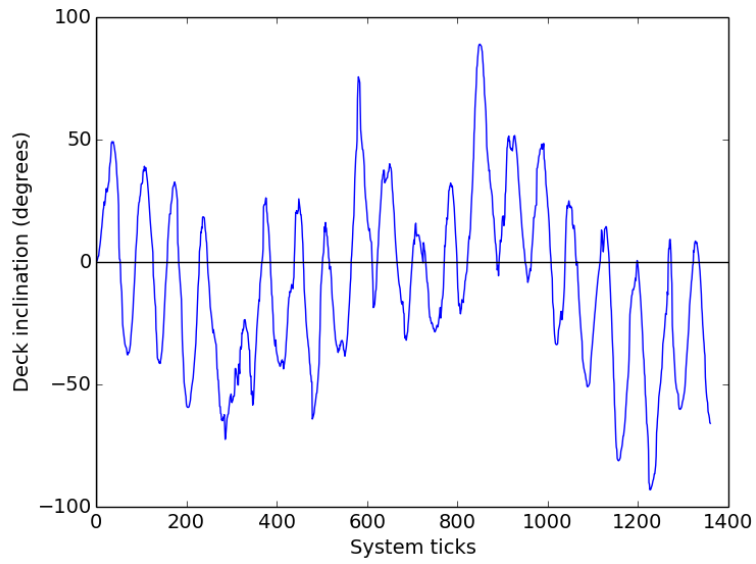


Figure A.73: Deck inclination during trial 2 with AOANN Fuz-28 controller using Let's Sway policy

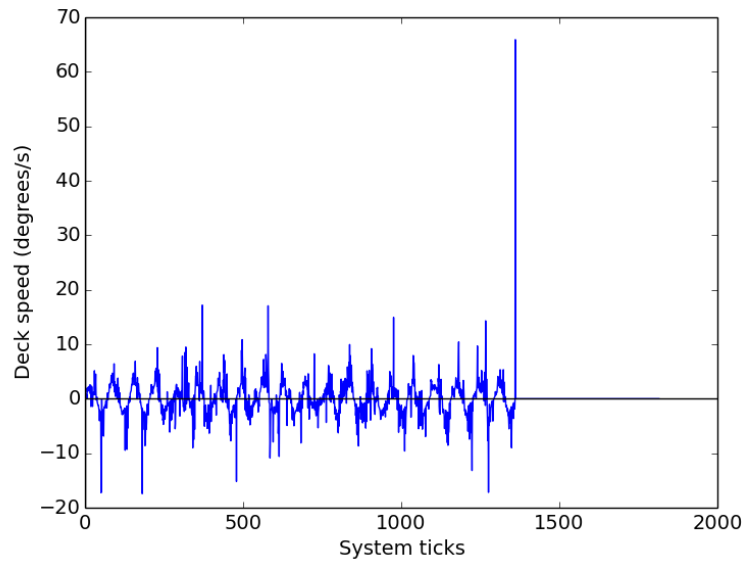


Figure A.74: Difference in deck inclination ($\Delta\phi$) during trial 2 with AOANN Fuz-28 controller using Let's Sway policy

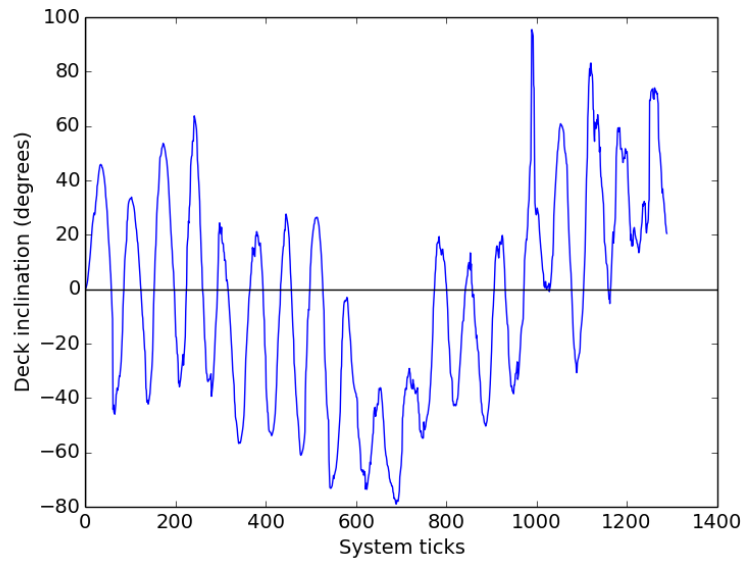


Figure A.75: Deck inclination during trial 3 with AOANN Fuz-28 controller using Let's Sway policy

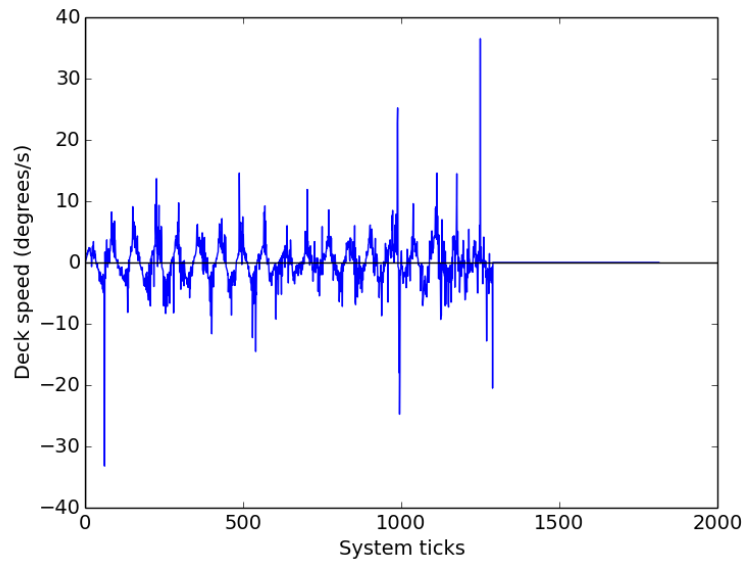


Figure A.76: Difference in deck inclination ($\Delta\phi$) during trial 3 with AOANN Fuz-28 controller using Let's Sway policy

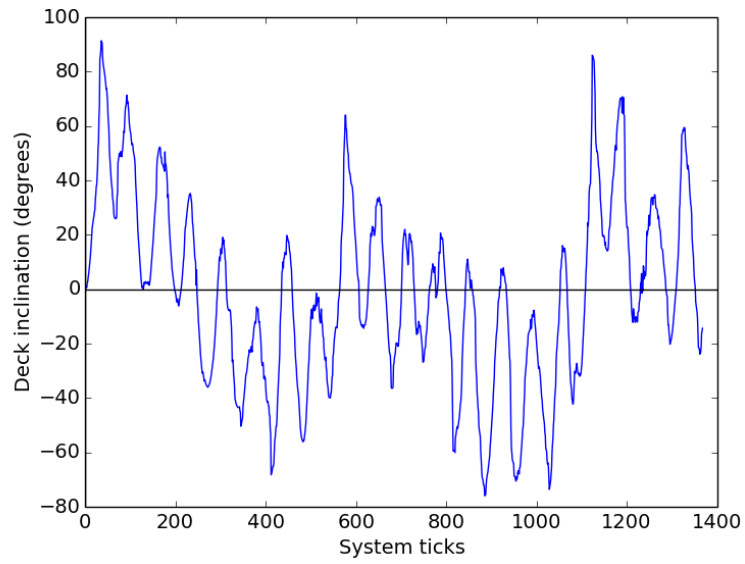


Figure A.77: Deck inclination during trial 4 with AOANN Fuz-28 controller using Let's Sway policy

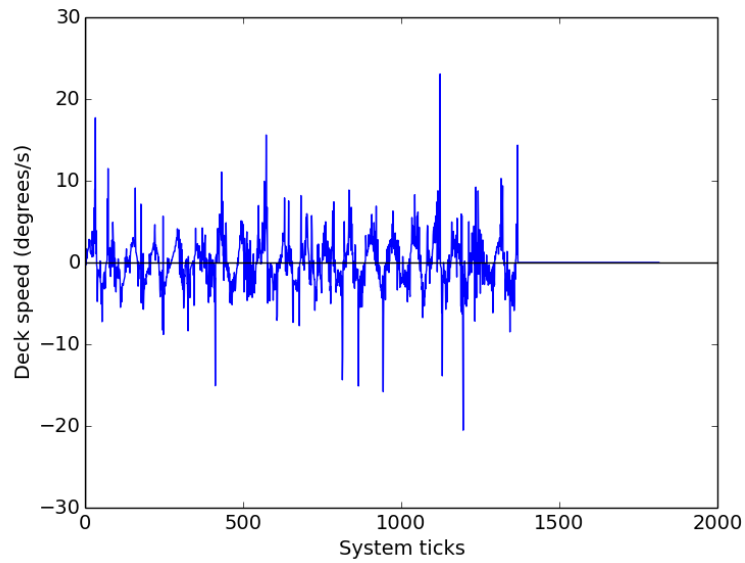


Figure A.78: Difference in deck inclination ($\Delta\phi$) during trial 4 with AOANN Fuz-28 controller using Let's Sway policy

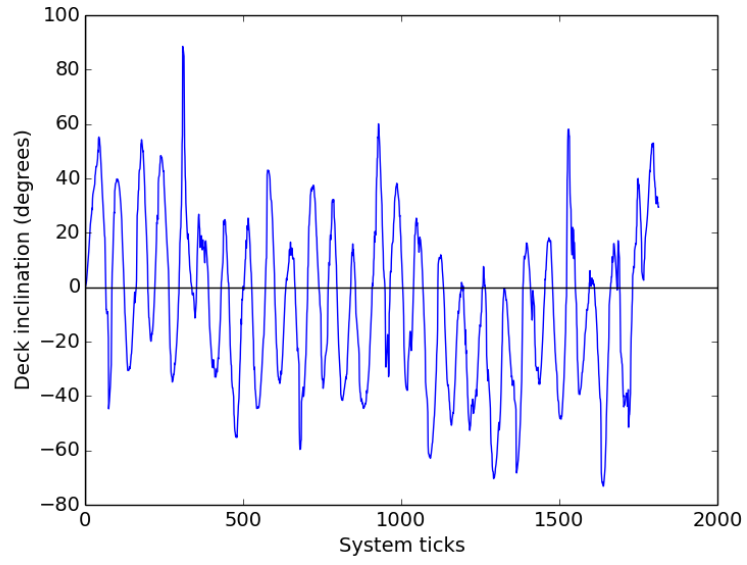


Figure A.79: Deck inclination during trial 5 with AOANN Fuz-28 controller using Let's Sway policy

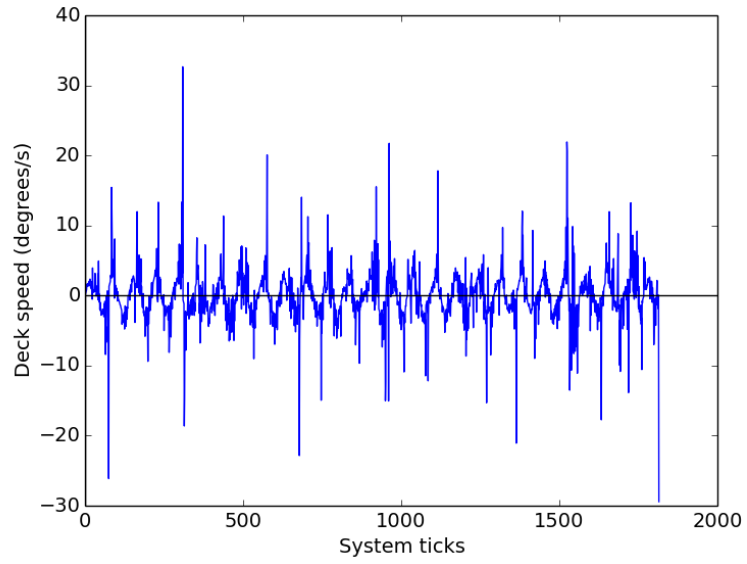


Figure A.80: Difference in deck inclination ($\Delta\phi$) during trial 5 with AOANN Fuz-28 controller using Let's Sway policy

B Sensor Data From All Trials

The following graphs show the torso inclination and angular velocity recorded during each trial of the experiment. These graphs are stacked in Figures 5.6 to 5.9.

B.1 Do the Shake Policy

This section shows the sensor data recorded when using the Do the Shake policy.

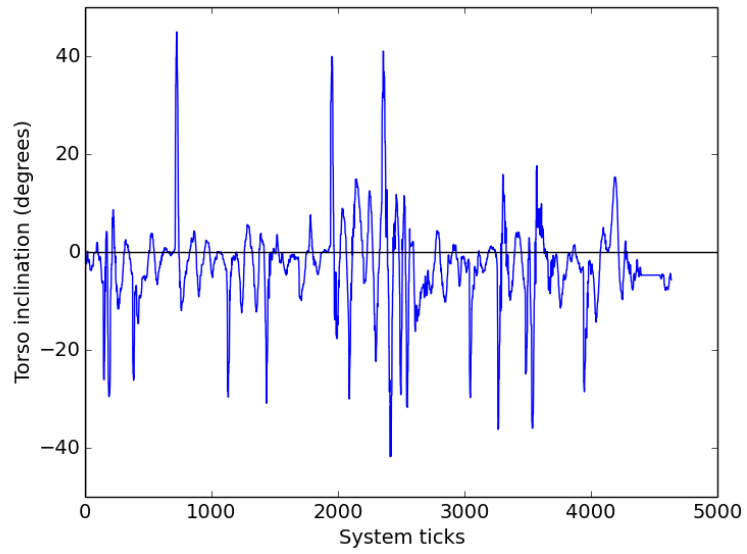


Figure B.1: Torso inclination during trial 1 with PID controller using Do the Shake policy

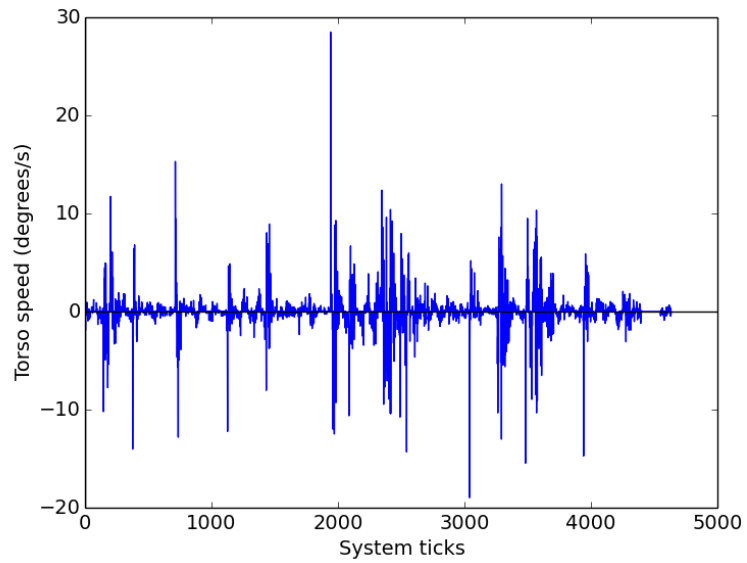


Figure B.2: Torso angular velocity during trial 1 with PID controller using Do the Shake policy

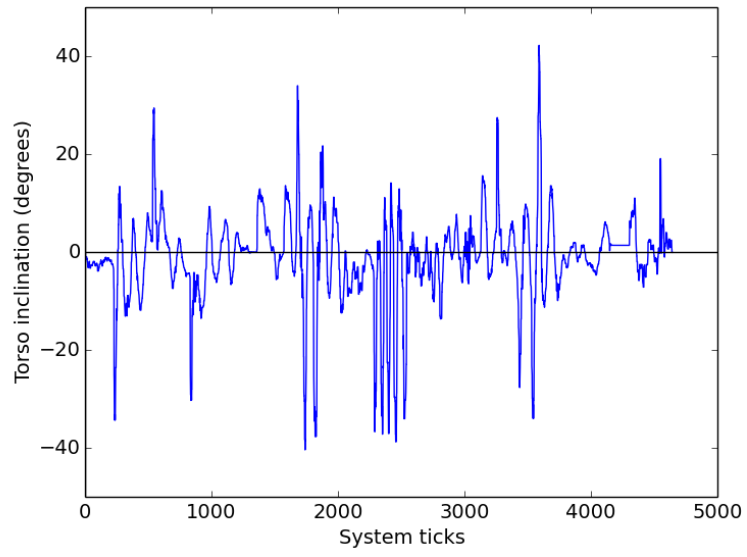


Figure B.3: Torso inclination during trial 2 with PID controller using Do the Shake policy

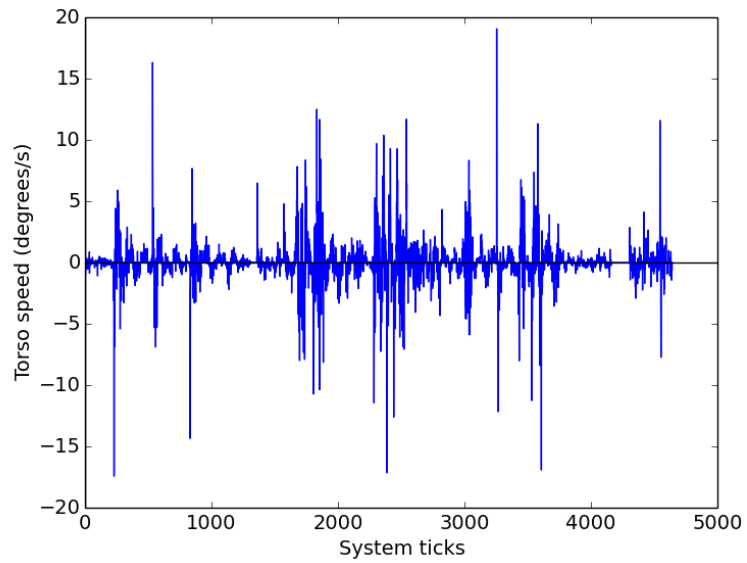


Figure B.4: Torso angular velocity during trial 2 with PID controller using Do the Shake policy

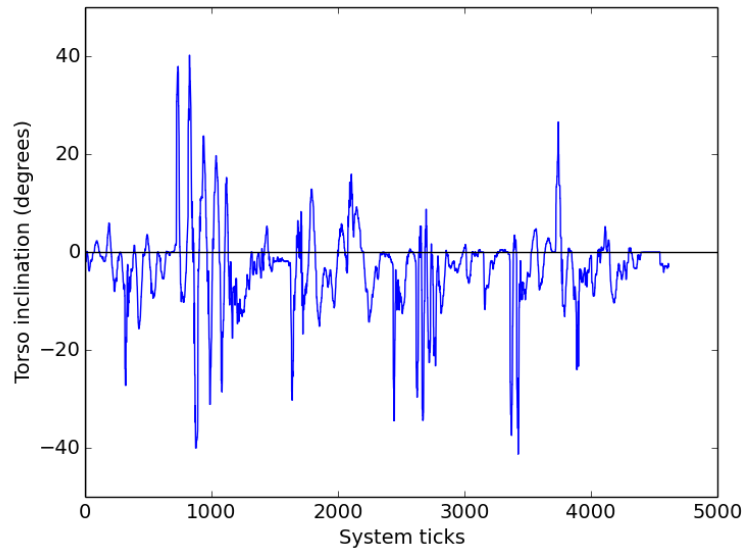


Figure B.5: Torso inclination during trial 3 with PID controller using Do the Shake policy

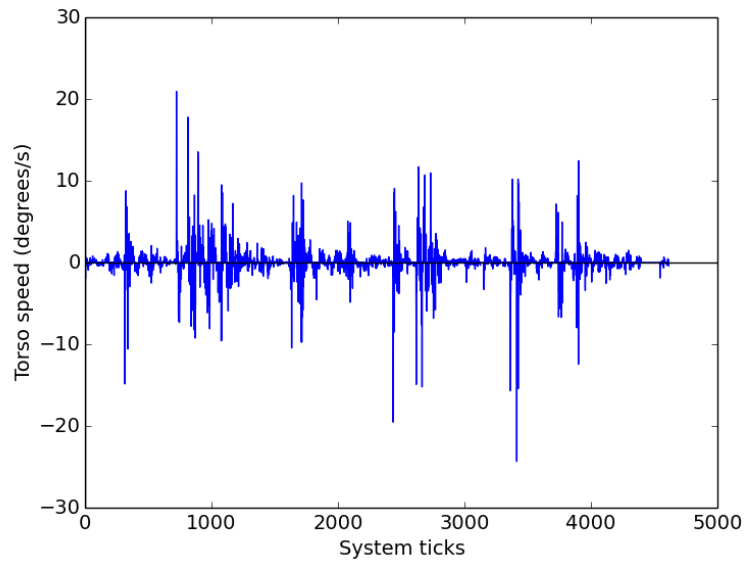


Figure B.6: Torso angular velocity during trial 3 with PID controller using Do the Shake policy

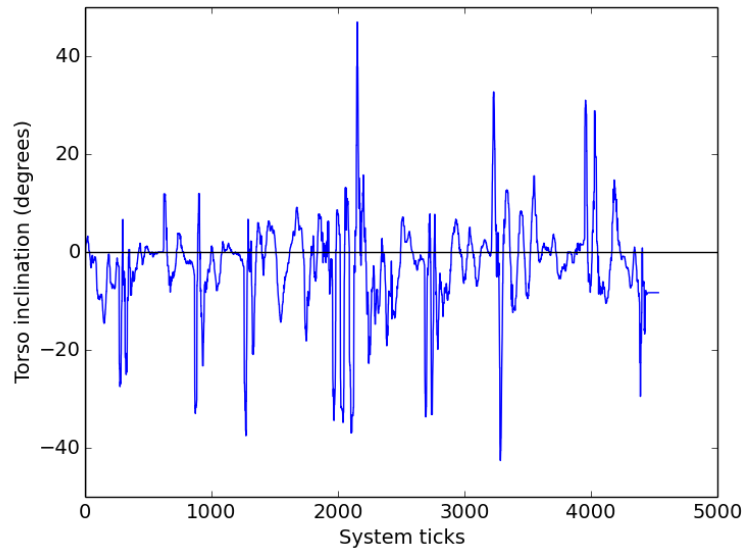


Figure B.7: Torso inclination during trial 4 with PID controller using Do the Shake policy

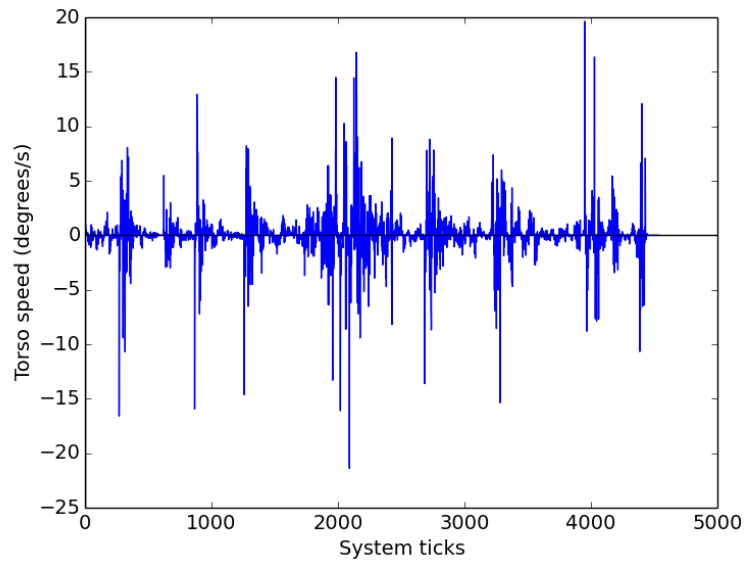


Figure B.8: Torso angular velocity during trial 4 with PID controller using Do the Shake policy

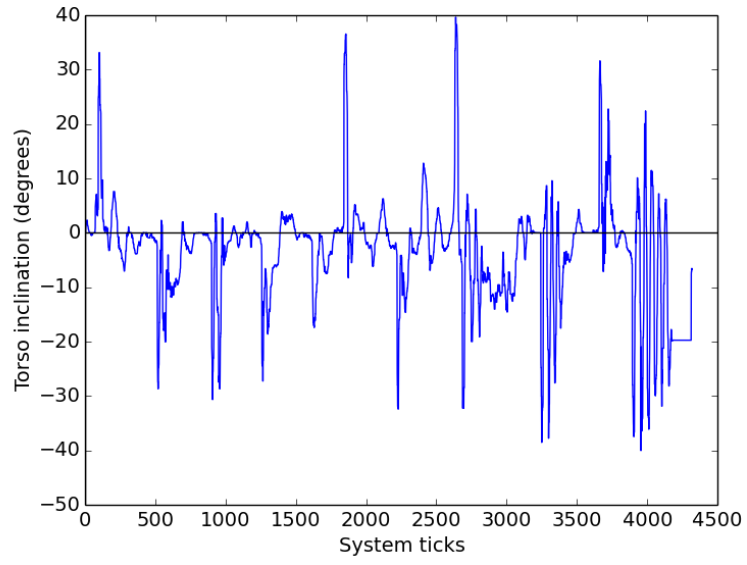


Figure B.9: Torso inclination during trial 5 with PID controller using Do the Shake policy

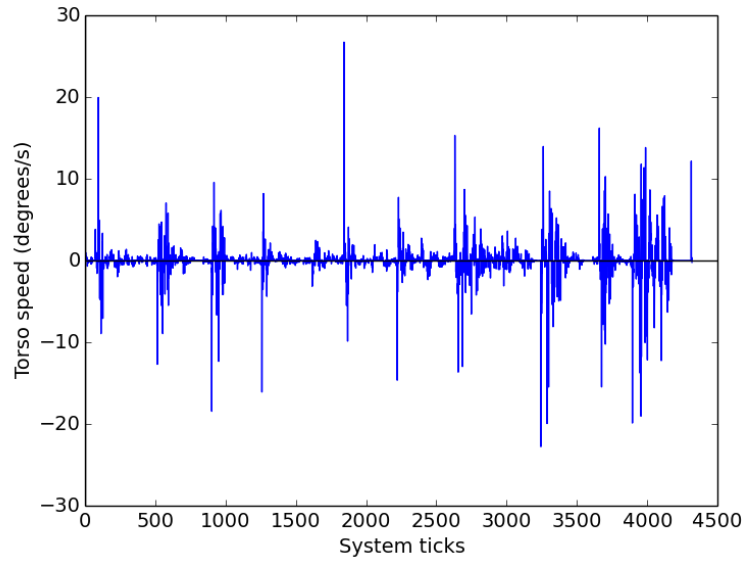


Figure B.10: Torso angular velocity during trial 5 with PID controller using Do the Shake policy

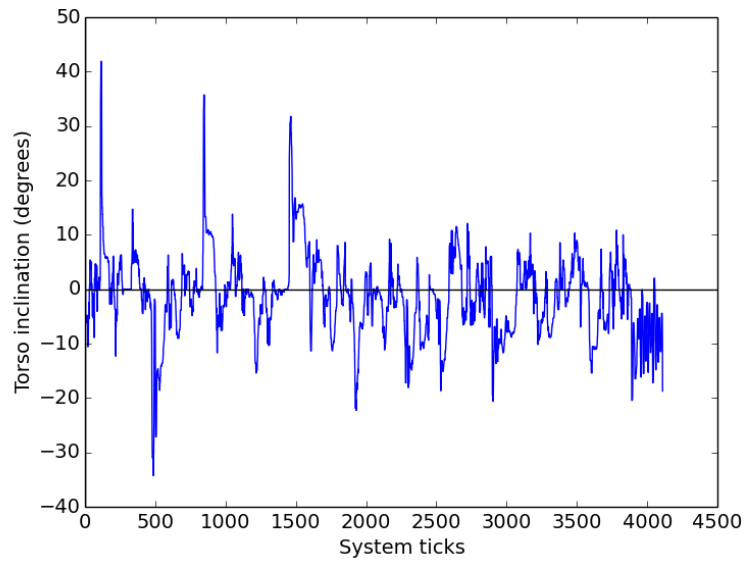


Figure B.11: Torso inclination during trial 1 with Fuzzy Logic controller using Do the Shake policy

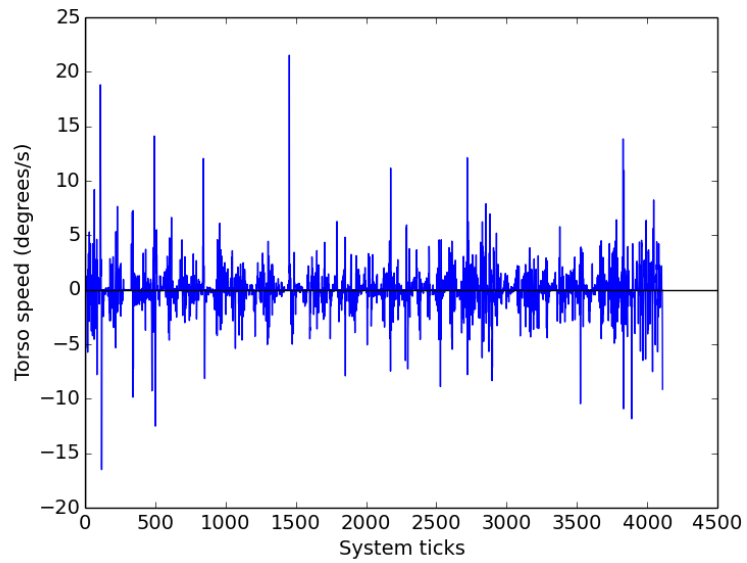


Figure B.12: Torso angular velocity during trial 1 with Fuzzy Logic controller using Do the Shake policy

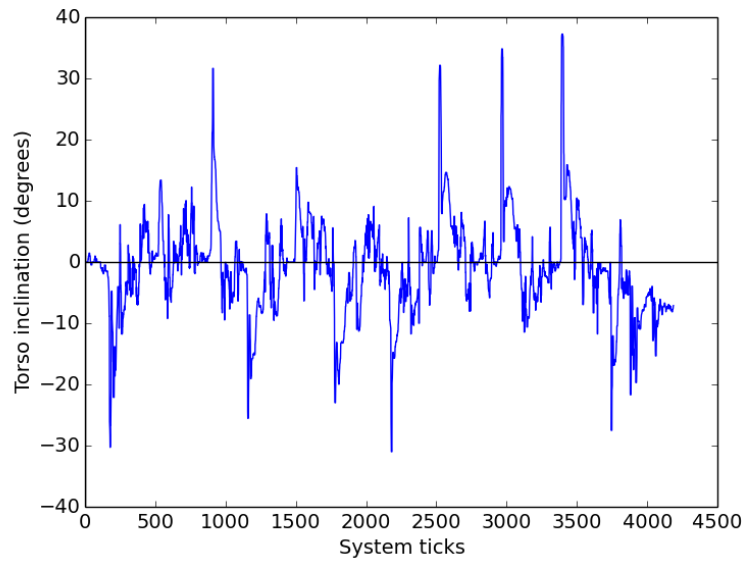


Figure B.13: Torso inclination during trial 2 with Fuzzy Logic controller using Do the Shake policy

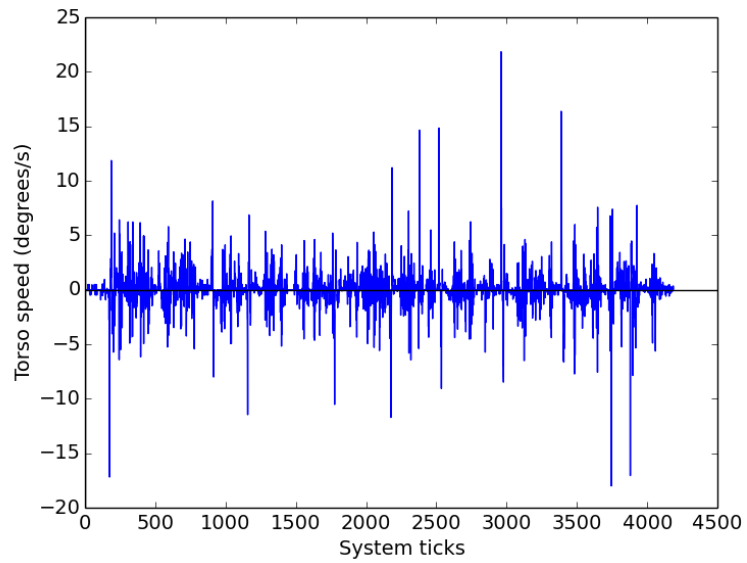


Figure B.14: Torso angular velocity during trial 2 with Fuzzy Logic controller using Do the Shake policy

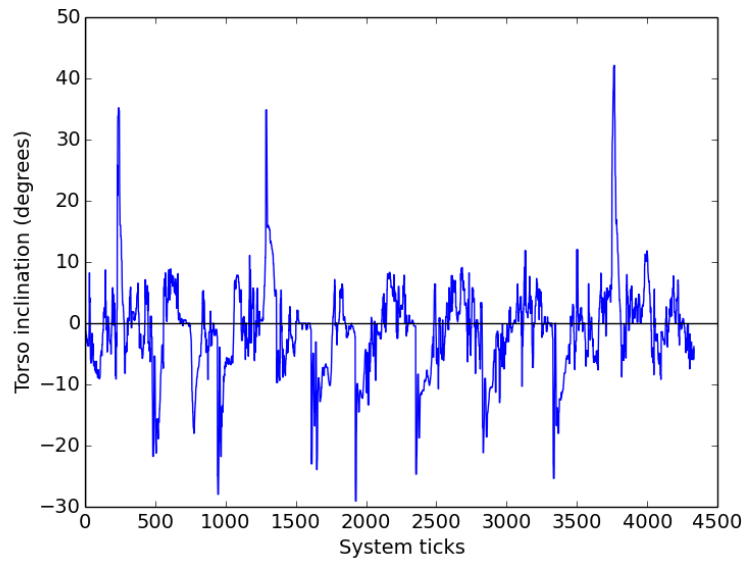


Figure B.15: Torso inclination during trial 3 with Fuzzy Logic controller using Do the Shake policy

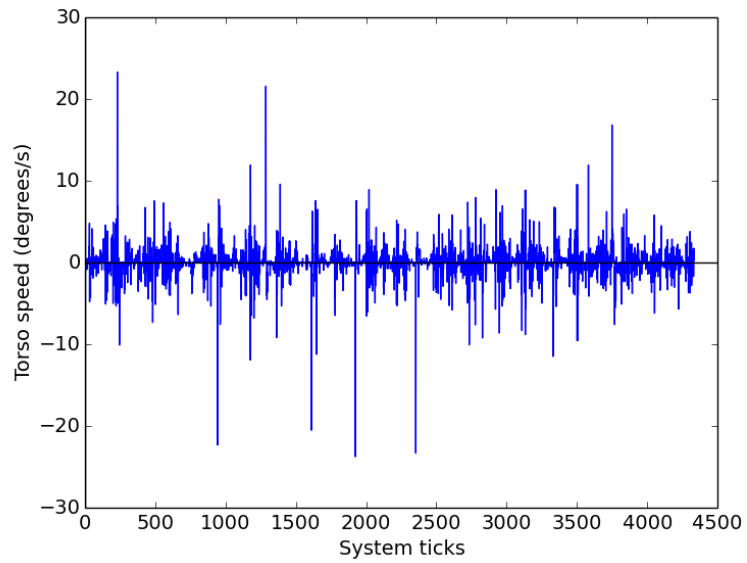


Figure B.16: Torso angular velocity during trial 3 with Fuzzy Logic controller using Do the Shake policy

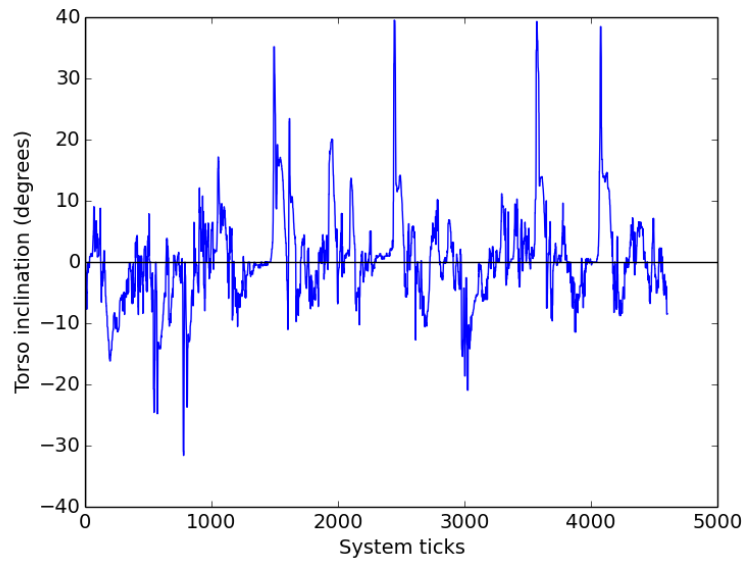


Figure B.17: Torso inclination during trial 4 with Fuzzy Logic controller using Do the Shake policy

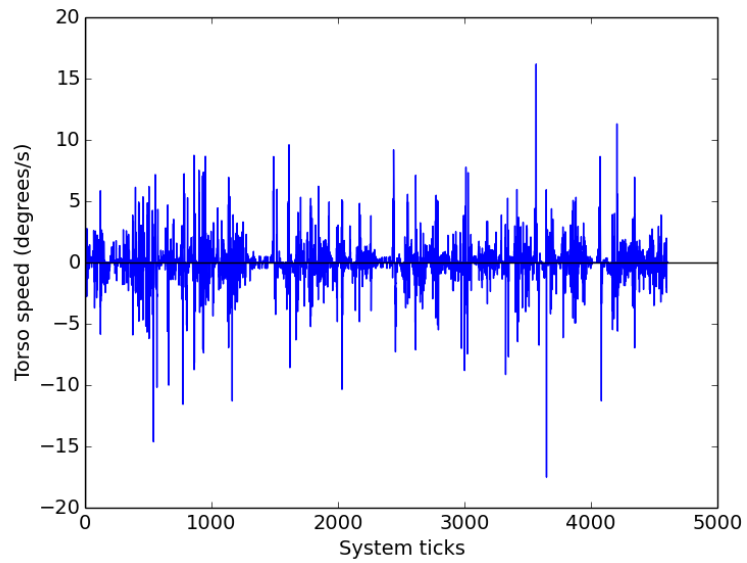


Figure B.18: Torso angular velocity during trial 4 with Fuzzy Logic controller using Do the Shake policy

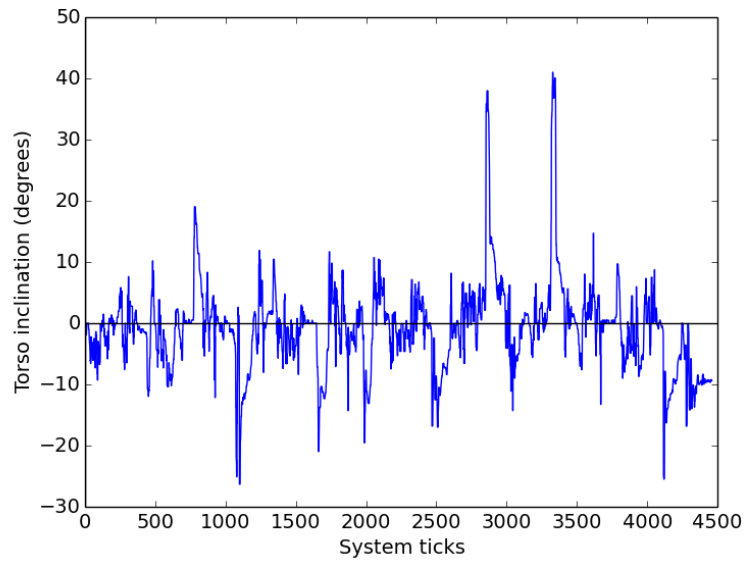


Figure B.19: Torso inclination during trial 5 with Fuzzy Logic controller using Do the Shake policy

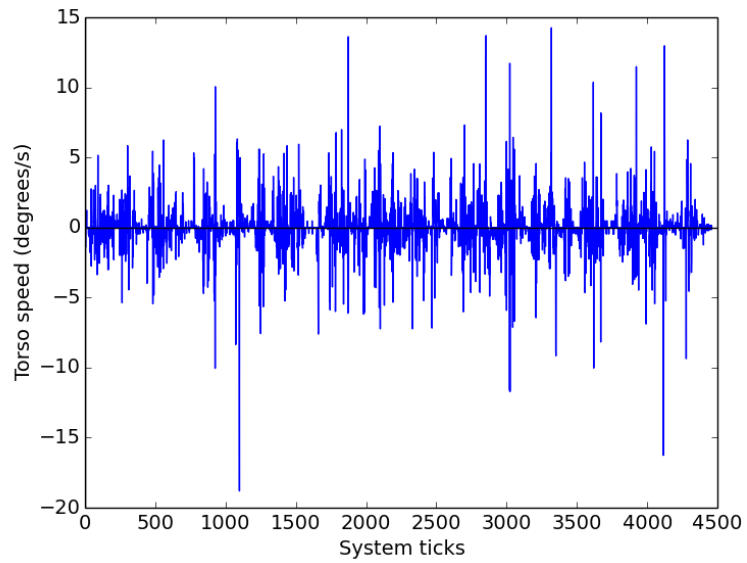


Figure B.20: Torso angular velocity during trial 5 with Fuzzy Logic controller using Do the Shake policy

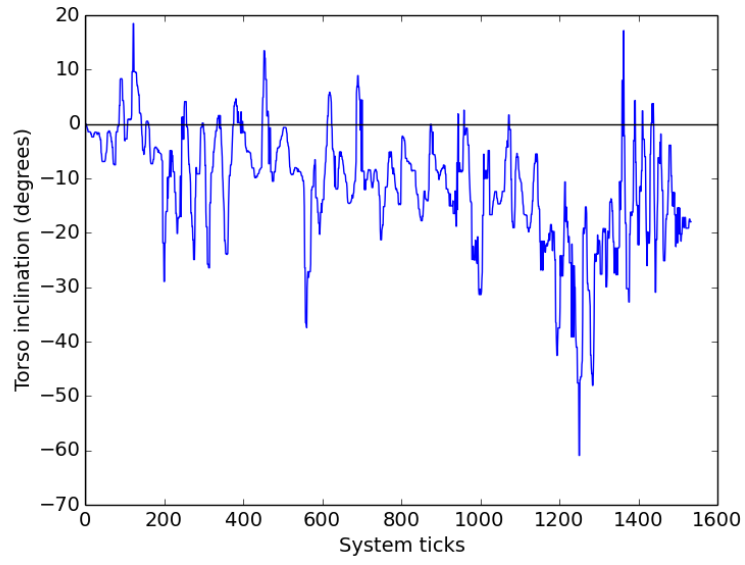


Figure B.21: Torso inclination during trial 1 with AOANN PID-28 controller using Do the Shake policy

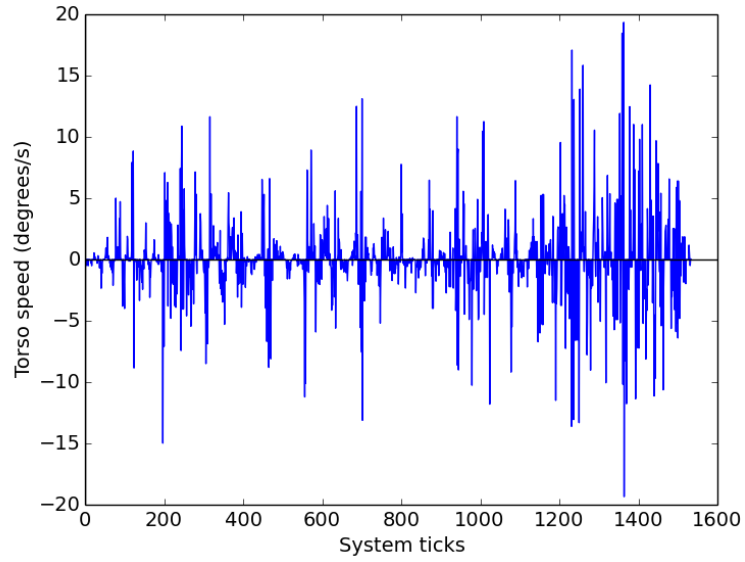


Figure B.22: Torso angular velocity during trial 1 with AOANN PID-28 controller using Do the Shake policy

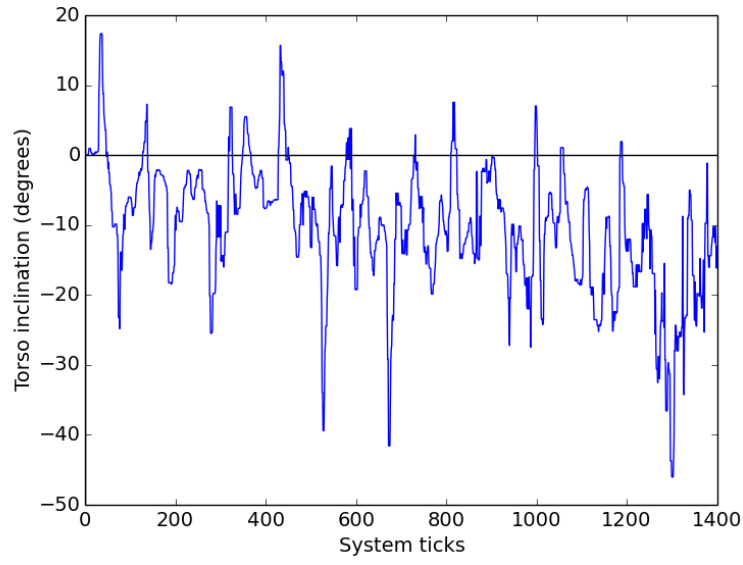


Figure B.23: Torso inclination during trial 2 with AOANN PID-28 controller using Do the Shake policy

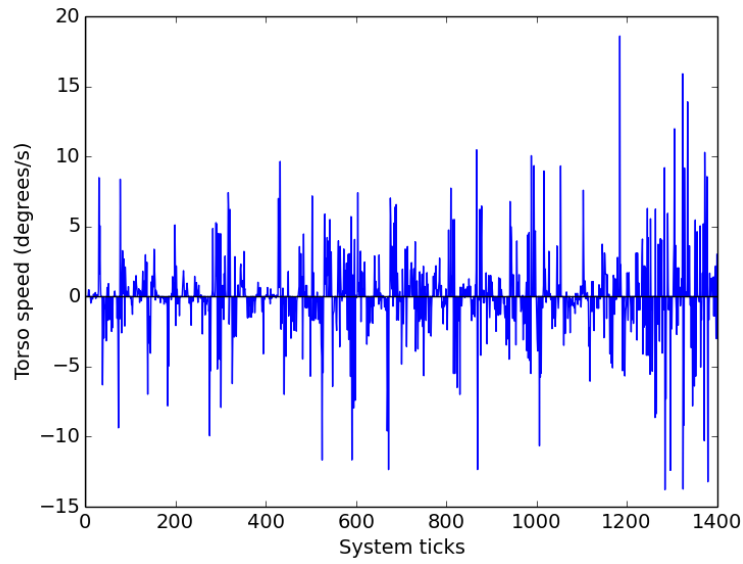


Figure B.24: Torso angular velocity during trial 2 with AOANN PID-28 controller using Do the Shake policy

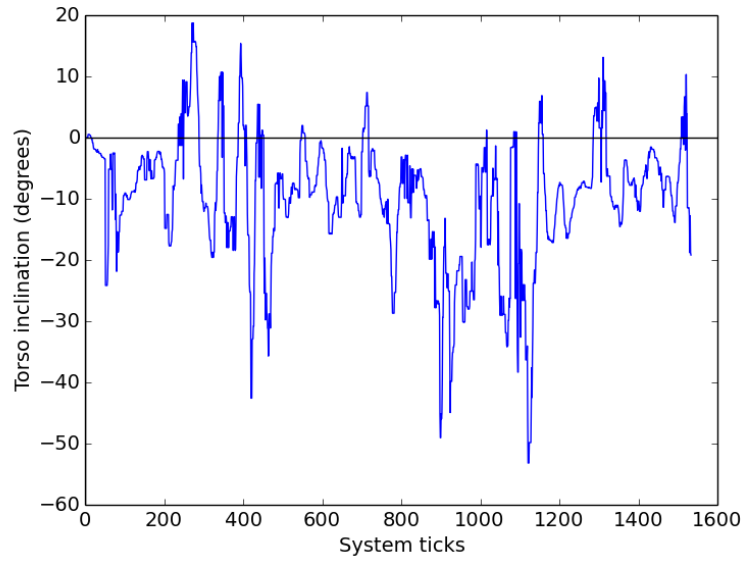


Figure B.25: Torso inclination during trial 3 with AOANN PID-28 controller using Do the Shake policy

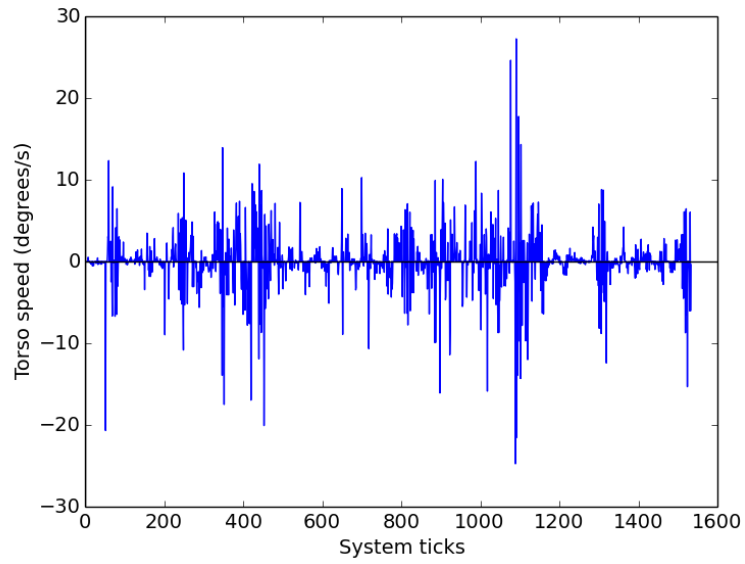


Figure B.26: Torso angular velocity during trial 3 with AOANN PID-28 controller using Do the Shake policy

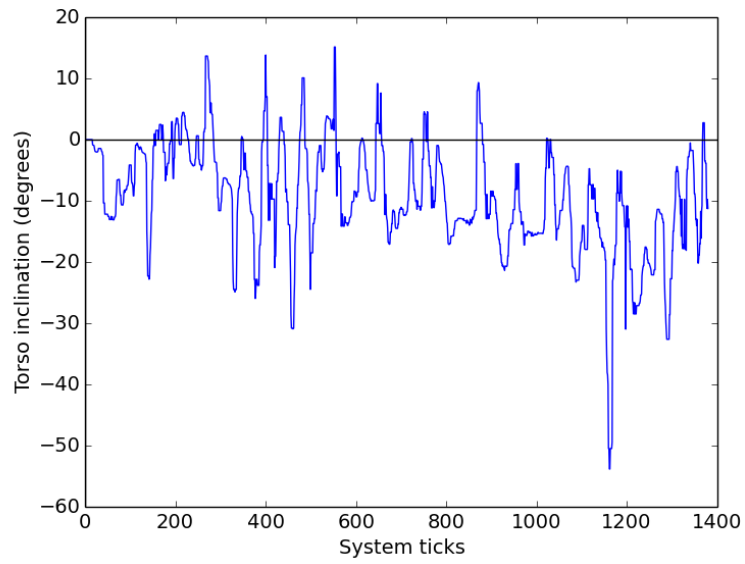


Figure B.27: Torso inclination during trial 4 with AOANN PID-28 controller using Do the Shake policy

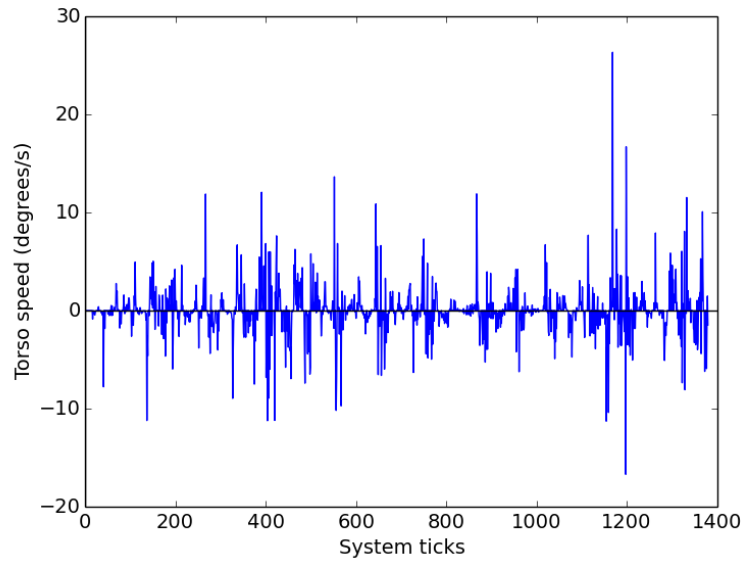


Figure B.28: Torso angular velocity during trial 4 with AOANN PID-28 controller using Do the Shake policy

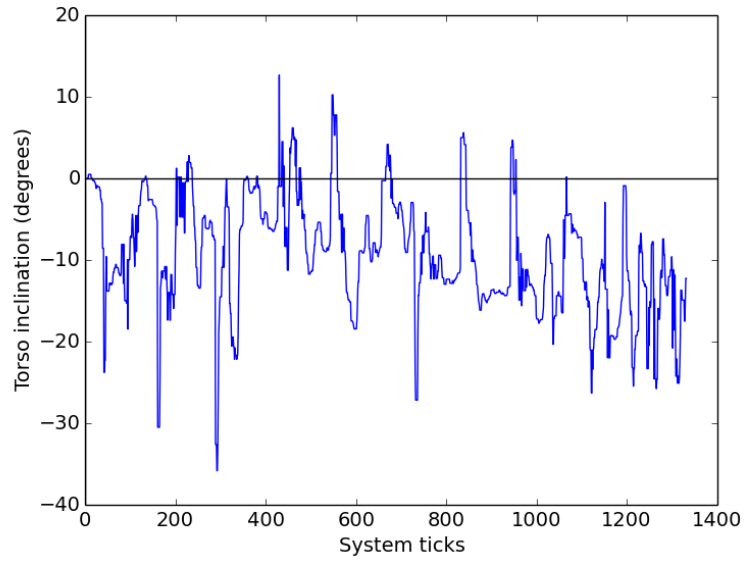


Figure B.29: Torso inclination during trial 5 with AOANN PID-28 controller using Do the Shake policy

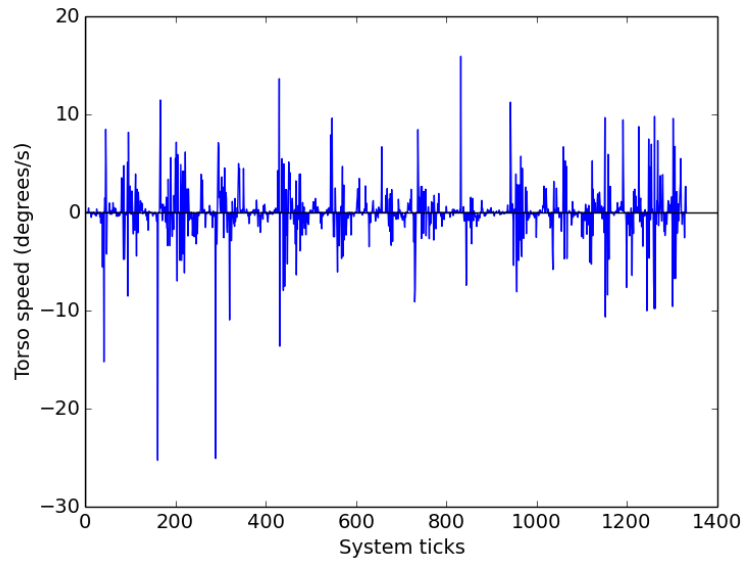


Figure B.30: Torso angular velocity during trial 5 with AOANN PID-28 controller using Do the Shake policy

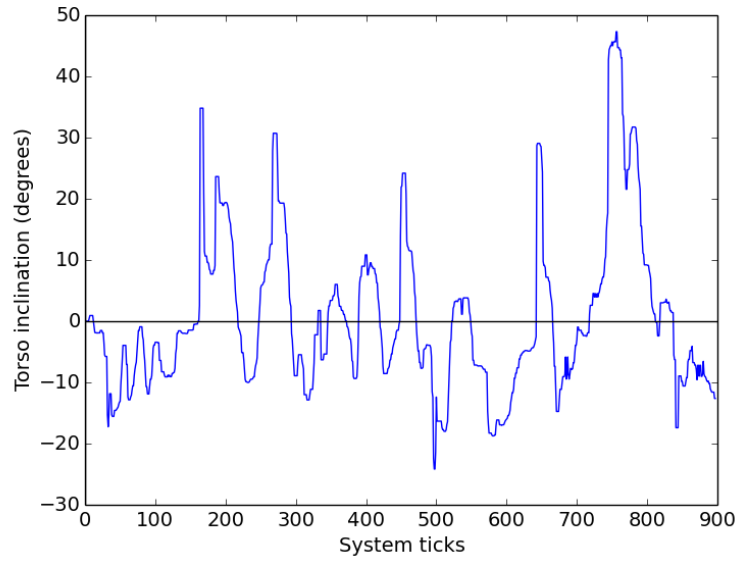


Figure B.31: Torso inclination during trial 1 with AOANN Fuz-28 controller using Do the Shake policy

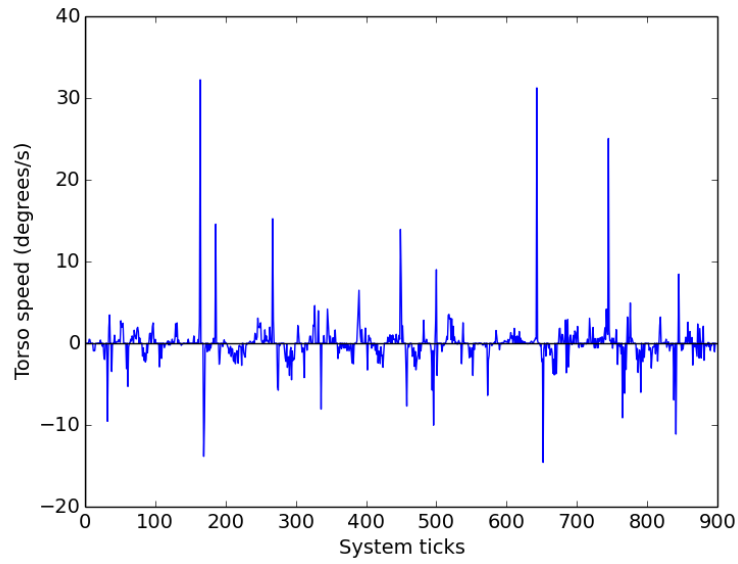


Figure B.32: Torso angular velocity during trial 1 with AOANN Fuz-28 controller using Do the Shake policy

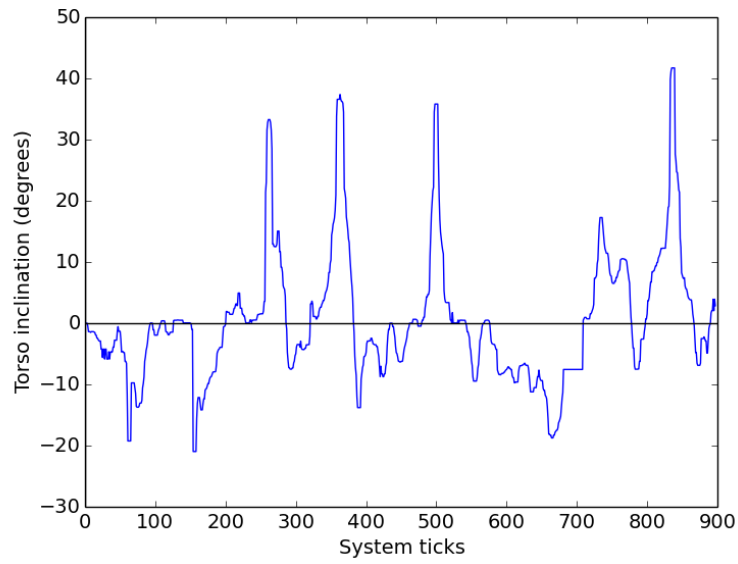


Figure B.33: Torso inclination during trial 2 with AOANN Fuz-28 controller using Do the Shake policy

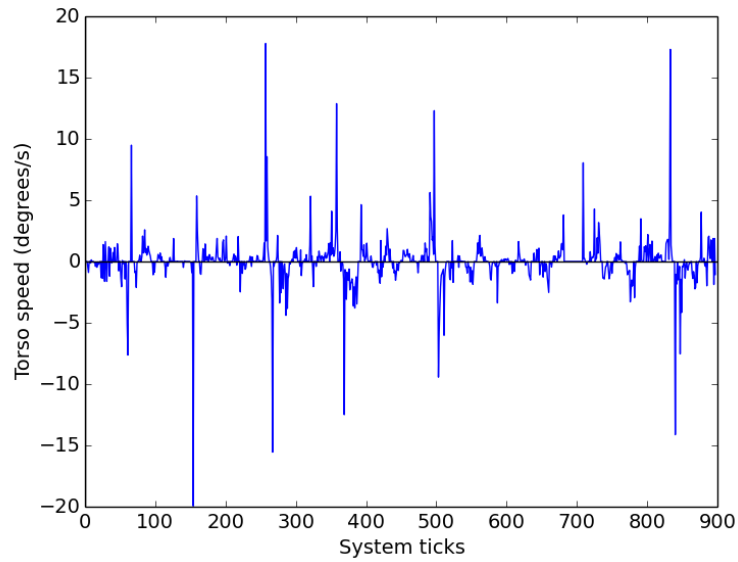


Figure B.34: Torso angular velocity during trial 2 with AOANN Fuz-28 controller using Do the Shake policy

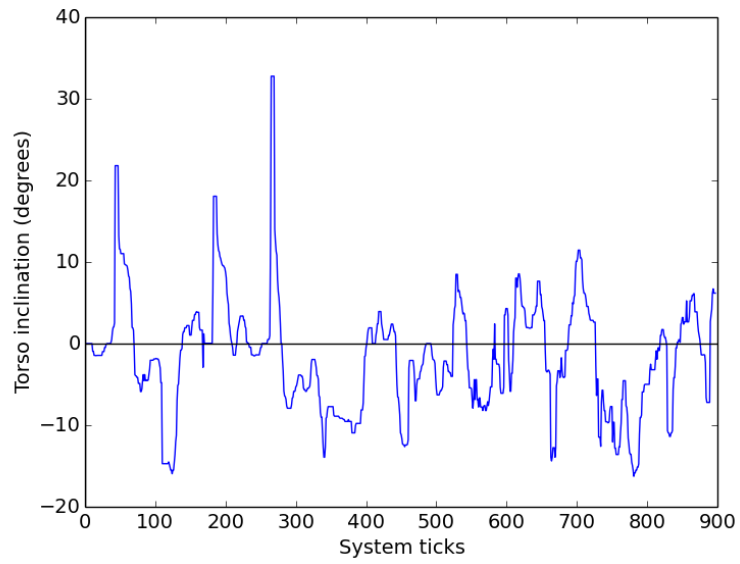


Figure B.35: Torso inclination during trial 3 with AOANN Fuz-28 controller using Do the Shake policy

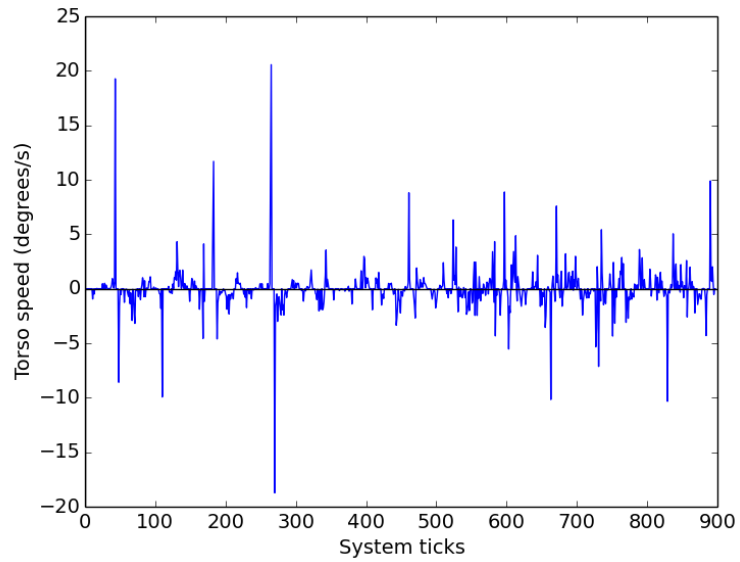


Figure B.36: Torso angular velocity during trial 3 with AOANN Fuz-28 controller using Do the Shake policy

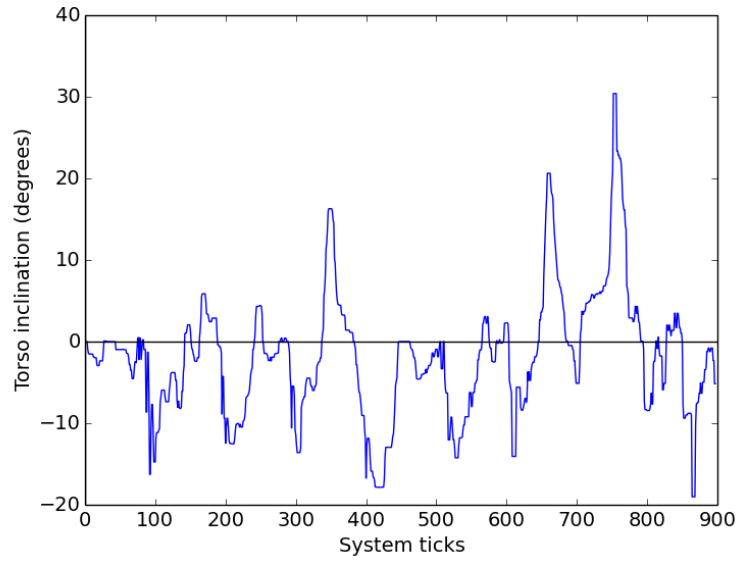


Figure B.37: Torso inclination during trial 4 with AOANN Fuz-28 controller using Do the Shake policy

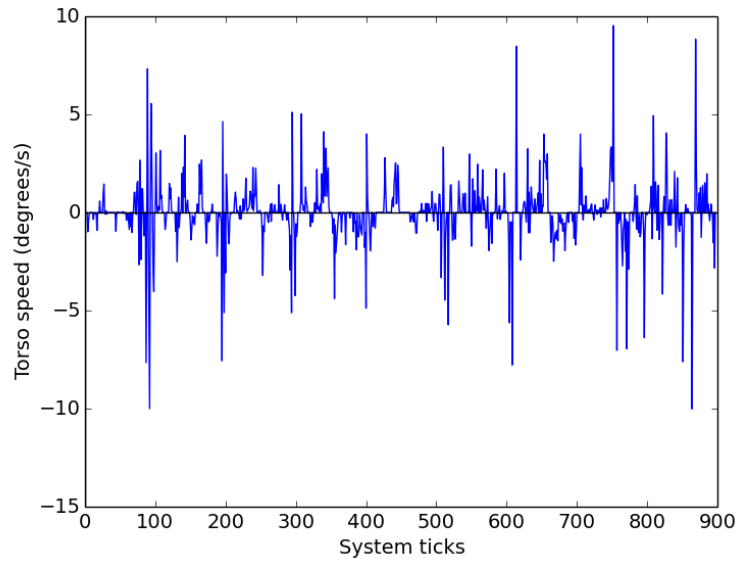


Figure B.38: Torso angular velocity during trial 4 with AOANN Fuz-28 controller using Do the Shake policy

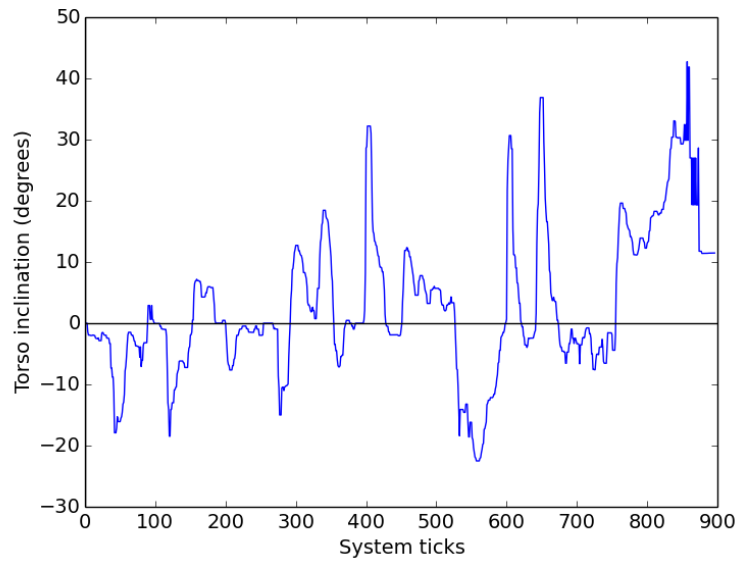


Figure B.39: Torso inclination during trial 5 with AOANN Fuz-28 controller using Do the Shake policy

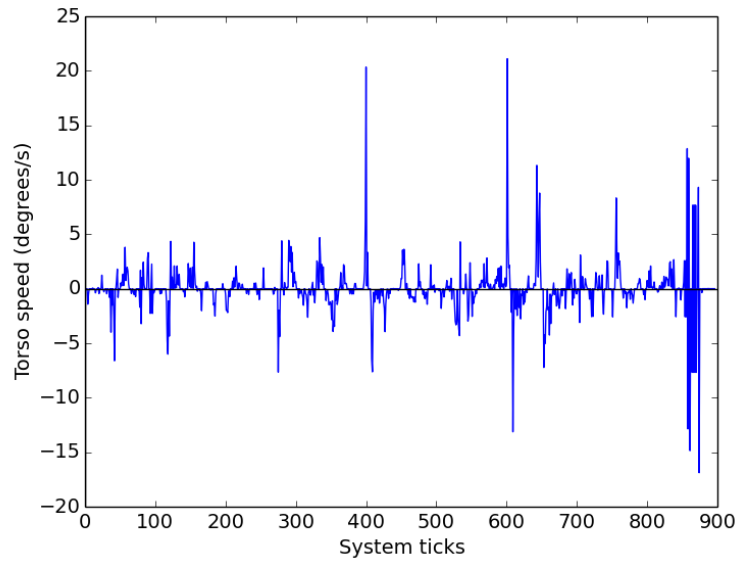


Figure B.40: Torso angular velocity during trial 5 with AOANN Fuz-28 controller using Do the Shake policy

B.2 Let's Sway Policy

This section shows the sensor data recorded when using the Let's Sway policy.

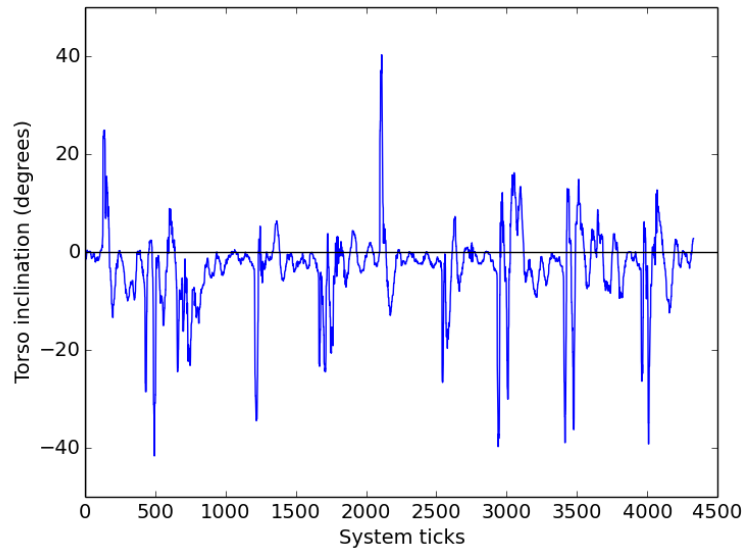


Figure B.41: Torso inclination during trial 1 with PID controller using Let's Sway policy

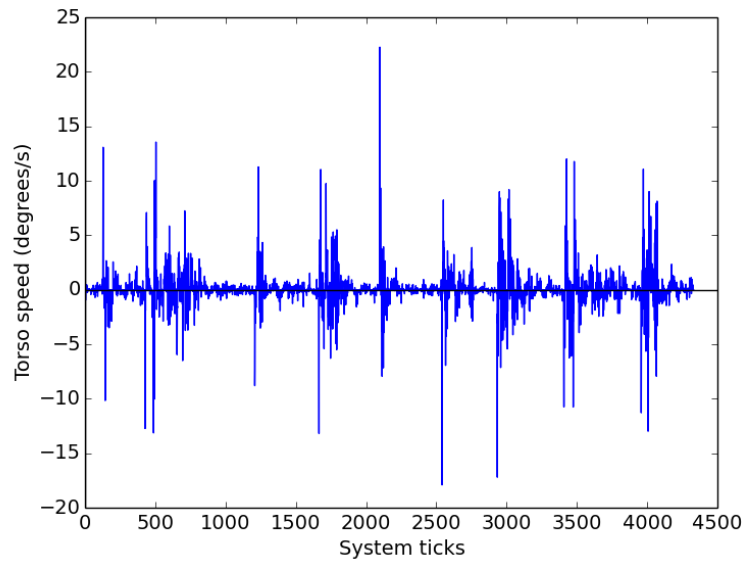


Figure B.42: Torso angular velocity during trial 1 with PID controller using Let's Sway policy

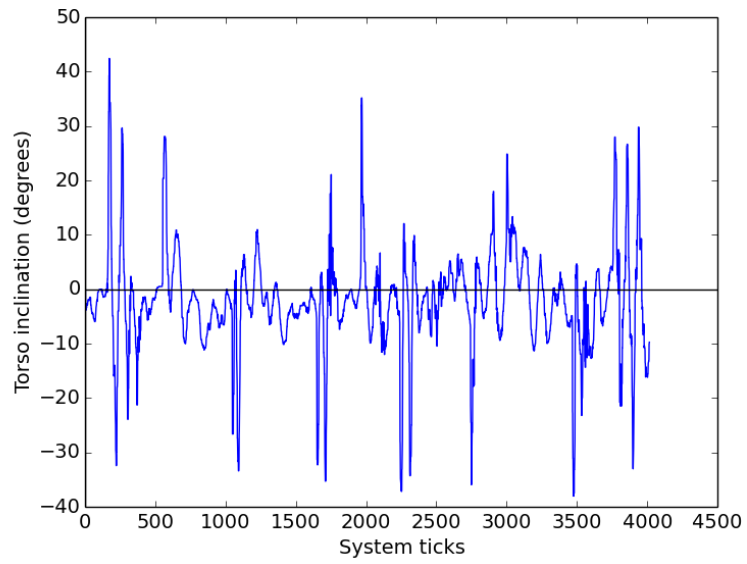


Figure B.43: Torso inclination during trial 2 with PID controller using Let's Sway policy

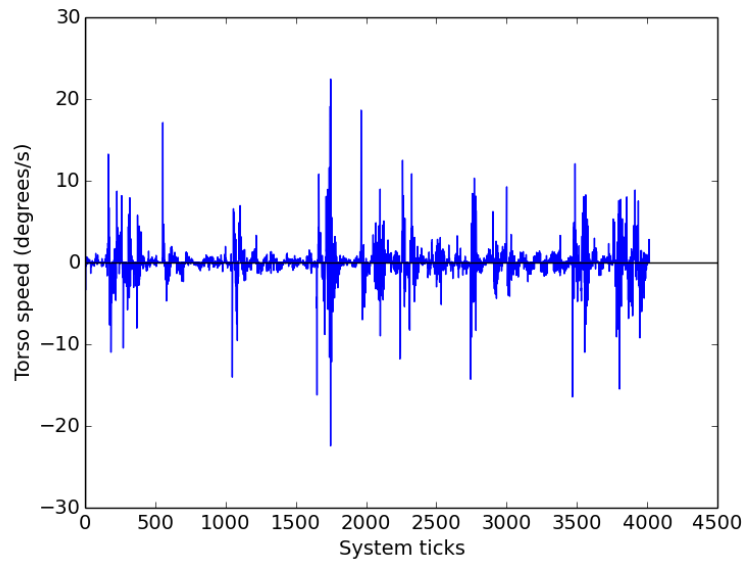


Figure B.44: Torso angular velocity during trial 2 with PID controller using Let's Sway policy

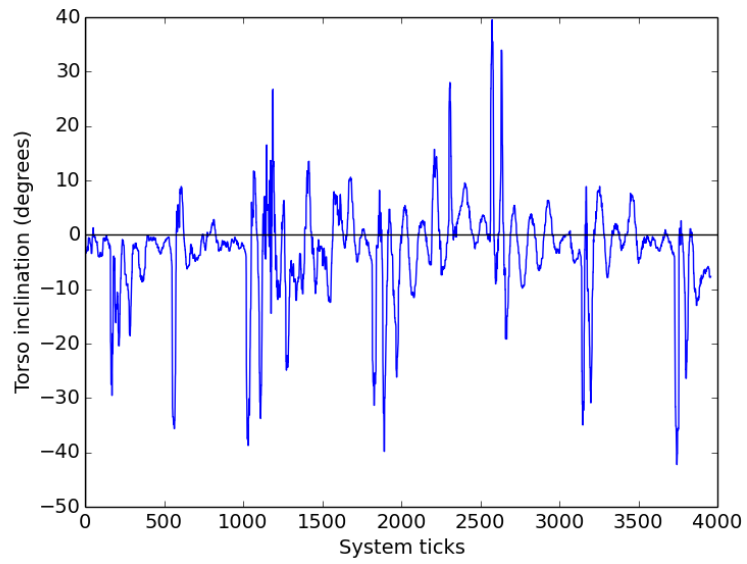


Figure B.45: Torso inclination during trial 3 with PID controller using Let's Sway policy

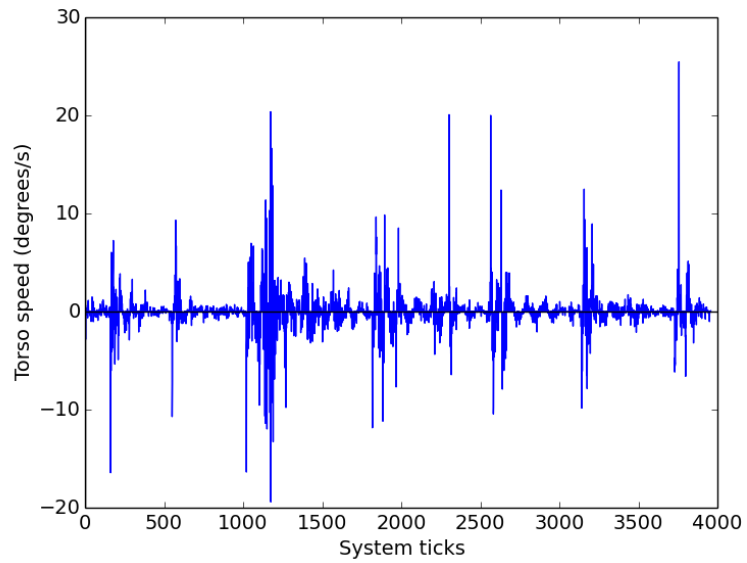


Figure B.46: Torso angular velocity during trial 3 with PID controller using Let's Sway policy

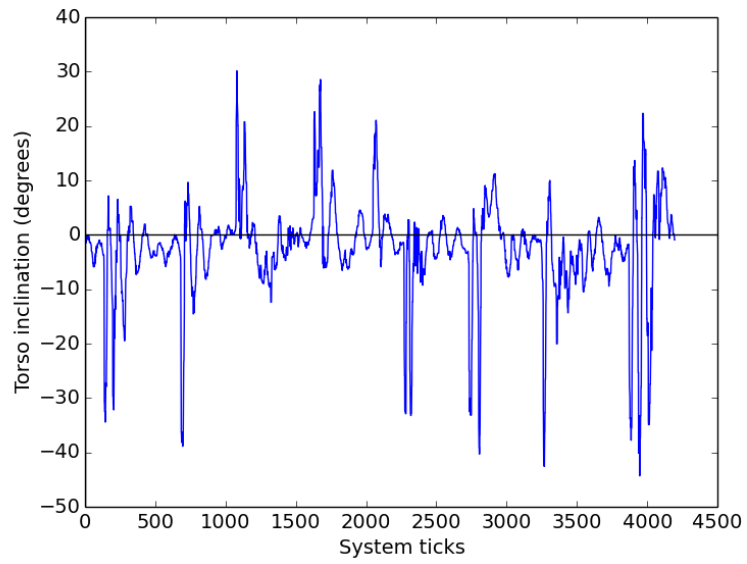


Figure B.47: Torso inclination during trial 4 with PID controller using Let's Sway policy

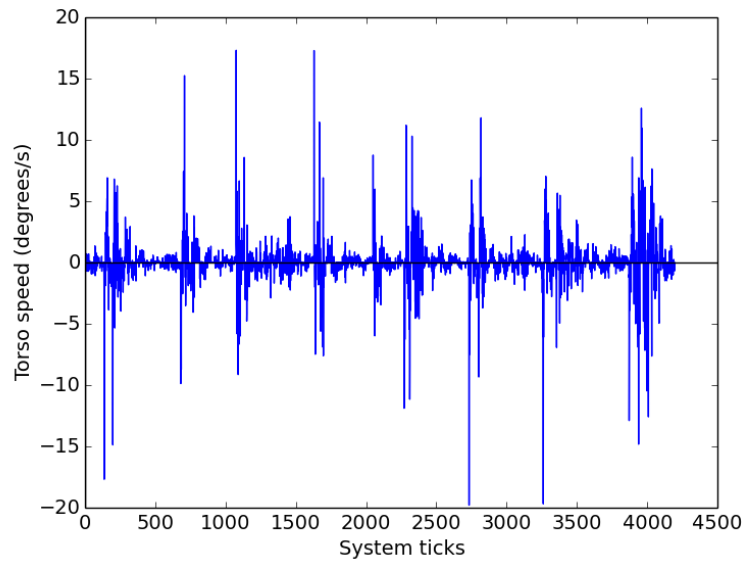


Figure B.48: Torso angular velocity during trial 4 with PID controller using Let's Sway policy

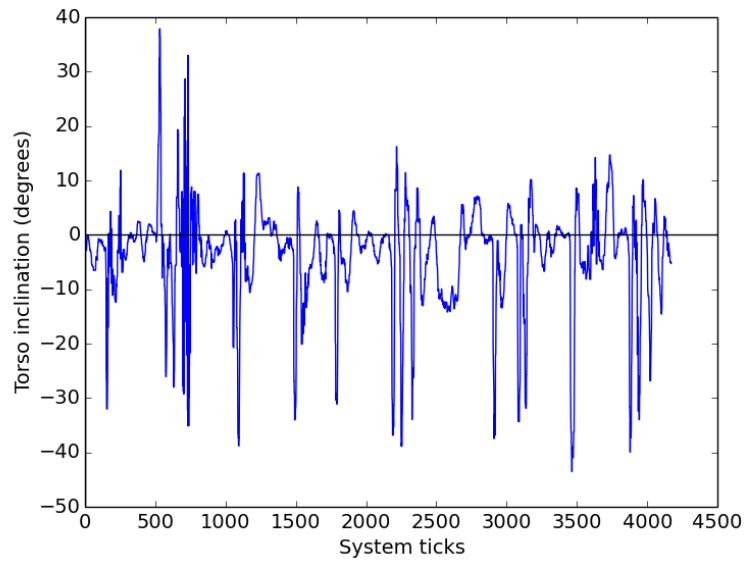


Figure B.49: Torso inclination during trial 5 with PID controller using Let's Sway policy

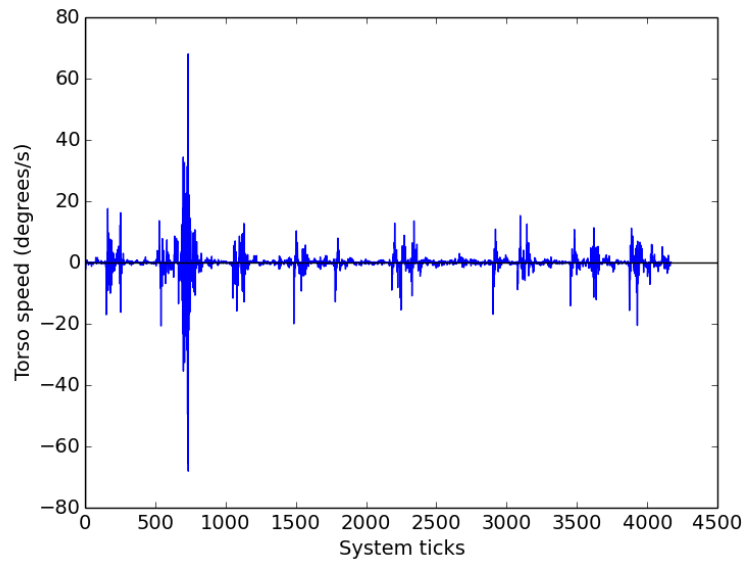


Figure B.50: Torso angular velocity during trial 5 with PID controller using Let's Sway policy

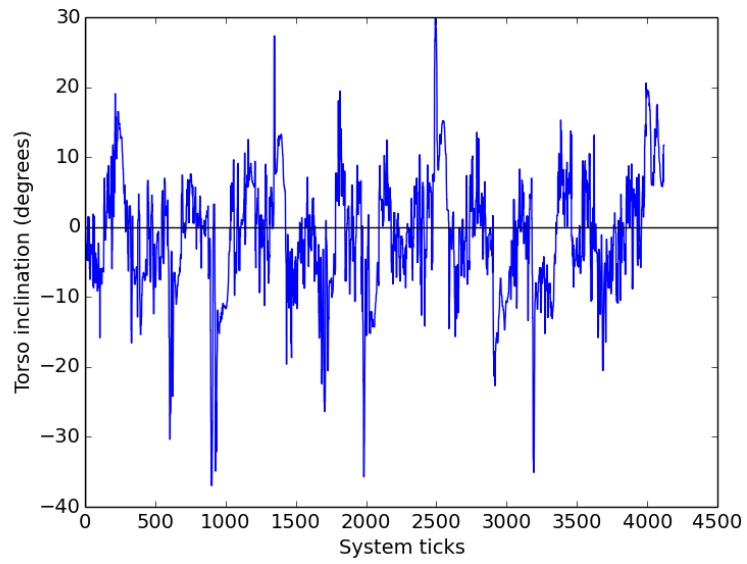


Figure B.51: Torso inclination during trial 1 with Fuzzy Logic controller using Let's Sway policy

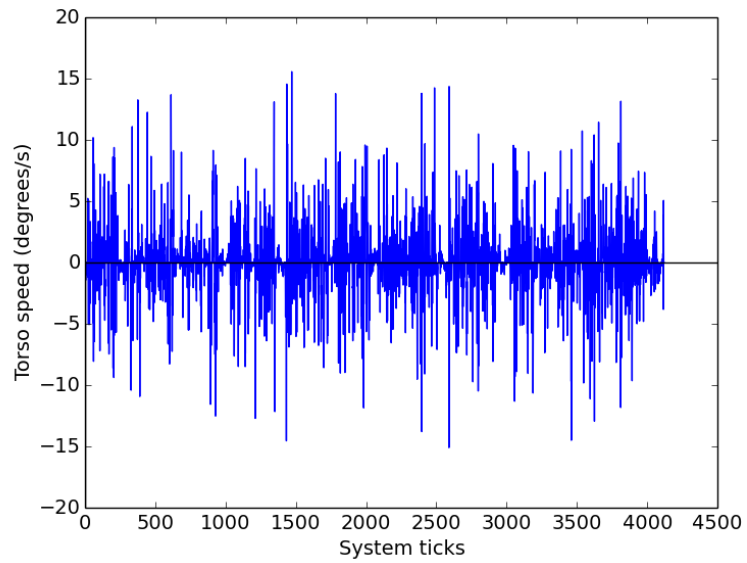


Figure B.52: Torso angular velocity during trial 1 with Fuzzy Logic controller using Let's Sway policy

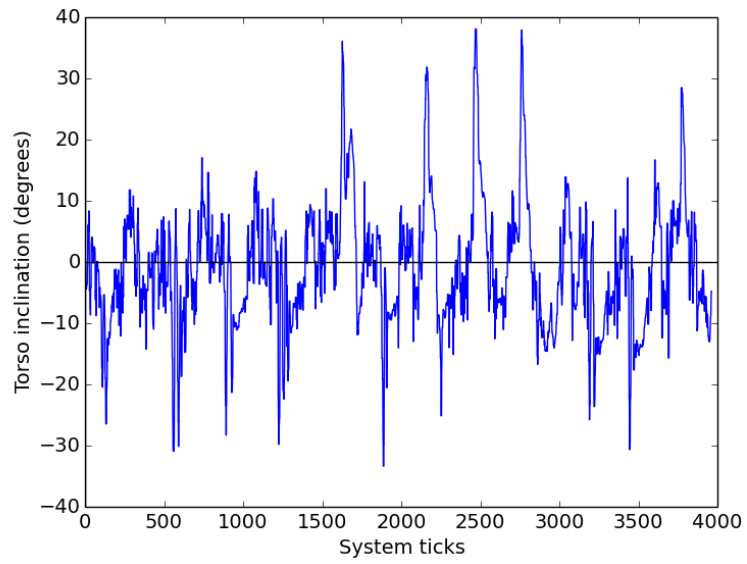


Figure B.53: Torso inclination during trial 2 with Fuzzy Logic controller using Let's Sway policy

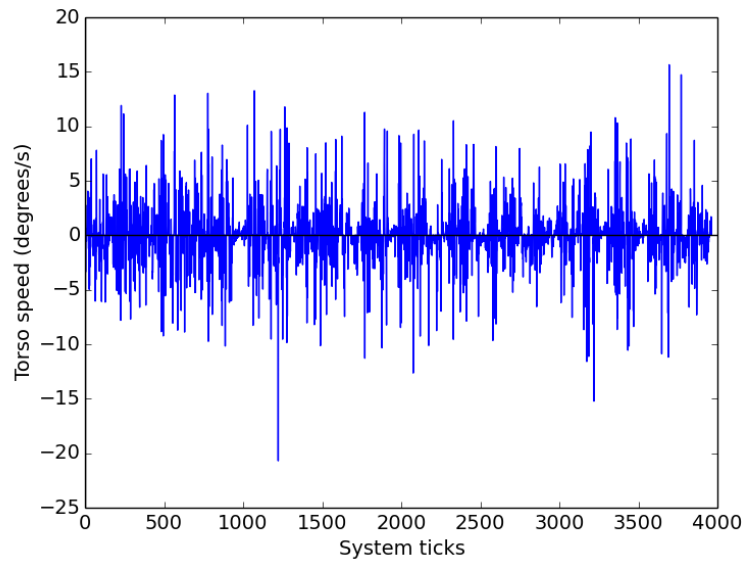


Figure B.54: Torso angular velocity during trial 2 with Fuzzy Logic controller using Let's Sway policy

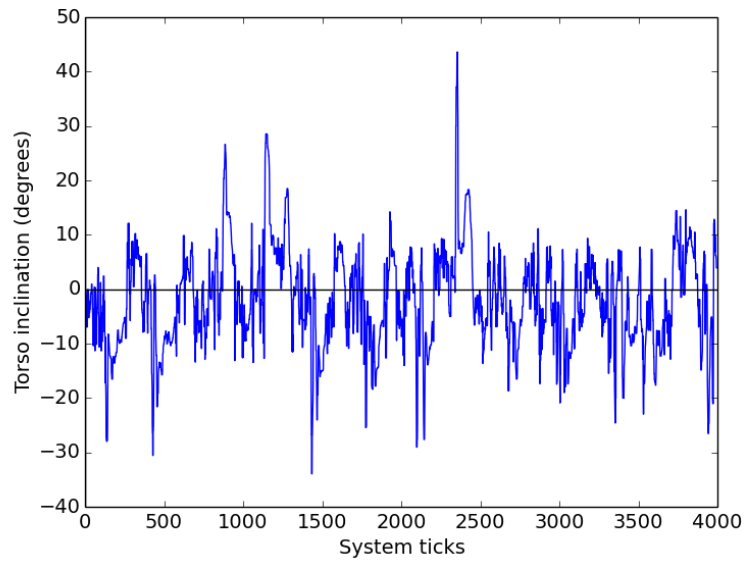


Figure B.55: Torso inclination during trial 3 with Fuzzy Logic controller using Let's Sway policy

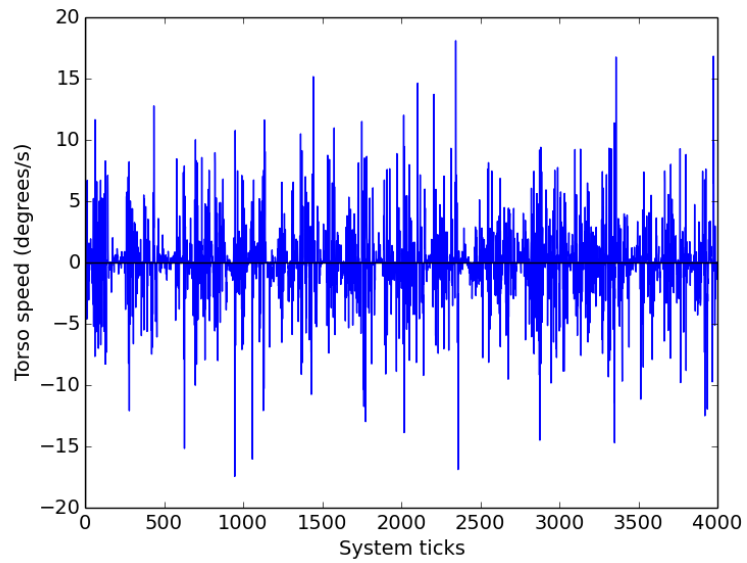


Figure B.56: Torso angular velocity during trial 3 with Fuzzy Logic controller using Let's Sway policy

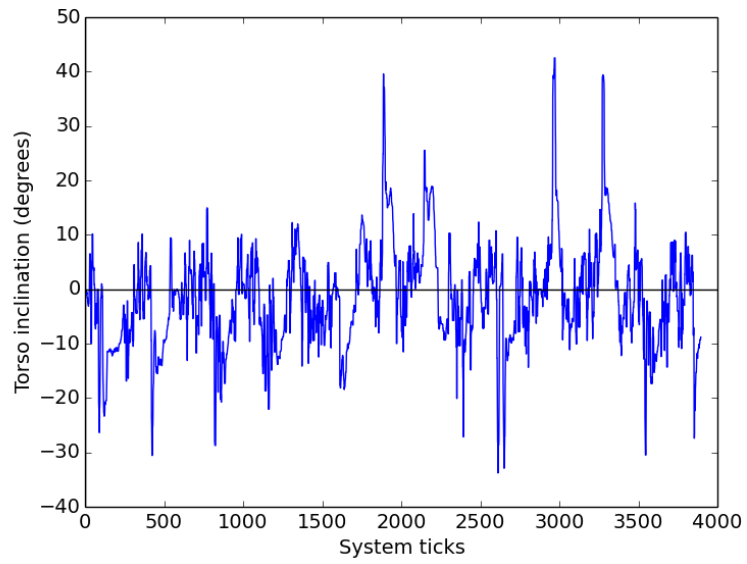


Figure B.57: Torso inclination during trial 4 with Fuzzy Logic controller using Let's Sway policy

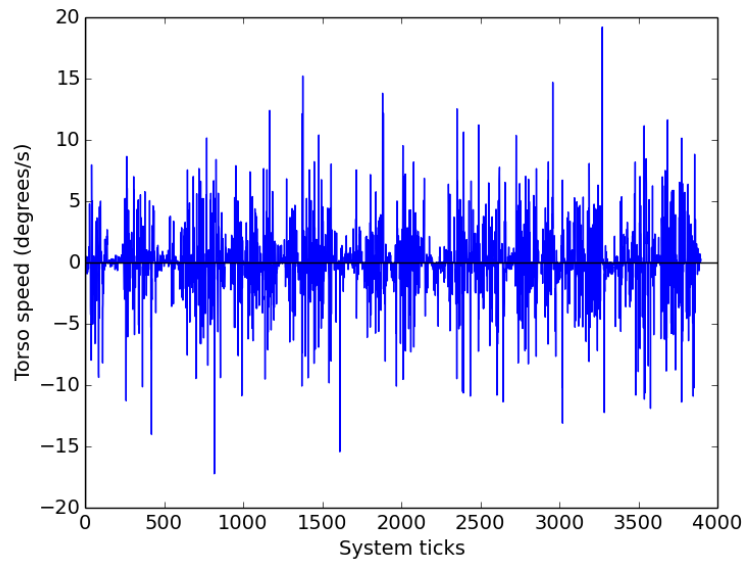


Figure B.58: Torso angular velocity during trial 4 with Fuzzy Logic controller using Let's Sway policy

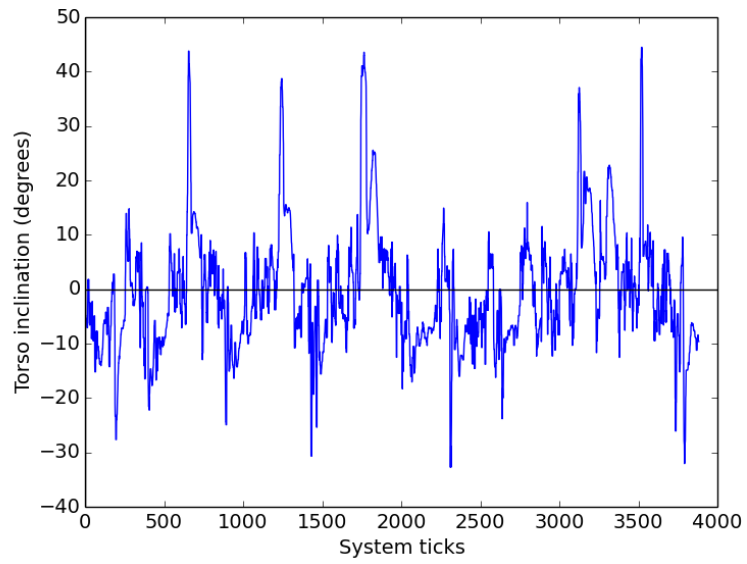


Figure B.59: Torso inclination during trial 5 with Fuzzy Logic controller using Let's Sway policy

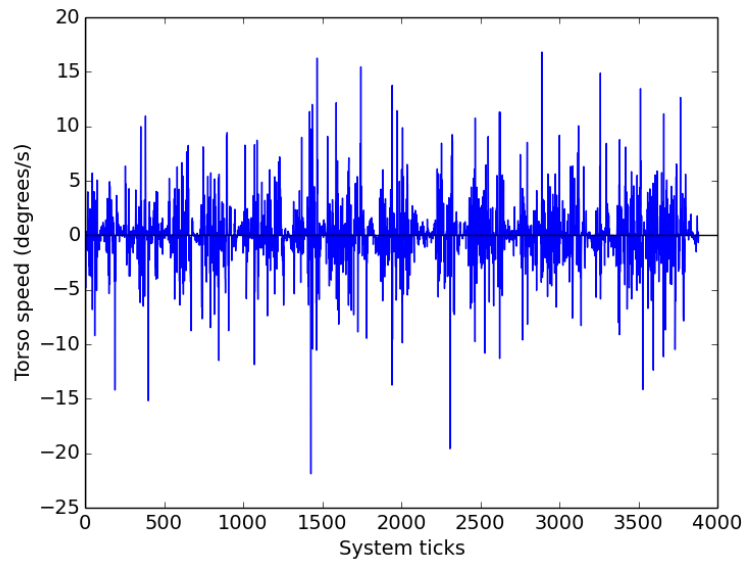


Figure B.60: Torso angular velocity during trial 5 with Fuzzy Logic controller using Let's Sway policy

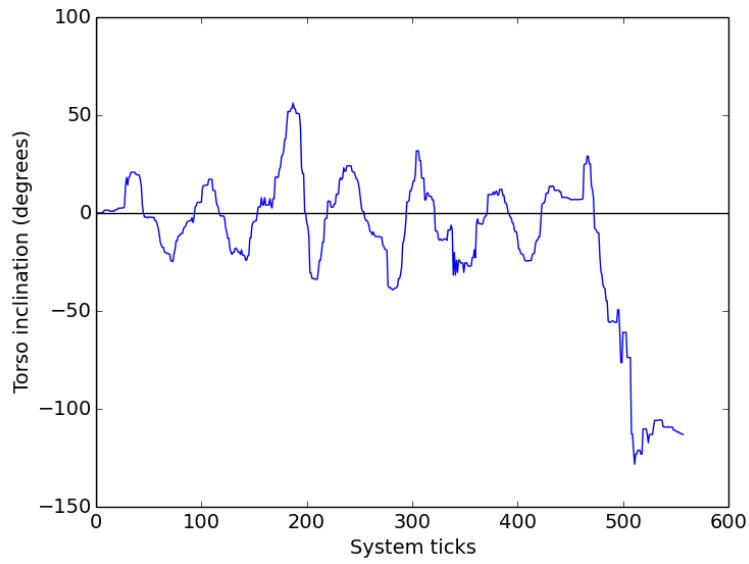


Figure B.61: Torso inclination during trial 1 with AOANN PID-28 controller using Let's Sway policy

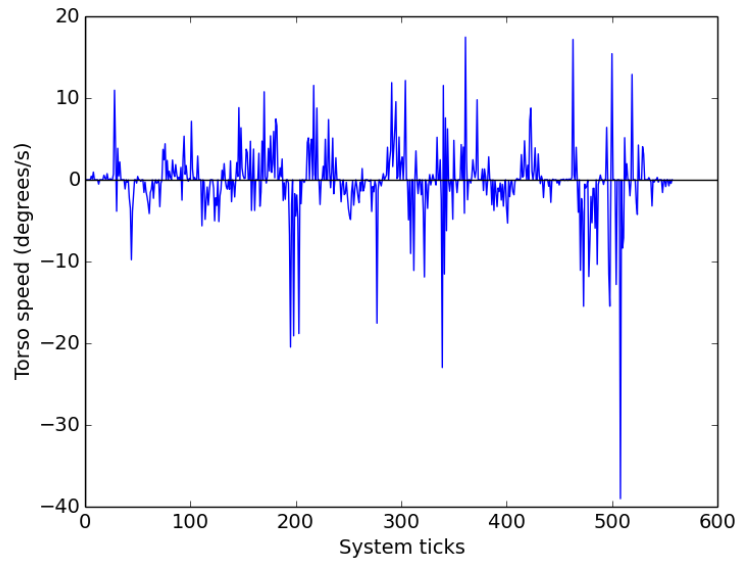


Figure B.62: Torso angular velocity during trial 1 with AOANN PID-28 controller using Let's Sway policy

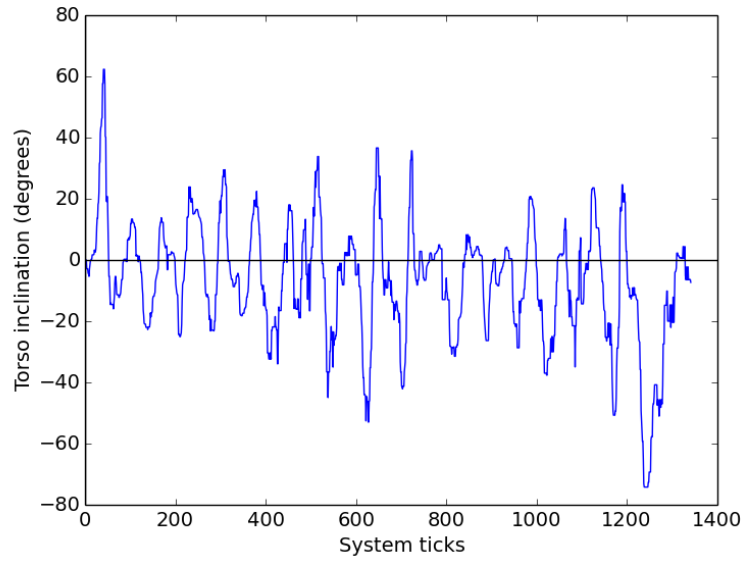


Figure B.63: Torso inclination during trial 2 with AOANN PID-28 controller using Let's Sway policy

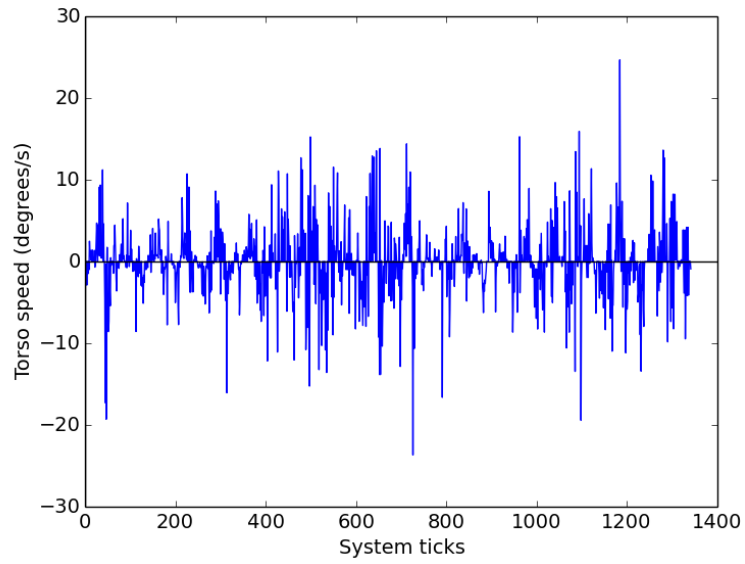


Figure B.64: Torso angular velocity during trial 2 with AOANN PID-28 controller using Let's Sway policy

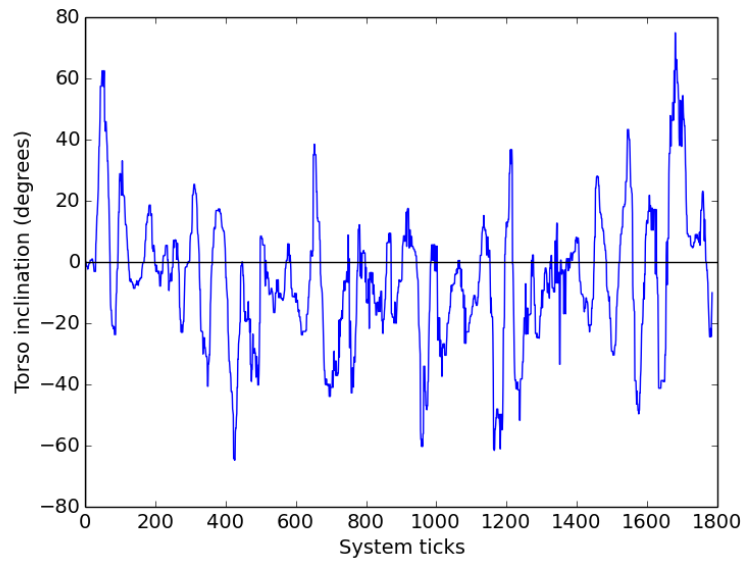


Figure B.65: Torso inclination during trial 3 with AOANN PID-28 controller using Let's Sway policy

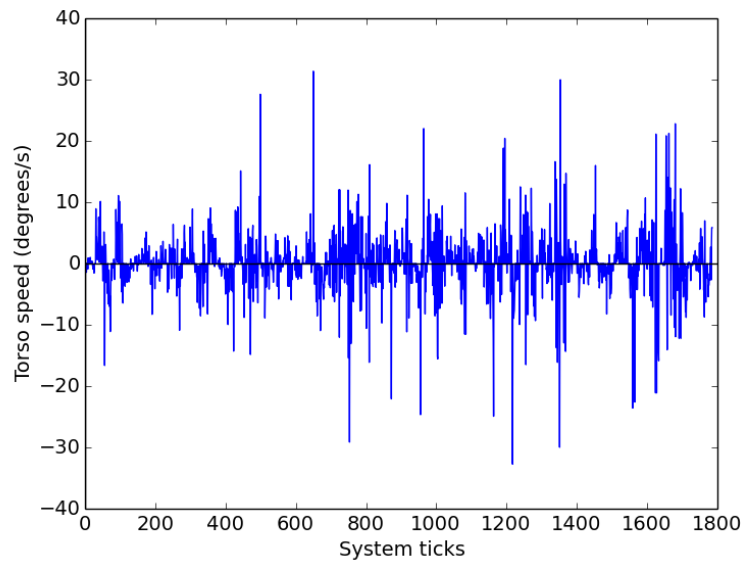


Figure B.66: Torso angular velocity during trial 3 with AOANN PID-28 controller using Let's Sway policy

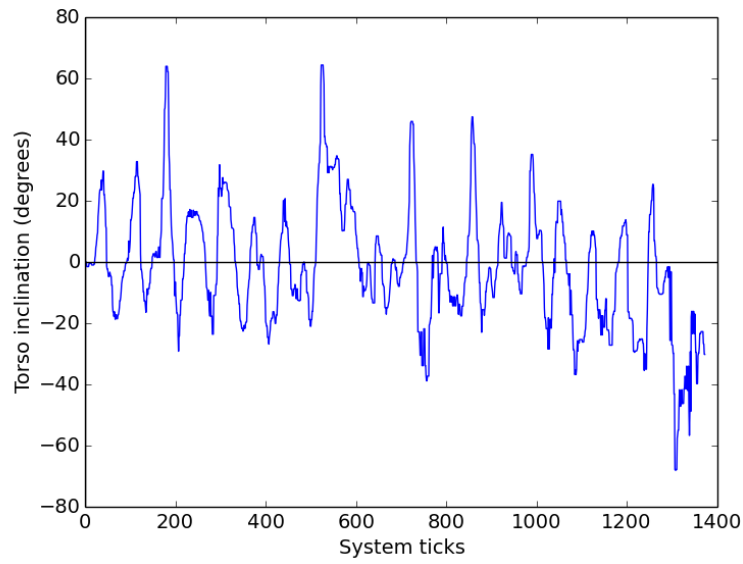


Figure B.67: Torso inclination during trial 4 with AOANN PID-28 controller using Let's Sway policy

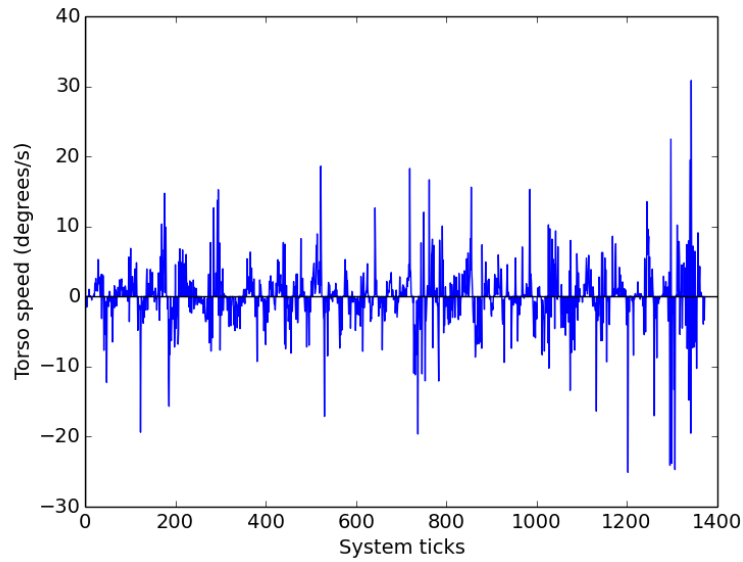


Figure B.68: Torso angular velocity during trial 4 with AOANN PID-28 controller using Let's Sway policy

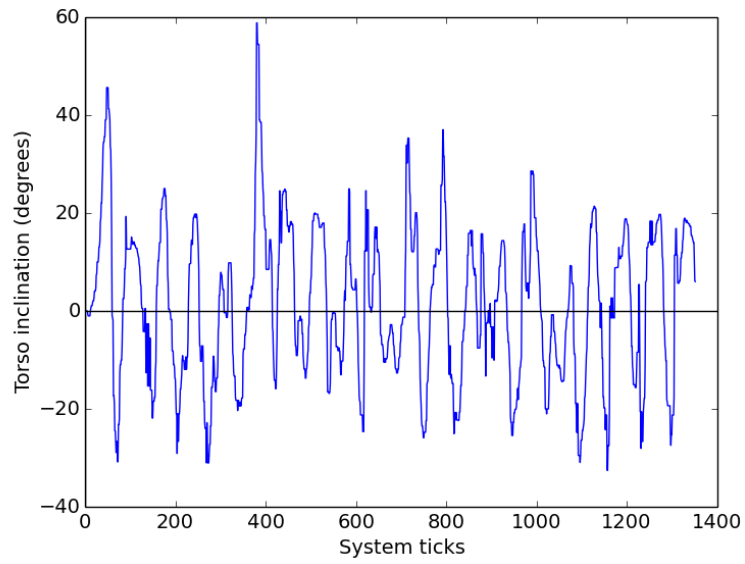


Figure B.69: Torso inclination during trial 5 with AOANN PID-28 controller using Let's Sway policy

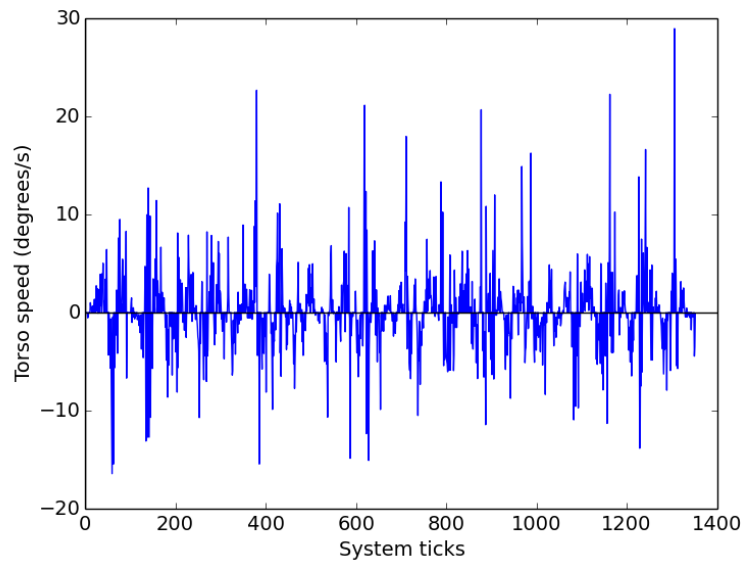


Figure B.70: Torso angular velocity during trial 5 with AOANN PID-28 controller using Let's Sway policy

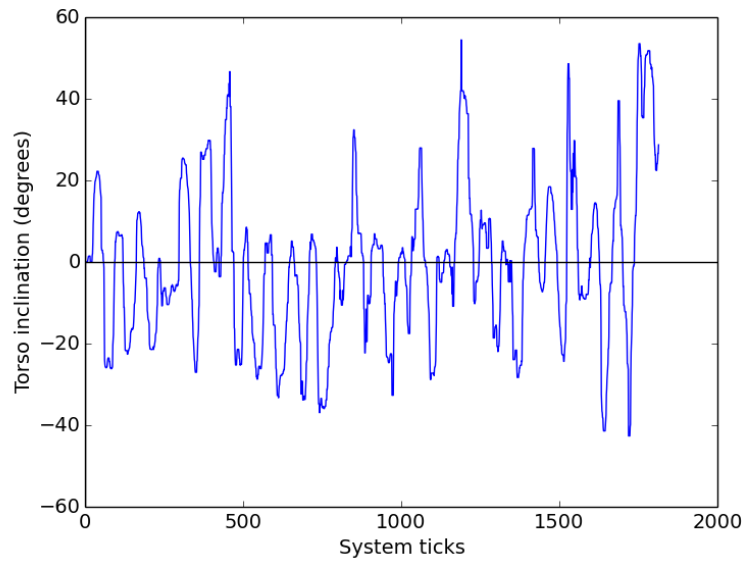


Figure B.71: Torso inclination during trial 1 with AOANN Fuz-28 controller using Let's Sway policy

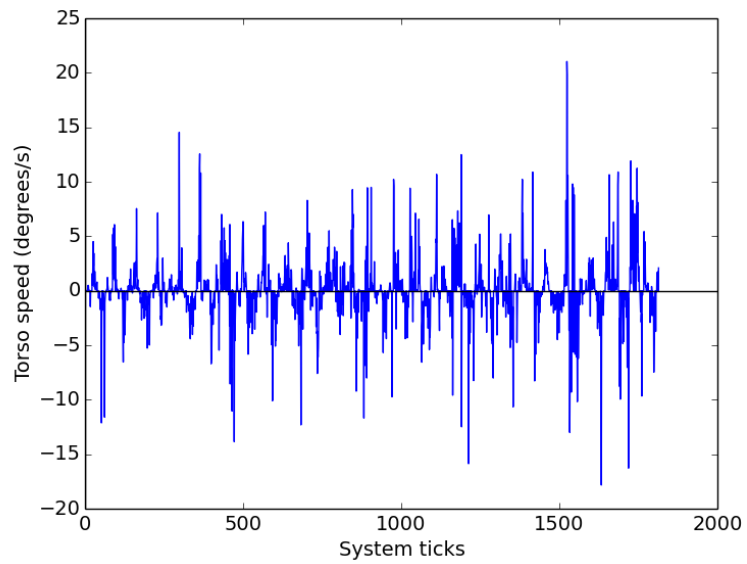


Figure B.72: Torso angular velocity during trial 1 with AOANN Fuz-28 controller using Let's Sway policy

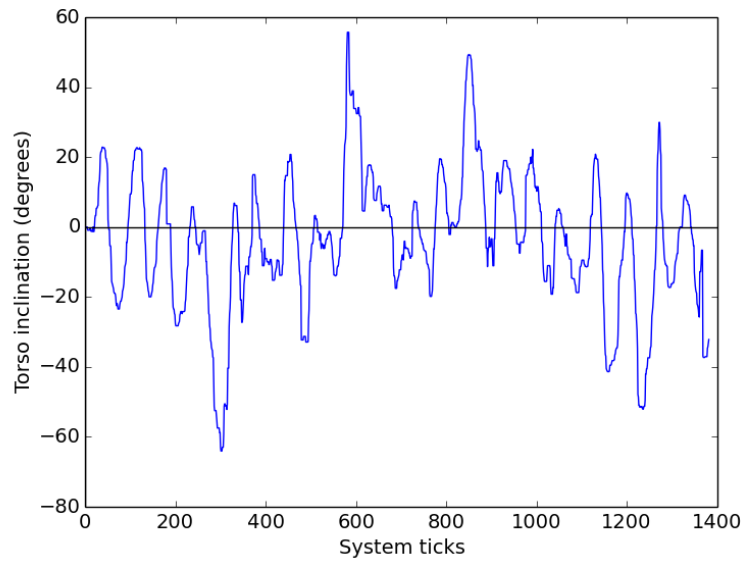


Figure B.73: Torso inclination during trial 2 with AOANN Fuz-28 controller using Let's Sway policy

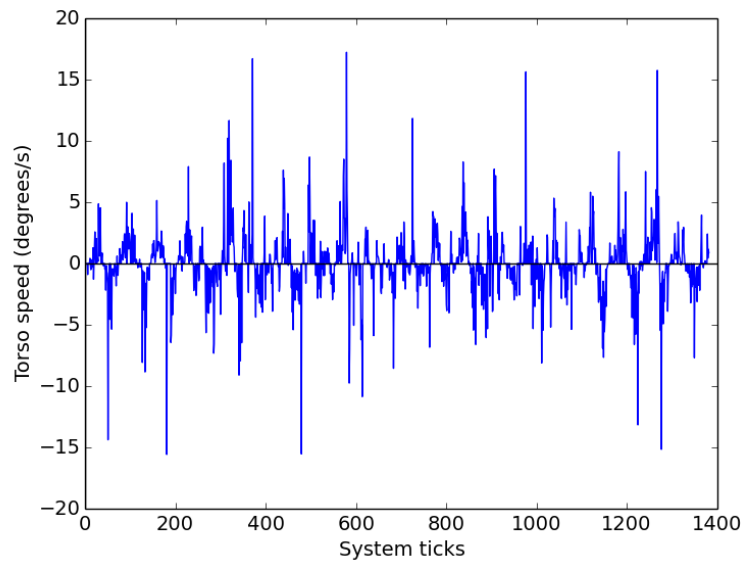


Figure B.74: Torso angular velocity during trial 2 with AOANN Fuz-28 controller using Let's Sway policy

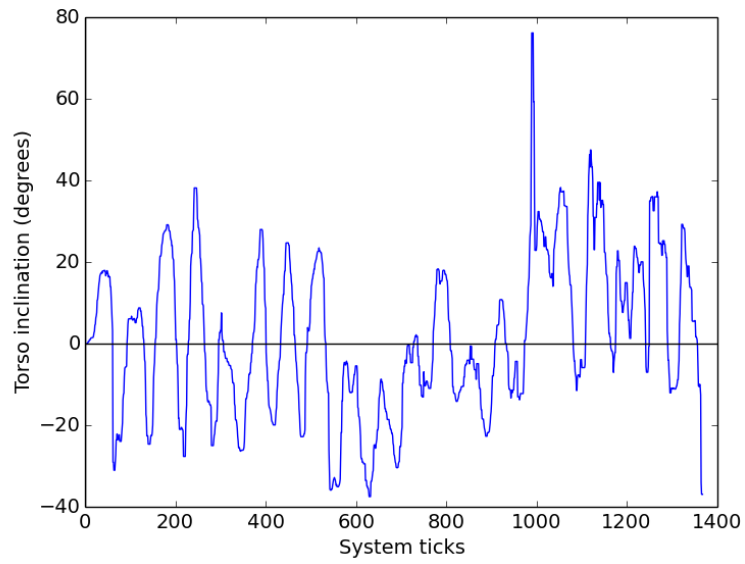


Figure B.75: Torso inclination during trial 3 with AOANN Fuz-28 controller using Let's Sway policy

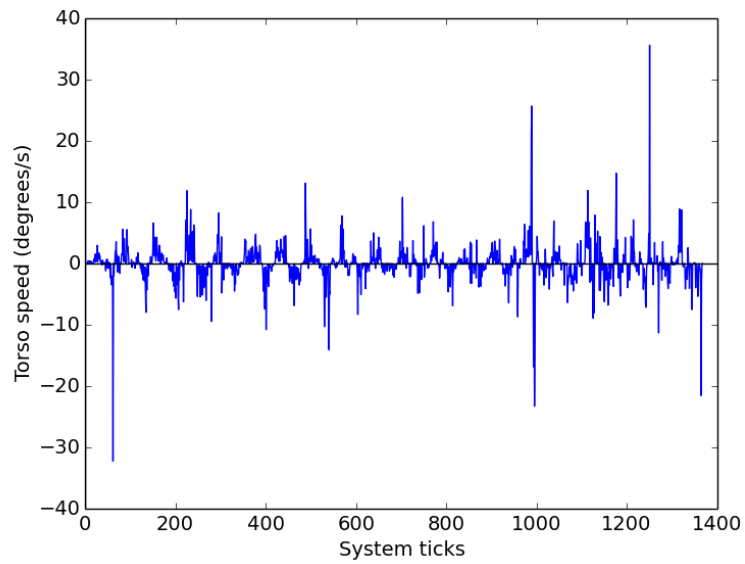


Figure B.76: Torso angular velocity during trial 3 with AOANN Fuz-28 controller using Let's Sway policy

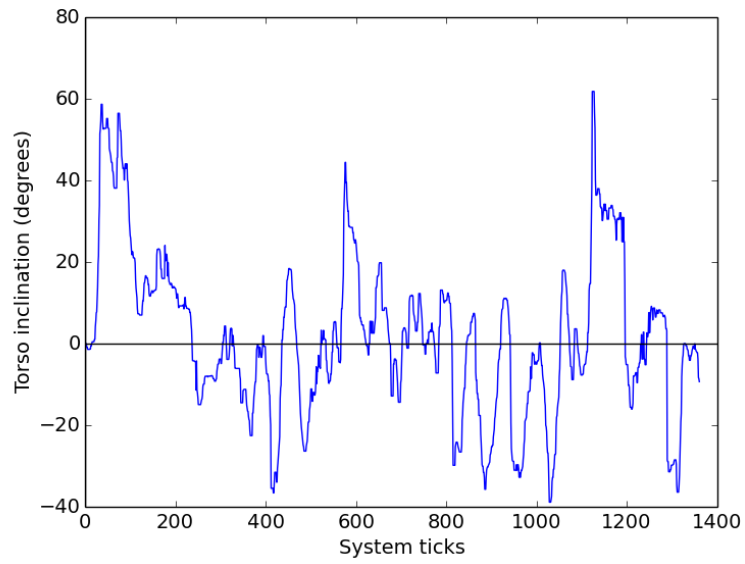


Figure B.77: Torso inclination during trial 4 with AOANN Fuz-28 controller using Let's Sway policy

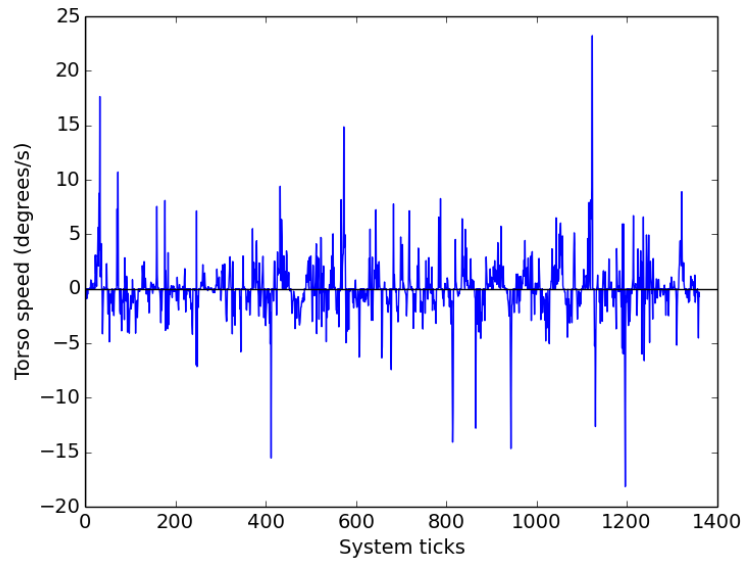


Figure B.78: Torso angular velocity during trial 4 with AOANN Fuz-28 controller using Let's Sway policy

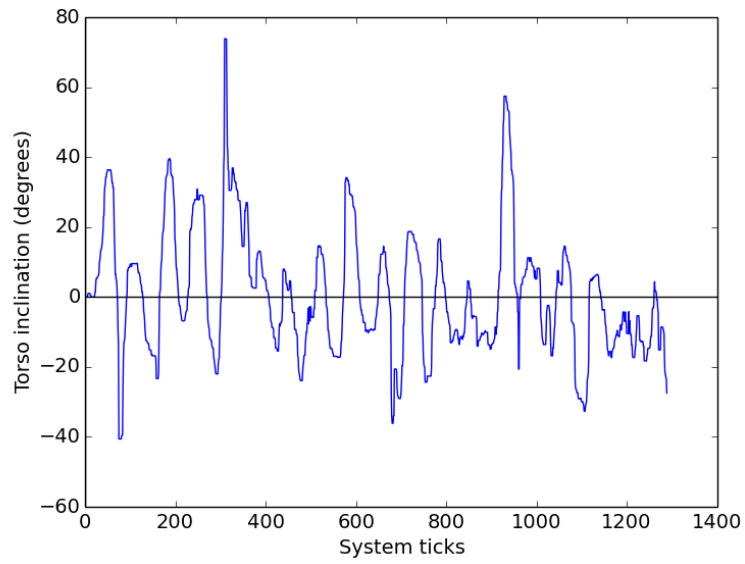


Figure B.79: Torso inclination during trial 5 with AOANN Fuz-28 controller using Let's Sway policy

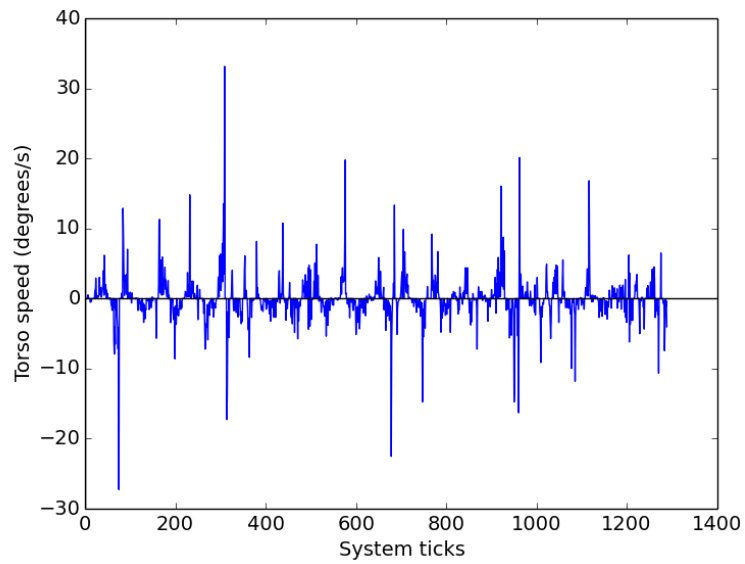


Figure B.80: Torso angular velocity during trial 5 with AOANN Fuz-28 controller using Let's Sway policy