INTELLIGENT MODEL-BASED CONTROL OF COMPLEX MULTI-LINK MECHANISMS

A thesis submitted to Cardiff University in the candidature for the degree of

Doctor of Philosophy

By

Hafizul Azizi Bin Ismail, B.Eng., M.Eng.

School of Engineering

Cardiff University

United Kingdom

DECEMBER 2016

ABSTRACT

Complex under-actuated multilink mechanism involves a system whose number of control inputs is smaller than the dimension of the configuration space. The ability to control such a system through the manipulation of its natural dynamics would allow for the design of more energy-efficient machines with the ability to achieve smooth motions similar to those found in the natural world. This research aims to understand the complex nature of the Robogymnast, a triple link underactuated pendulum built at Cardiff University with the purpose of studying the behaviour of non-linear systems and understanding the challenges in developing its control system.

A mathematical model of the robot was derived from the Euler-Lagrange equations. The design of the control system was based on the discrete-time linear model around the downward position and a sampling time of 2.5 milliseconds.

Firstly, Invasive Weed Optimization (IWO) was used to optimize the swing-up motion of the robot by determining the optimum values of parameters that control the input signals of the Robogymnast's two motors. The values obtained from IWO were then applied to both simulation and experiment. The results showed that the swing-up motion of the Robogymnast from the stable downward position to the inverted configuration to be successfully achieved.

Secondly, due to the complex nature and nonlinearity of the Robogymnast, a novel approach of modelling the Robogymnast using a multi-layered Elman neural

network (ENN) was proposed. The ENN model was then tested with various inputs and its output were analysed. The results showed that the ENN model to be capable of providing a better representation of the actual system compared to the mathematical model.

Thirdly, IWO is used to investigate the optimum Q values of the Linear Quadratic Regulator (LQR) for inverted balance control of the Robogymnast. IWO was used to obtain the optimal Q values required by the LQR to maintain the Robogymnast in an upright configuration. Two fitness criteria were investigated: cost function J and settling time T. A controller was developed using values obtained from each fitness criteria. The results showed that LQRT performed faster but LQRJ was capable of stabilizing the Robogymnast from larger deflection angles.

Finally, fitness criteria J and T were used simultaneously to obtain the optimal Q values for the LQR. For this purpose, two multi-objective optimization methods based on the IWO, namely the Weighted Criteria Method IWO (WCMIWO) and the Fuzzy Logic IWO Hybrid (FLIWOH) were developed. Two LQR controllers were first developed using the parameters obtained from the two optimization methods. The same process was then repeated with disturbance applied to the Robogymnast states to develop another two LQR controllers. The response of the controllers was then tested in different scenarios using simulation and their performance was evaluated. The results showed that all four controllers were able to balance the Robogymnast with the fastest settling time achieved by WMCIWO with disturbance followed by in the ascending order: FLIWOH with disturbance, FLIWOH, and WCMIWO.

ACKNOWLEDGEMENTS

Alhamdullilah. I would like to thank Dr. Michael Packianather, Dr. Roger Grosvenor and Dr. Eldaw Eldhukhri for the countless supervision and help that they had offered from day one of my PhD until this thesis is successfully produced. I would also like to thank my colleagues Dr. Haider Kamil and Mr. Abdul Syafiq Abdull Sukor for all their comments and suggestions.

Also, special thanks to the Malaysian Government especially Majlis Amanah Rakyat (MARA) and German Malaysian Institute (GMI) for sponsoring my studies and family while I am on study leave.

I would like to give my deepest appreciation to my loving parents, Mr. Ismail Borhan and Mrs. Hasnah Nordin, my parents-in-law, Mr. Ahmad and Mrs. Rodziah Lebai Mat for all their sacrifices and prayers. I would also like to thank my sisters for making me believe that I can achieve my dreams.

To the Malaysian community of Cardiff, especially Dr. Awanis and Mr. Arbaq, thank you for making my stay in Cardiff a time to be cherished for the rest of my life.

Last but not least, I thank my beloved wife, Hafisoh for all her love and support and my son Aariz for brightening up my day with all his smiles and laughter.

DEDICATION

Thank you Allah for giving me the strength and opportunity to gain knowledge.

****

To

My Wife, Hafisoh

My Son, Aariz Rahil

My Parents, Ismail Borhan and Hasnah Nordin

My sisters, Jaja, Yan, Nora, Shira and Adik

My brother in-law Zamir

My nieces Odah, Nana and Rowa

Thank you for being my stars in the dark night.

DECLARATIONS AND STATEMENTS

DECLARATION

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ……..…… (Hafizul A. Ismail) Date…….…….

STATEMENT 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD).

Signed ……..…… (Hafizul A. Ismail) Date…….…….

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ……..…… (Hafizul A. Ismail) Date…….…….

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ……..…… (Hafizul A. Ismail) Date…….…….

CONTENTS

Abstract i
Acknowledgements iii
Dedication iv
Declarations and Statements v
Contents vi
LIST OF FIGURES x
LIST OF TABLES xiv
Abbreviations xvi
List of Symbols xix
Chapter 1 Introduction 1
1.1 Introduction 1
1.2 Motivation 1
1.3 Aim and Objectives 2
1.4 Methodology 3
1.5 Thesis outline 4
1.6 Publications 8
Chapter 2 Background Review 9
2.1 Introduction 9
2.2 Background 10
2.3 Complex multi-link mechanism 11
2.4 Swing-up control 12
2.5 Artificial neural network modelling 15
2.5.1 Types of neural network models 16
2.5.2 Learning Algorithm Categorization 22
2.6 Optimization Algorithms 23

2.7 Stochastic Optimization Methods 26
2.7.1 Particle Swarming Optimization 26
2.7.2 Ant Colony 27
2.7.3 Intelligent Water Drops 28
2.7.4 Bees Algorithm 29
2.7.5 Invasive weed optimisation 31
2.8 Fuzzy Logic 32
2.8.1 Fuzzy Sets 35
2.8.2 Fuzzy Rules 35
2.8.3 Fuzzy Inference 36
2.9 Multi-objective optimization 32
2.9.1 Types of multi-objective optimization 33
2.10 Upright balancing of a pendulum 39
2.11 Summary 41
Chapter 3 System Description and Mathematical Modelling 42
3.1 Introduction 42
3.2 System Description 43
3.3 Mathematical Model 51
3.4 Summary 60
Chapter 4 Swing-Up Control of the Triple Link Pendulum 61
4.1 Introduction 61
4.2 Swing-up control 62
4.3 Invasive Weed Optimization Algorithm 65
4.4 Tuning the swing–up control parameters using IWO 68
4.5 Results 72
4.5.1 IWO Results 72
4.5.2 Simulation Results 74
4.5.3 Experiment Results 76
4.6 Discussion and Conclusion 79

4.7 Summary 82
Chapter 5 Artificial Neural Network Modelling of the Robogymnast 83
5.1 Introduction 83
5.2 Elman Neural Networks 86
5.3 Activation Function 89
5.4 Back-Propagation Algorithm 90
5.5 Training the ENN Model 93
5.6 Results 96
5.7 Discussion and Conclusion 102
5.8 Summary 104
Chapter 6 Upright Balancing of the Robogymnast 106
6.1 Introduction 106
6.2 Model of the Robogymnast in the upright position 108
6.3 Linear Quadratic Regulator 108
6.4 Application of IWO in LQR controller design 110
6.5 LQR controller designed using cost function (J) as the fitness criterion. 112
6.5.1 Simulation results of LQR designed using IWO with cost function J as the fitness criterion 116
6.6 LQR controller designed using time (T_{st}) as the fitness criterion 125
6.6.1 Simulation results of LQR designed using IWO with T_{st} as the fitness criterion 127
6.7 Discussion and conclusion 133
6.8 Summary 136
Chapter 7 Multi-Objective Weed Optimization of the LQR Controller 138
7.1 Introduction 138
7.2 Weighted Criteria Method Invasive Weed Optimization 140
7.2.1 Simulation results of LQR designed using WCMIWO 144
7.3 Fuzzy Logic Invasive Weed Optimization Hybrid 149
7.3.1 Simulation results of LQR designed using FLIWOH 158

7.4 Training with disturbance 163
7.4.1 WCMIWO training with disturbance results 164
7.4.2 FLIWOH training with disturbance results 166
7.5 Discussion and conclusion 169
7.6 Summary 175
Chapter 8 Conclusion, Contribution and Future Work 176
8.1 Conclusions 176
8.2 Contributions 178
8.3 Future work 179
Appendix 181
References 203

LIST OF FIGURES

Figure 1. 1: Thesis structure and research objectives 7

Figure 2.1 : The Acrobot (Spong 1994) 13

Figure 2.2: Feed Forward Neural Network Diagram of Robogymnast 18

Figure 2.3: Jordan Network (Wysocki and Lawrynczuk 2015)..... 19

Figure 2.4: A two-dimensional cellular neural network (Chua and Yang 1988b)..... 21

Figure 2.5: Flowchart of Basic Bees Algorithm (Ahmad 2012)..... 31

Figure 2.6: A set of Fuzzy rules..... 36

Figure 3.1: Robogymnast System Diagram 45

Figure 3.2: Robogymnast (a) Front view (b) Side view 46

Figure 3.3: Block diagram representation of Robogymnast overall system 47

Figure 3.4: Block diagram representation of the experimental apparatus 48

Figure 3.5: Circuit diagram of 1st order filter in series with operational amplifier..... 50

Figure 3.6: Circuit diagram of the power amplifier..... 51

Figure 3.7: Schematic representation of Robogymnast 52

Figure 4.1: Robogymnast in mid-swing..... 62

Figure 4.2: IWO Flow Chart (Madivada Hymavathi and Rao 2012) 67

Figure 4.3: Pseudo-code for IWO..... 70

Figure 4.4: Flowchart for Invasive Weed Optimization Algorithm..... 71

Figure 4.5: Simulated angular position Θ_1 for Set 1..... 74

Figure 4.6: Simulated angular position Θ_1 for Set 2..... 74

Figure 4.7: Simulated angular position Θ_1 for Set 3..... 75

Figure 4.8: Simulated angular position Θ_1 for Set 4 75

Figure 4.9: Measured angular position Θ_1 for Set 1. 77

Figure 4.10: Measured angular position Θ_1 for Set 2 77

Figure 4.11: Measured angular position Θ_1 for Set 3 78

Figure 4.12: Measured angular position Θ_1 for Set 4 78

Figure 4.13: Flowchart of Robogymnast swing-up sequence 81

Figure 5.1: Elman Neural Network Diagram of Robogymnast 88

Figure 5.2: Backpropagation Configuration 91

Figure 5.3: Flow Chart of Back Propagation Training for Robogymnast 95

Figure 5.4: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6924, $\Delta\alpha_2$= 0.1966, $\Delta\delta$=0.051129 for (a)Experimental; (b) Mathematical Model; (c) ENN Model 97

Figure 5.5: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6616, $\Delta\alpha_2$= 0.1699, $\Delta\delta$=0.05512 for (a)Experimental; (b) Mathematical Model; (c) ENN Model 98

Figure 5.6: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6635, $\Delta\alpha_2$= 0.1827, $\Delta\delta$=0.05935 for (a)Experimental; (b) Mathematical Model; (c) ENN Model 99

Figure 5. 7: Output of Θ_1 with its input voltages u_1 and u_2 103

Figure 6.1: Configurations of Robogymnast (a) θ_1=-3°; θ_2=-3°; θ_3=-3°, (b) θ_1=-3°; θ_2=3°; θ_3=-3°, (c) θ_1=3°; θ_2=3°; θ_3=3° 118

Figure 6.2: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 119

Figure 6.3: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3° 120

Figure 6.4: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3° 121

Figure 6.5: Disturbance to Link 1 122

Figure 6.6: Disturbance to Link 2 123

Figure 6.7: Disturbance to Link 3 124

Figure 6.8: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 128

Figure 6.9: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°..... 129

Figure 6.10: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3° 130

Figure 6.11: Disturbance to Link 1 131

Figure 6.12: Disturbance to Link 2 132

Figure 6.13: Disturbance to Link 3 133

Figure 7.1: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 145

Figure 7.2: Simulation of LQR with initial deflection of θ_1=-3°; θ_2= 3°; θ_2 = -3° 145

Figure 7.3: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3° 146

Figure 7.4: Disturbance to Link 1 147

Figure 7.5: Disturbance to Link 2 148

Figure 7.6: Disturbance to Link 3 148

Figure 7.7: The main flowchart of the FLIWOH Algorithm 150

Figure 7.8: The flowchart of the Fuzzy Logic Algorithm..... 151

Figure 7.9: Fuzzy Logic Membership Functions 155

Figure 7.10: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 159

Figure 7.11: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°..... 159

Figure 7.12: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3° 160

Figure 7.13: Disturbance to Link 1 161

Figure 7.14: Disturbance to Link 2 162

Figure 7.15: Disturbance to Link 3 162

Figure 7.16: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 165

Figure 7.17: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°..... 165

Figure 7.18: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 166

Figure 7. 19: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3..... 167

Figure 7. 20: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°..... 168

Figure 7. 21: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3° 168

LIST OF TABLES

Table 2. 1: Comparison between Mamdani FIS and Sugeno FIS (Hamam and Georganas 2008) 38

Table 3.1: Parameters of the Robogymnast 56

Table 3.2: Motor parameters 56

Table 4.1 : IWO Parameter 69

Table 4.2: IWO Results 73

Table 4.3: Simulation Results vs. Experimental Results 80

Table 5. 1: Parameters of Back-Propagation Training of ENN Model 94

Table 5.2: Error Comparison 101

Table 6. 1: IWO parameters with J as the fitness criterion 113

Table 6.2: IWO Results using the cost function J as the fitness criterion 115

Table 6.3: IWO Results using Time (T_{st}) as the fitness criterion 126

Table 6. 4: Comparison of the performance of the LQRJ and LQRT in different initial angular configurations. 135

Table 7.1: WCMIWO parameters 142

Table 7.2: WCMIWO Results 143

Table 7.3: FLIWOH parameters 153

Table 7.4: Fuzzy Logic Rule 154

Table 7.5: FLIWOH Results 157

Table 7. 6: Comparison of the performance of controllers in different initial angular *(small angles)* configurations. 170

Table 7. 7: Comparison of the performance of controllers in different initial angular *(large angles)* configurations. 171

Table 7.8: Ranking of Performance 172

Table 7.9: Comparison with LQRJ and LQRT……174

ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
Acrobot	Acrobat Robot
AD/DA	Analogue to Digital/Digital to Analogue
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BA	Bees Algorithm
BP	Back Propagation
C	Capacitor
CMAC	Cerebellar model articulation control
CNN	Cellular Neural Network
DC	Direct Current
DE	Differential Evolution
DLQR	Discrete Linear Quadratic Regulator
DOF	Degree of Freedom
ENN	Elman Neural Network
ENNM	Elman Neural Network Model
FIS	Fuzzy Inference Systems
FL	Feedback Linearisable
FLIWOH	Fuzzy Logic Invasive Weed Optimization Hybrid
FNN	Feed Forward Neural Network
FO	Fractional Order
GA	Genetic Algorithm
GMDH	Group Method of Data Handling

ICA	Imperialist Competitive Exclusion
Inf	Infinite
IWD	Intelligent water drops
IWO	Invasive Weed Optimization
LQR	Linear Quadratic Regulator
LQRJ	Linear Quadratic Regulator (J)
LQRT	Linear Quadratic Regulator (T)
LVQ	Learning vector quantization
MA	Mean Absolute
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
MKP	Multiple Knapsack Problem
MLP	Multi Layer Perceptron
MM	Mathematical Model
MOIWO	Multi Objective Invasive Weed Optimization
MOO	Multi Objective Optimization
NBI	Normal Boundary Intersection
NC	Normal Constraint
PC	Personal Computer
Pot	Potentiometer
PSO	Particle Swarming Optimization
R	Resistor
RMS	Root Mean Square Error
RNN	Recurrent Neural Network
Robogymnast	Robot gymnast
STLC	Small Time Local Controllable

TRMS	Twin Rotor Multi-Input Multi-Output System
T-S	Takagi-Sugeno
TSP	Travelling Salesman Problem
WCMIWO	Weighted Criteria Method Invasive Weed Optimization
ZMP	Zero Moment Point

LIST OF SYMBOLS

$\emptyset_i$	Angle of ith sinusoidal control signal
$\emptyset_i$	Angle of ith sinusoidal control signal
A1,A2,η,$\Delta\alpha_1$,$\Delta\delta_1$, $\Delta\alpha_2$,$\Delta\delta_2$	Constant values
a_i	Centre of gravity of ith link
a_i	Centre of gravity of ith link
AVG	Average
C_i	Viscous friction coefficient of the ith joint
Cp_i	Viscous friction coefficient of ith motor/gearbox reflected at the out-put shaft of the gearbox dissipation energy
D	Dissipation energy
e	Error
F	Gain matrix
g	Acceleration of gravity (9.81m/s2)
g	Acceleration of gravity (9.81m/s2)
G	Good
G_i	Static gain of ith motor/gearbox
I_i	Moment of inertia of ith link around its centre of gravity
Ip_i	Moment of inertia of ith motor/ gearbox reflected at the output shaft of the gearbox
iter	Iteration
$iter_{max}$	Number of maximum iterations
J	Performance index

J	Cost function
JT	Combined fitness criterion of J and T_{st}
K	Kinetic energy
k_i	Ratio of the ith gearbox
k_i	Ratio of the ith gearbox
l_i	Length of the ith link
m_i	Mass of the ith link
MVal	Membership value of Quality
n	Nonlinear modulation index
NG	No Good
P	Potential energy
p	Plant
p_{init}	Number of randomly chosen values from the solution space
Q ,R	Weight matrices
q_i	Relative angle of ith link
q_i	Relative angle of ith link
s	Seconds
Si	ith seed
S_{Max}	Maximum population of solutions
S_{Min}	Minimum population of solutions
T_i	Generalised torque at angle θi
Tm_i	Torque generated by the ith motor
Ts	Sampling time
T_{st}	Settling time
u_i	Input voltage from computer to ith driving motor

W	Colony
W_{cx}	Context layer weight matrix
W_{ho}	Output layer weight matrix
W_{ih}	Input layer weight matrix
W_J	Weightage of J
W_{Tst}	Weightage of T_{st}
y	Output signal from Robogymnast to computer
y_i	ith output
α_i, δ_i	ith Variable values
η	Learning rate
θ_i	Angle of ith link from the vertical line
μ	Mean
σ_{final}	Final standard deviation used for spatial distribution of plants
$\sigma_{initial}$	Standard deviation used for spatial distribution of plants

CHAPTER 1

Introduction

1.1 Introduction

Advancements in robotic technology have accelerated at an unbelievable rate in the past decade. From the powerful manipulators on the production floors to the first walking android, robotics has taken large leaps in terms of efficiency and flexibility. However, robots today still move far too conservatively, due to attempts to obtain full control authority of each limb at all times. This leads to inefficient and jerky motions. Humans and animals move much more aggressively by routinely executing motions which involve a loss of instantaneous control authority (Tedrake 2009). Controlling nonlinear systems without complete control authority requires methods that can reason about and exploit the natural dynamics of our machines.

1.2 Motivation

Complex (under-actuated) multi-link structures provide useful test beds for evaluation, optimization and comparison of different control techniques. They are inherently nonlinear and present challenging modelling and control problems that

are commonly found in many real-life applications. In particular, the study of such systems will enable researchers to develop solutions that are aimed at addressing motion problems encountered by disabled and/or injured people experiencing limb impairment.

1.3 Aim and Objectives

This research is a study on improving and analysing the implementation of intelligent model-based control methods on complex multi-link mechanisms, focusing on the integration of artificial intelligence and knowledge-based systems. It aims to implement modelling, simulation and control of under-actuated mechanisms to gain in-depth understanding of modern control techniques and their applications and optimization for the benefit of industry and society.

The above aim will be accomplished by fulfilling the following research objectives:

1. Develop a swing-up method for the Robogymnast through the manipulation of its motors' control signals.
2. Apply a swarm-based optimisation technique to optimise the parameters of the control signals.
3. Apply the optimised parameters for the swing-up on the real system.
4. Develop and apply an alternative model of the system using neural networks.
5. Analyse and validate the alternative model.

6. Develop and simulate controllers to balance the Robogymnast in an inverted configuration.
7. Select the optimised parameters of the controllers using swarm-based optimisation techniques.
8. Develop modified swarm-based multi-objective optimisation techniques to optimise the selection of the controller parameters.
9. Validate the proposed controllers through simulation.

1.4 Methodology

To achieve the above objectives, the following methodology was adopted:

- Review of previous work: an extensive survey was performed of the state of the art in order to identify the main requirement for the control of and problems encountered in the control of complex multi-link mechanisms. This investigation also covers the study of control methods to be implemented and analysed.
- The Euler-Lagrange approach is used to derive a mathematical model and dynamic equations of Robogymnast at the stable equilibrium point.
- The swing-up control simulation is achieved using the MATLAB® software and its associated toolboxes. The parameters are optimised using the IWO and the findings are implemented on the real systems via a C++ program environment.

- The candidates for the alternative model are investigated and evaluated. The developed model is validated by comparing it with the mathematical model and data from experiments.
- The problem of balancing the Robogymnast in an inverted configuration is investigated. The LQR controller is developed and its parameters are selected using the conventional Invasive Weed Optimization (IWO) and the modified Multi Objective Optimization (MOO) IWO. The controllers are then validated via MATLAB® simulations.

1.5 Thesis outline

The remainder of the thesis is organized as follows:

Chapter 2 reviews the background literature related to the field of complex multi-link mechanisms. Problems related to complex multi-link mechanisms such as swing-up and balancing control are discussed. The chapter also provide reviews on artificial neural networks, optimisation algorithms and fuzzy logic.

Chapter 3 presents the system description and mathematical modelling of the Robogymnast. The overall system is discussed and illustrated using figures and diagrams. The mathematical model of the Robogymnast in the downward position is derived in detail.

Chapter 4 describes the design of the Robogymnast's swing-up controller. A technique to control the swing-up motion by manipulating the control signals' parameters is proposed. The parameters of the control signals are then optimised using the IWO. Results from simulations and experiments are presented.

Chapter 5 elaborates the design of a neural network model of the Robogymnast. The purpose of this work is to provide a more accurate representation of the Robogymnast system. An analysis of the neural network model is performed and compared with that of the mathematical model.

Chapter 6 presents the application of IWO in the design of LQR controllers for the upright balancing of the Robogymnast. This chapter investigates the use of the cost function (J) and settling time (T_{st}) as the fitness criteria of the IWO. Two controllers were designed based on the two fitness criteria and their performance is evaluated.

Chapter 7 introduces two MOO methods (WCMIWO and FLIWOH) based on the IWO. The two MOO methods are used in the selection of the LQR controller parameters. External disturbances were applied to the MOO process with the objective of creating more robust controllers. Four controllers are proposed in this chapter and their performances are evaluated.

Chapter 8 lists the contributions of this research, summarises the conclusions reached and provides suggestions for further research.

Figure 1.1 shows an outline of the thesis structure and the research objective which it addresses.

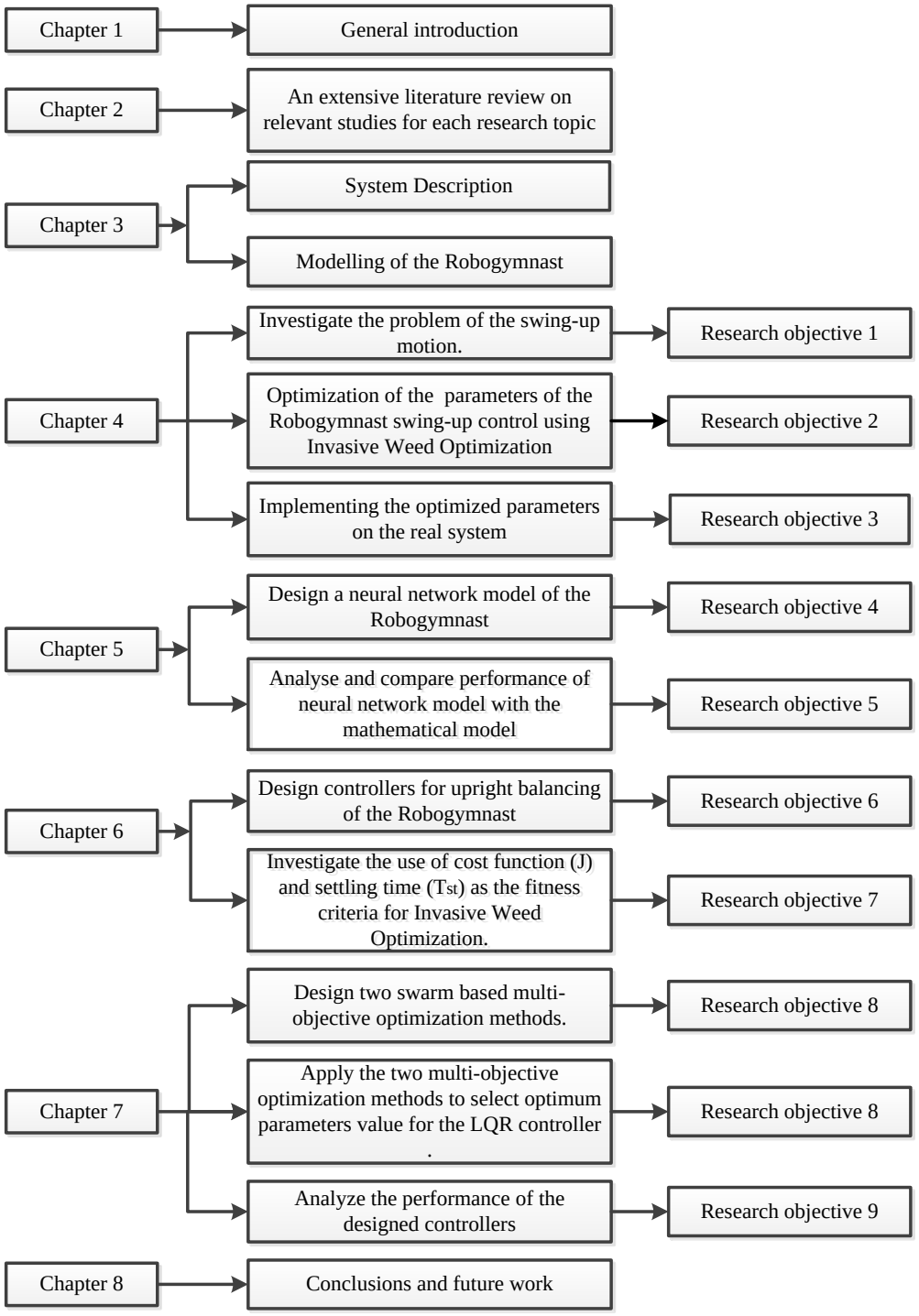


Figure 1. 1: Thesis structure and research objectives

1.6 Publications

Journal (Accepted for publication):

Ismail, H.A., Packianather, M.S., Grosvenor, R.I., Multi-Objective Invasive Weed Optimization of the LQR Controller. In International Journal of Automation and Computing (IJAC).

Conferences: -

Ismail, H.A., Packianather, M.S., Grosvenor, R.I. and Eldhukri, E.E., 2015, November. The application of IWO in LQR controller design for the Robogymnast. In *SAI Intelligent Systems Conference (IntelliSys), 2015* (pp. 274-279). IEEE.

Ismail, H.A., Eldhukhri, E.E. and Packianather, M.S., 2014, July. Invasive weed optimization of swing-up control parameters for robot gymnast. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*(pp. 88-93). IEEE.

CHAPTER 2

Background Review

2.1 Introduction

In this chapter, the literature associated with complex multi-link mechanisms and solutions to the inverted pendulum problem is reviewed and analysed. The swing-up control and upright balancing problems are discussed and previous studies are examined. Most researchers opt to combine the study of the swing-up control with the upright balancing problem, while others approach the two problems separately.

The remainder of this chapter is organized as follows. Section 2.2 sets out the background to the study. Section 2.3 discusses complex multi-link mechanisms and why the study of such mechanisms is important. Section 2.4 discusses the first problem, which is the swing-up control of the inverted pendulum. This is followed by Section 2.5, which presents a review of artificial neural network modelling and its components. Section 2.6 describes optimization algorithms and their categories. Section 2.7 discusses several popular stochastic optimization algorithms. An introduction to fuzzy logic is presented in Section 2.8. Multi-objective optimization

is discussed in Section 2.9 and Section 2.10 provides a review of the upright balancing problem. Finally, a summary of the chapter is given in Section 2.11.

2.2 Background

Many studies have investigated the dynamics of inverted pendulums (Chiu 2010). Most control experiments use the rail-cart structure when studying the inverted pendulum (Jaiwat and Ohtsuka 2014; Anderson 1989; Xiong and Wan 2010). However, swinging pendulums such as acrobot and Robogymnast are quickly gaining popularity due to their applications in walking robots (DeJong and Spong 1994; Liu and Yamaura 2011).

The study of inverted pendulums generally consists of two parts: swing-up motion and upright balancing. Pendulums such as the acrobot are difficult to control, due to being a four-dimensional, highly nonlinear, under-actuated control problem (Dracopoulos and Nichols 2012) .

Various approaches have been taken to solve this problem. Classical control methods have been employed with varying success according to (Jose et al. 2015), while other studies employ intelligent control methods (Liu et al. 2008; Kawada et al. 2004). The literature review discusses all the components applied in the control and modelling of the Robogymnast.

2.3 Complex multi-link mechanism

A mechanical linkage is an assembly of bodies connected together to manage forces and movement. The movement of a body, or link, is studied using geometry, so the link is considered to be rigid (Manickavelan et al. 2014). The connections between links are modelled as providing ideal movement, pure rotation or sliding, for example, and are called joints. A linkage modelled as a network of rigid links and ideal joints is called a kinematic chain (Manickavelan et al. 2014). A mechanism is defined as a connected system of links ensuring transmission and transformation of mechanical motion (Kolovsky et al. 2012). Complex multi-link mechanisms are mechanisms with a number of linkages that is less than the number of degrees of freedom (DOF) (Uicker et al. 2003). Complex multi-link mechanisms are also known as under-actuated mechanisms. Under-actuated mechanisms bring many advantages in energy, material and space consumption of numerous applications (Cheng et al. 2013). In the field of academia, under-actuated mechanisms provide a useful test bed for the evaluation and comparison of different control techniques (Eldukhri and Pham 2010). Most under-actuated systems are not full-state feedback linearisable (FL) around any equilibrium point, and some are not even small-time local controllable (STLC). This makes the control of such systems a challenging problem (Lai et al. 2011). Control of such mechanisms forms one of the recent major research topics in control engineering and robotics (Takashiro and Yoshihiko

1997). A popular example of an under-actuated mechanism is the inverted pendulum.

A pendulum is a body suspended from a fixed point so as to swing freely to and fro under the action of gravity, and is commonly used to regulate movements (Anon 2000). However, because of their nonlinear nature, inverted pendulums have maintained their usefulness and are now used to illustrate many of the ideas emerging in the field of nonlinear control, such as swinging up and catching the pendulum. Pendulums are also excellently suited to illustrate hybrid systems and the control of chaotic systems (Åström and Furuta 2000). Numerous studies have been conducted on non-linear control using the double and triple link pendulum as a test bench.

2.4 Swing-up control

The swing-up control of a pendulum is a popular topic that has been extensively researched. The main problem is to determine and track a valid swing-up trajectory that accomplishes the boundary restrictions and minimises the effort made by the actuator on the base (Rubi et al. 2002). The acrobot, as seen in Figure 2.1, so named because of its similarity to a human acrobat, is an under-actuated unstable robot that is useful as a test bed for studying the theory and application of non-linear control (Brown and Passino 1997). The acrobot is a planar, two-link robot with an actuator

at the elbow (joint 2) but no actuator at the shoulder (joint 1). The task of the acrobot is to swing up, in the minimum time, from the initial stable pendant position to the inverted unstable position and to remain balanced in that position (Dracopoulos and Nichols 2015).

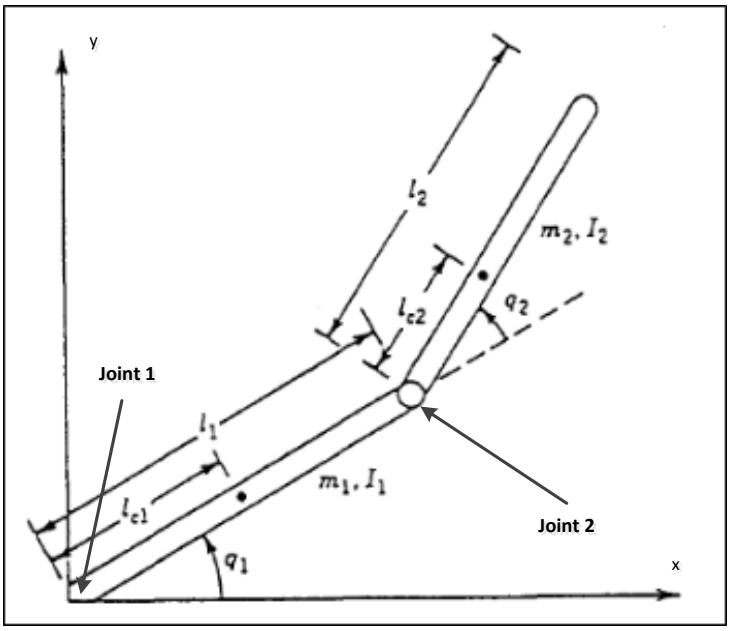


Figure 2.1: The Acrobot adapted from (Spong 1994)

Research conducted by Spong successfully produced a swing-up controller for the acrobot based on the method of partial feedback linearization of "unstable zero dynamics" (Spong, 1994, 1995). Spong also designed a controller to capture and balance the acrobot at the top of its swing using a linear quadratic regulator (Spong 1994). In his research, Spong noted the fact that the entire swing-up motion is produced by the natural response of an automated system. This shows that by

making the motion of the second link dependent on the motion of the first link, the entire system can be made autonomous. Brown and Passino (1997) continued Spong's research and developed intelligent controllers for swing-up and balancing of the acrobot. They developed and compared the performance of classical, fuzzy and adaptive fuzzy controllers for balancing the acrobot in its inverted equilibrium region. Two genetic algorithms were then used for tuning the balancing and swing-up controllers. Their results show that the swing-up motion of the acrobot can be further optimized by tuning the control parameters through the use of Genetic Algorithms (GA). Awrejcewicz et al. (2012) presented a swing-up controller using a bang-bang control torque exerted about the suspension point.

A skilled gymnast pointed out that in achieving an effective swing, the shoulders play a more important role than do the hips. Thus, to mimic gymnastic routines more realistically and to understand the control mechanism inside the routine better, one should model the gymnast on a high bar at least as a 3-DOF underactuated robot: that is, the gymnast's shoulder should be modelled as an actuated joint as well as the hips (Xin and Kaneda 2007b). Eldukhri and Pham used a new method for swing-up control of a triple link pendulum. This method does not require control signals to be derived in terms of measurements of variables such as speed and acceleration, but rather by manipulating the frequencies and amplitudes of oscillating functions applied to two motors mounted at the robot's shoulder and hip joints (Eldukhri and Pham 2010). The Bees Algorithm, a population-based

search algorithm that emulates the foraging behaviour of honeybees, was later used to optimize the swing-up control of the robot. This was done by independently manipulating the amplitude and the frequencies of the control signals (Kamil et al. 2012). This technique was reported to be successful in obtaining a smoother and faster swing-up motion.

2.5 Artificial neural network modelling

Artificial Neural Networks (ANN) are computational models of the brain (Pham and Liu 1995). A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modelling tools. ANNs have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that (Luma and Yaseen 2013):

- Information processing occurs at many simple elements called neurons that are fundamental to the operation of ANNs.
- Signals are passed between neurons over connection links.

- Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.
- Each neuron applies an action function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.
- The units in a network are organized into a given topology by a set of connections, or weights, typically shown as lines in a network diagram.

ANN are characterized by:

- Architecture: its pattern of connections between the neurons.
- Training algorithm: its method of determining the weights on the connections.
- Activation function.

ANNs are often classified as single layer or multilayer. In determining the number of layers, the input units are not conventionally counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnecting links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information.

2.5.1 Types of neural network models

- Feedfoward Neural Networks

In a Feedfoward Neural Network (FNN), all signals flow in one direction only, i.e. from lower layers (input) to upper layers (output). Feedforward networks consist of at least three layers of neurons: an input layer, a hidden layer and an output layer. The nodes of the input layer are passive, meaning that they do not modify the data. They receive a single value on their input and duplicate the value to their multiple outputs. The nodes of the hidden and output layers are active, meaning that they modify the data that they receive (Smith 1999). Examples of feedfoward networks are multi-layer perceptron (MLP), the learning vector quantization (LVQ) network, the cerebellar model articulation control (CMAC) network and the group-method of data handling (GMDH) network (Pham and Liu 1995). Figure 2.2 illustrates the FFN of the Robogymnast.

- Recurrent Neural Networks

Recurrent neural networks were an important focus of research and development during the 1990s (Medsker and Jain 2001). In a Recurrent Neural Network (RNN), signals from neurons in upper layers are fed back to either its own layer or to neurons in lower layers via an extra layer called a context layer. Examples of RNNs include the Hopfield network, the Elman network and the Jordan network (Figure 2.3). RNNs have dynamic memories where their outputs at a given instant reflect the current input as well as previous inputs and outputs (Pham and Liu 1995).

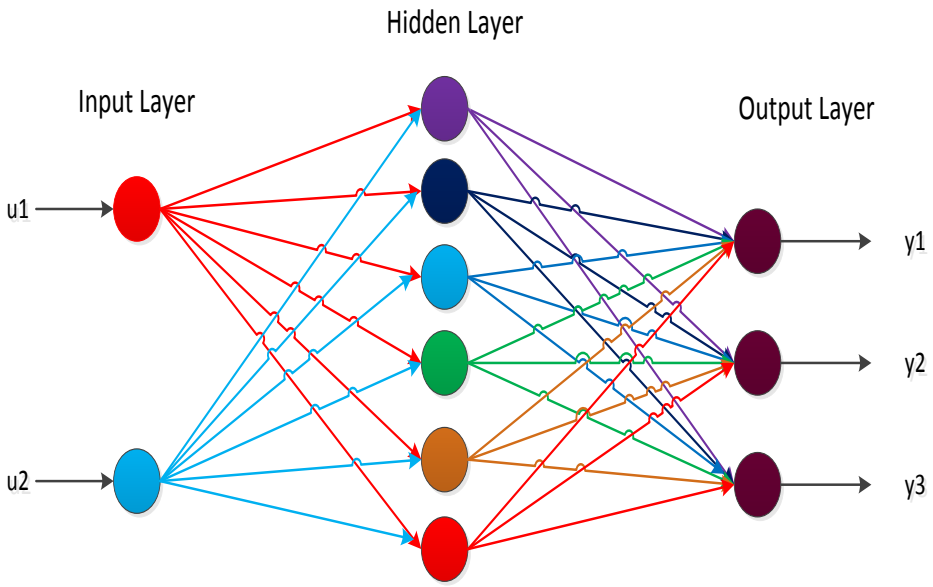


Figure 2.2: Feed-forward Neural Network Diagram of Robogymnast

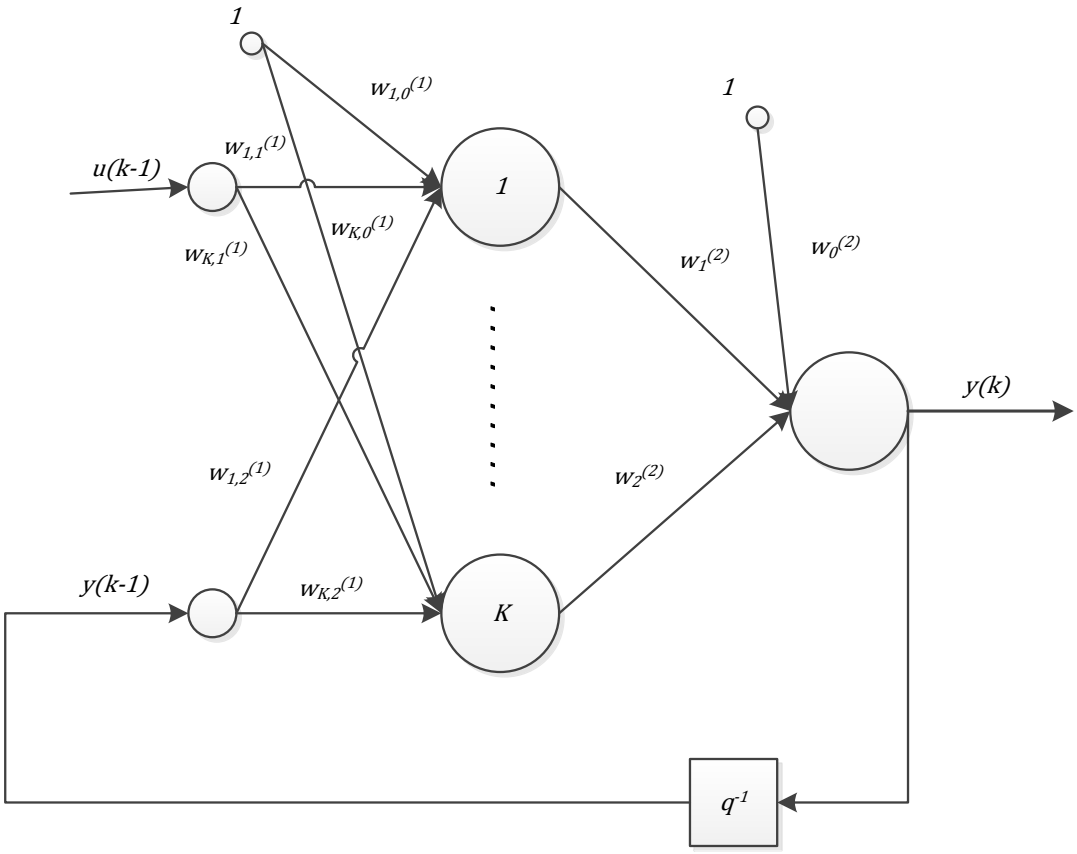


Figure 2.3: Jordan Network (Wysocki and Lawryczuk 2015)

- Cellular Neural Networks

A Cellular Neural Network (CNN) is an artificial neural network consisting of separate neurons or cells. It consists of simple analogue circuits (cells) arranged in a matrix (Namba and Zhang 2006) as seen in Figure 2.4. Each cell is made up of a linear capacitor, a non-linear voltage-controlled current source and a few resistive linear elements. The structure of cellular networks is similar to that found in cellular automata: i.e. any cell in a cellular network is connected only to its neighbouring cells (Chua and Yang 1988a). The adjacent cells can interact directly with each other. Cells not directly connected together may affect each other indirectly because of the propagation effects of the continuous-time dynamics of CNNs (Chua and Yang 1988b). Due to their computation efficiency, CNNs have been applied to various fields, such as image recognition (Namba and Zhang 2006) and estimation (Habtie et al. 2015).

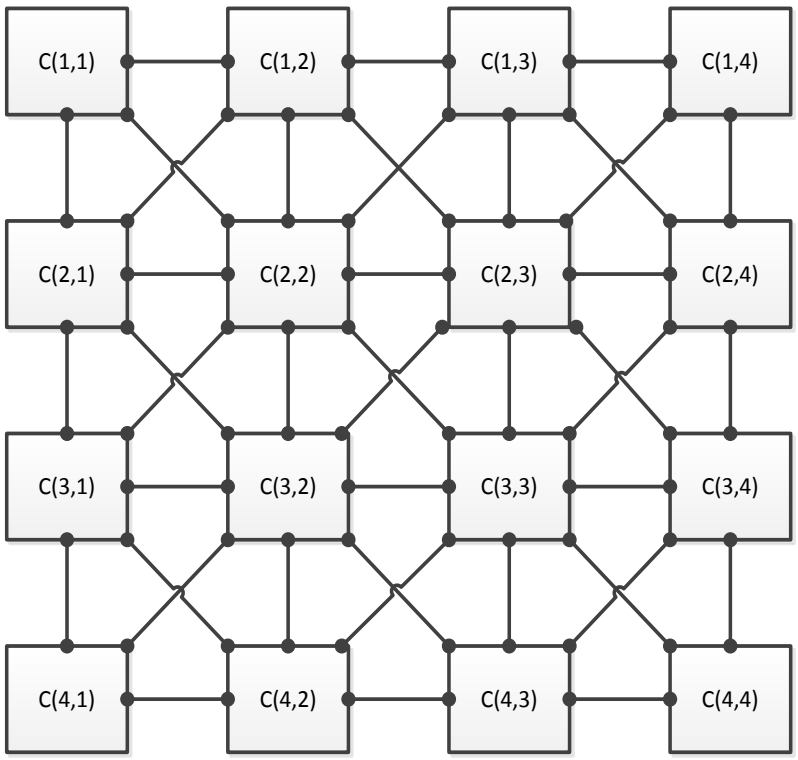


Figure 2.4: A two-dimensional cellular neural network (Chua and Yang 1988b).

2.5.2 Learning Algorithm Categorization

In all of the neural paradigms, the application of an ANN involves two phases: the learning phase and the recall phase. In the learning phase, the ANN is trained through the adaptation of its weights until it has learned its tasks, while the recall phase is used to solve the tasks.

There are three types of learning algorithm (Pham and Liu 1995):

- Supervised learning

  A supervised learning algorithm adjusts the strengths or weights of the inter-neuron connections according to the difference between the desired and actual network outputs corresponding to a given input.

- Unsupervised learning

  In unsupervised learning, the ANN is trained without teaching signals or targets. It is only supplied with examples of the input patterns that it will eventually solve.

- Reinforcement learning

  Reinforcement learning is a special case of supervised learning that employs a critic only to evaluate the goodness of the neural network output corresponding to a given input.

2.6 Optimization Algorithms

Optimization in the current context, means determining the best course of action amongst the different alternatives available in a decision-making problem. It can be regarded as a process of finding the optimal value of a function under a given set of circumstances, often called ‘constraints’ (Mohan and Deep 2009). An optimization algorithm is a procedure which is executed iteratively by comparing various solutions until an optimum or a satisfactory solution is found. With the advent of computers, optimization has become a part of computer-aided design activities.

Each optimization problem consists of the following basic ingredients (Engelbrecht 2005):

- An objective function which represents the quantity to be optimized.
- A set of unknowns or variables which affects the value of the objective function.
- A set of constraints that restricts the values that can be assigned to the unknowns.

The goal of an optimization method is then to assign values from the allowed domain to the unknowns, such that the objective function is optimized and the constraints are satisfied.

Two distinct types of optimization technique are widely used today, namely deterministic optimization and stochastic optimization (Heyman and Sobel 2003). The optimization techniques are classed by the type of algorithm implemented.

- Deterministic Optimization

Deterministic optimization techniques use specific rules for moving one solution to another. It embodies algorithms which rely heavily on linear algebra because they are commonly based on the computation of the gradient of the response variable. Deterministic optimization techniques are faster compared to stochastic optimization because they require a lower number of evaluations of the response variable to reach the solution. However, deterministic optimization algorithms look for a stationary point in the response variable: thus, the optimal solution eventually found could be a local optimum and not the global optimum. Deterministic algorithms are also intrinsically single objective (Cavazzuti 2013).

- Stochastic Optimization

Stochastic optimization techniques are of the same nature as probabilistic translation rules. These optimization techniques are more suitable for problems where the relation between the variables and the outputs is unknown. Stochastic optimization falls within the spectrum of the general-purpose type of approximation search techniques (Godfrey and Babu 2013).

There are two classes of stochastic optimization:

i. Local Search

In local search, a predefined solution is maintained and its neighbours are explored to find better quality solutions.

ii. Population-based Search

In population-based search, the single current solution is replaced by a population or collection of different current solutions. Members of this population are first selected to be current candidates and then changes are made to these current candidates' solutions to produce new candidate solutions.

For the Robogymnast swing-up optimisation problem, the stochastic optimization population-based search technique is selected due to its random nature and flexibility, which better suits the characteristic of the problem, such as the unknown relationship between the variable and the output.

2.7 Stochastic Optimization Methods

2.7.1 Particle Swarm Optimization

Particle swarm optimization (PSO), introduced by Eberhart and Kennedy (1995), is based on a social-psychological model of social influence and social learning (De Oca et al. 2006). In PSO, a number of simple entities (particles) are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best-fitness location with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved (Poli et al. 2007). The collective behaviour that emerges is that of discovering optimal regions of a high dimensional search space following the main principle of swarm intelligence (Engelbrecht 2005):

- Proximity principle
- Quality principle
- Principle of diverse response
- Principle of stability
- Principle of adaptability

The PSO pseudo-code is as shown below (Bharti and Singh 2015):

Step 1: Take the training data.

Step 2: Initialize the particles' population with their position and velocity parameters.

Step 3: Evaluate individual particle by calculating the fitness value: if fitness value > Pbest, then update current value as Pbest.

Step 4: Select the particle which has the best fitness value among all the particles.

Step 5: Calculate particle velocity and position according to equations 3 and 4.

Step 6: Continue until either the minimum error is not attained or up to the maximum iterations.

2.7.2 Ant Colony

Ant colony optimization (ACO) takes its inspiration from the foraging behaviour of some ant species. These ants deposit pheromones on the ground in order to mark some favourable path that should be followed by other members of the colony. Ant colony optimization exploits a similar mechanism for solving optimization problems (Dorigo and Birattari 2010). In ACO, a number of 'artificial ants' build solutions to the considered optimization problem at hand and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants.

ACO algorithms are based on the following ideas (Parpinelli et al. 2002):

- Each path followed by an ant is associated with a candidate solution for a given problem.
- When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem.
- When an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone have a greater probability of being chosen by the ant.

2.7.3 Intelligent Water Drops

The Intelligent Water Drops (IWD) algorithm is a swarm-based optimization algorithm which mimics the dynamics of river systems and the actions of water drops in the rivers (Duan et al. 2008). The IWD is a population-based constructive optimisation algorithm that may be used for maximization or minimization problems. The IWD has been used for the travelling salesman problem (TSP) and the multiple knapsack problem (MKP) with promising results (Shah-Hosseini 2009b). The IWDs are created with two main properties: velocity and soil. The IWD begins its trip with an initial velocity and zero soil. From its current location to its next location, the IWD's velocity is increased by an amount that is non-linearly proportional to the inverse of the soil between the two locations. An IWD

needs a mechanism to select the path to its next location or step (Shah-Hosseini 2009a). In this mechanism, the IWD prefers paths with low soils to paths with high soils. Therefore, it can be said that soil is the source material of information such that the environment and water drops both have memories of soil (Shah-Hosseini 2009b). The algorithm of the IWD is as follows:

1. Representation of the graph, which establishes the number of nodes of the problem that the water drop will visit and creates a route.
2. Establish the number of iterations.
3. Representation of static parameters, number of drops, initial velocity.
4. Representation of dynamic parameters, the soil and velocity would change every time the drop moves across the nodes established.
5. The condition is set: Did the drop visit all the nodes? If the answers is no, we go to number four.
6. Save the best result.

2.7.4 Bees Algorithm

The Bees Algorithm (BA) is an optimization algorithm inspired by the natural foraging behaviour of honey bees to find the optimal solution (Pham et al. 2006). BA tries to model the natural foraging behaviour of honey bees. Honey bees use several mechanisms such as the waggle dance to optimally locate food sources and

to search for new ones. This makes them a good candidate for developing new algorithms for solving optimization problems (Özbakir et al. 2010).

The main steps of the BA are listed as follows (Darwish 2009) and the flowchart is shown in Figure 2.5:

1. Initialise the population with random solutions.
2. Evaluate the fitness of the population.
3. While (stopping criterion not met) // Forming new population.
4. Select sites for neighbourhood search.
5. Determine the patch size.
6. Recruit bees for selected sites and evaluate their fitness.
7. Select the representative bee from each patch.
8. Amend the Pareto optimal set.
9. Abandon sites without new information.
10. Assign remaining bees to search randomly and evaluate their fitness.
11. End While.

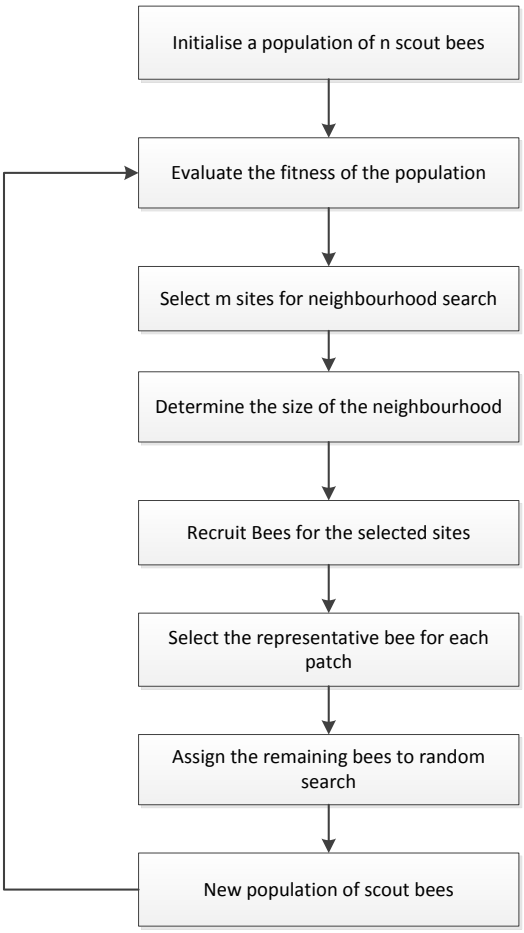


Figure 2.5: Flowchart of Basic Bees Algorithm (Ahmad 2012)

2.7.5 Invasive weed optimisation

Mehrabian and Lucas (2006) developed a new algorithm called Invasive Weed Optimisation (IWO). IWO is attractive due to its flexibility and robustness. A detailed explanation of IWO is provided in Chapter 4.

2.8 Multi-objective optimization

Multi-objective optimization (MOO) is the process of optimizing systematically and simultaneously a collection of objective functions. It originally grew out of three areas: economic equilibrium and welfare theories, game theory and pure mathematics (Marler and Arora 2004). MOO has been extensively researched and applied in various applications (Mohamed et al. 2009; Taherkhorsandi et al. 2015; Akbari et al. 2012). There is now increasing interest in MOO, as most engineering design problems involve multiple and often conflicting issues (Pham and Ghanbarzadeh 2007). Formally, MOO refers to simultaneous optimization (i.e., maximization and/or minimization) of two or more objective functions, which are often in conflict with one another. This optimization problem can be stated as follows (Rangaiah and Bonilla-Petriciolet 2013):

$$Optimize\ (f_1(x), f_2(x), \ldots \ldots . f_n(x)) \qquad (2.1)$$

Subject to

$$g_i(x) \leq 0\ i = 1,2 \ldots ., n_i$$

$$h_i(x) = 0\ i = 1,2 \ldots ., n_e \qquad (2.2)$$

$$x_l \leq x \leq x_u$$

where n is the number of objective functions to be simultaneously optimized, x is the vector of m decision variables (continuous and/or discontinuous) with lower

(x_l) and upper (x_u) bounds, n_i and n_e are the number of inequality (g) and equality (h) constraints, respectively. The feasible space, $\boldsymbol{F}$, is the set of vectors $\boldsymbol{x}$ that satisfy all the constraints and bounds in equation (2.2). In contrast to the single-objective optimization case, where the optimal solution is clearly defined, in MOO problems there is a whole set of trade-offs giving rise to numerous Pareto Optimal solutions (Parsopoulos and Vrahatis 2002).

2.8.1 Types of multi-objective optimization

The primary goal of MOO is to model a decision maker's preference: thus, MOO methods are categorized depending on how the decision-maker articulates these preferences. MOO can be divided into three major categories (Marler and Arora 2004):

a) Methods with *a priori* articulation of preferences - these allow the user to specify preferences which may be articulated in terms of goals or the relative importance of different objectives. Examples of these methods are:

- Weighted global criterion method
- Weighted sum method
- Lexicographic method
- Weighted min-max method
- Exponential weighted criterion

- Weighted product method
- Goal programming methods
- Bounded objective function method
- Physical programming

b) Methods for *a posteriori* articulation of preference - preferences are selected from a group of solutions through the use of an algorithm that is used to determine the representation of the generated Pareto optimal set. Examples of these methods are:
   - Physical programming
   - Normal boundary intersection (NBI) method
   - Normal constraint (NC) method

c) Methods with no articulation of preferences - these methods do not require any articulation of preferences. Examples of these methods are:
   - Global criterion methods
   - Nash arbitration and objective product method
   - Rao's method

2.9 Fuzzy Logic

The main idea behind Fuzzy systems is that truth values (in fuzzy logic) or membership values are indicated by a value in the range 0-1, with 0 representing absolute falsity and 1 representing absolute truth, in contrast to classical set theory, according to which each element either fully belongs to the set or is completely excluded from the set. In other words, classical set theory represents a limited case of the more general fuzzy set theory (Klir and Yuan 1995).

2.9.1 Fuzzy Sets

A fuzzy set is a class of objects with a continuum of grades of membership (Zadeh 1965). A fuzzy set $\tilde{A}$ on the given universe U is that, for any $u \epsilon U$, there is a corresponding real number $\mu_{\tilde{A}}(u) \epsilon [0,1]$ to u, where $\mu_{\tilde{A}}(u)$ is called the grade of membership of u belonging to $\tilde{A}$ (Li and Yen 1995). Fuzzy sets allow the elements in the set to have partial memberships within the range of 0-1. Thus, a fuzzy set is a generalization of an ordinary set by allowing a degree of membership for each element.

2.9.2 Fuzzy Rules

At the root of fuzzy set theory lies the idea of linguistic variables. A linguistic variable is a fuzzy variable. In fuzzy expert systems, linguistic variables are used in

fuzzy rules (Negnevitsky 2005). The fuzzy rules are normally expressed in a form that will allow the rules to be easily programmed. An example of a set of fuzzy rules is provided in Figure 2.6.

IF J is High and T is High, THEN Q is NG

IF J is AVG and T is High, THEN Q is NG

IF J is Low and T is High, THEN Q is NG

……..

……

IF J is Low and T is Low, THEN Q is G

Figure 2.6: A set of Fuzzy rules

2.9.3 Fuzzy Inference

The Inference Mechanism provides the mechanism for invoking or referring to the rule base such that the appropriate rules are fired. The steps of fuzzy reasoning performed by Fuzzy Inference Systems (FIS) are (Jang 1993):

1. Compare the input variables with the membership functions on the premise part to obtain the membership values (or compatibility measures) of each linguistic label. (This step is often called fuzzification).
2. Combine (through a specific T-norm operator, usually multiplication or min) the membership values on the premise part to get the firing strength

(weight) of each rule. Generate the qualified consequent (either fuzzy or crisp) of each rule depending on the firing strength.

3. Aggregate the qualified consequents to produce a crisp output. (This step is called defuzzification.)

There are two well established types of FIS, namely Mamdani-style inference and Sugeno-style inference (Kaur and Kaur 2012). Table 2.1 shows the comparison between the two FIS types.

Table 2. 1: Comparison between Mamdani FIS and Sugeno FIS (Hamam and Georganas 2008)

Mamdani	**Sugeno**
Output membership function	No output membership function
Output distribution	No output distribution, only 'resulting action': Mathematical combination of the rule strength and the output
Crisp result obtained through defuzzication of rules' consequent	No defuzzification: crisp result is obtained using weighted average of the rules' consequent
Non-continuous output surface	Continuous output surface
MISO and MIMO systems	Only MISO systems
Expressive power and interpretable rule consequents	Loss of interpretability
Less flexibility in system design	More flexibility in system design; more parameters in the output

2.10 Upright balancing of a pendulum

In the past few years, the single inverted pendulum model has been falsified as an explanatory approach for a quiet (standing) human stance. Double inverted pendulum models have recently proven to be inappropriate. Human topology, with three major leg joints, suggests a natural way to examine triple inverted pendulum models as an appropriate approach (Günther and Wagner 2015). The dynamics of balancing a pendulum at the unstable position can be employed in the applications of controlling walking robots, rocket thrusters, etc. (Huang and Huang 2000; McGrath et al. 2015; Kuo 2007; Yamamoto et al. 2015). This makes it a very popular experiment for educational purposes in modern control theory (Grossimon et al. 1996; Awtar et al. 2002; Rahimi et al. 2013; Boubaker 2013).

Most inverted pendulums are underactuated mechanical systems. This means that the angular acceleration and position of the pendulum cannot be controlled directly. Therefore, the techniques developed for fully actuated mechanical robot manipulators cannot be used to control inverted pendulums (Lozano et al. 2000).

Furuta et al. (1984) designed a controller for an inverted triple link pendulum using attitude control. By controlling the angles of the upper two arms around specified values, the pendulum can be stabilized inversely with the specified attitude.

Medrano-Cerda et al. (1995) proposed a robust computer control system for balancing and attitude control of double and triple inverted pendulums. The controller was designed using a blend of state-space and frequency domain methods. Experimental results indicate that the controller was successful in stabilizing the triple link pendulum, but the system's performance is greatly degraded due to backlash in the gearboxes.

Another method to balance the inverted pendulum was proposed by Park et al. (2004) using Q-learning. Two mode Q-learning was used to stabilize the Zero Moment Point (ZMP) of a biped robot in the standing posture. The controller was successful in stabilizing the biped robot in both simulations and experiments. The two mode Q-learning was more successfully in balancing the biped robot compared to conventional Q-learning, but took a longer time.

Based on the work done by Park et al. (2004), Raj and Kumar (2013) approached the inverted pendulum problem by using the Q-learning based reinforcement learning to balance a double inverted pendulum. They were able to prove through simulation that Q-learning is a simple but robust learning method. However, when implemented on the Robogymnast, the author discovered that the training process is far too difficult due to the numerous number of states a triple link pendulum has.

A paper published by Kamil et al. (2014) combined a Discrete-time Linear Quadratic Regulator (DLQR) controller and an integral control action to satisfy the required performance of the system. Kamil (2015) was able to prove through simulation results that the Robogymnast could be settled in the upright position for an acceptable amount of time (1 - 12 seconds).

2.11 Summary

Overviews of the various aspects that are applied in this thesis have been presented in this chapter. The literature review includes a discussion of the complex multi-link mechanism and the problems associated with its control. The literature also includes the elements used in designing its controller. In the next chapter, the system description of the Robogymnast is discussed in detail and its mathematical model is derived.

CHAPTER 3

System Description and Mathematical Modelling

3.1 Introduction

This chapter discusses the system description of the Robogymnast and the derivation of its mathematical model. The Robogymnast is a triple link pendulum and is classified as a complex multi-link mechanism. It can also be called an underactuated mechanism due to its lack of full actuation. This characteristic introduces challenges when designing a controller for the Robogymnast.

The Robogymnast has three degrees of freedom, where two of the degrees of freedom are actuated while one is unactuated. Due to its complex nature, the design of controllers for the Robogymnast requires computer-simulated tests to ensure their functionality before implementing them on the real system itself. To achieve this, a mathematical model of the Robogymnast had to be derived. The mathematical model is derived based on the Euler-Lagrange equations of motion (Spong 1994; Eldukhri and Pham 2010). The Euler-Lagrange equations describe the evolution of a mechanical system subject to holonomic constraints.

In order to determine the Euler-Lagrange equations in a specific situation, one has to form the Lagrangian of the system, which is the difference between the kinetic energy and the potential energy of the system (Spong et al. 2006).

Section 3.2 presents a description of the entire system and explanations of its individual components. Dimensions and other physical details of the system are given. Schematic diagrams and sketches of the Robogymnast are also provided in this section. In Section 3.3, the derivation of the mathematical model of the system is presented and discussed. The step-by-step derivation from the Euler-Lagrange equation to the state space model of the system is demonstrated in this section. A summary of the entire chapter is given in Section 3.4.

3.2 System Description

The triple link under-actuated mechanism (Robogymnast) is depicted in Figure 3.1 (Eldukhri and Pham 2010). The frame of the Robogymnast is made from 50mm diameter carbon fibre tubes weighing 0.213kg/m. Aluminium components are attached to the ends of each link to provide the structures for mounting sensors and actuators. Physical parameters of the system are designed according to the features of a human gymnast swinging on a freely rotating high bar with his hands firmly fixed to the bar. Each link represents a body part or a group of body parts on a human. Link 1 represents the arms (without elbows

and wrists). Link 2 represents the head and torso. Link 3 represents the legs (without knees and ankles). Joint 1 (hands) consists of a steel shaft mounted on ball bearings with a potentiometer mounted to measure the angle of rotation of link 1. Joints 2 (shoulders) and 3 (hip) are split into two sections. The first section is similar to joint 1 with a potentiometer to measure the relative angle of each link. The second section is the output shaft of the drive unit (DC motor/gearbox). The Robogymnast is controlled by a PC equipped with appropriate AD/DA converters. C++ programmes are used to transmit the input/output commands between the PC and Robogymnast (Kamil et al. 2012). Figure 3.2 shows the Robogymnast in its actual test environment. Figure 3.3 illustrates the overall system of the Robogymnast, while Figure 3.4 shows the setup of the system's experimental apparatus.

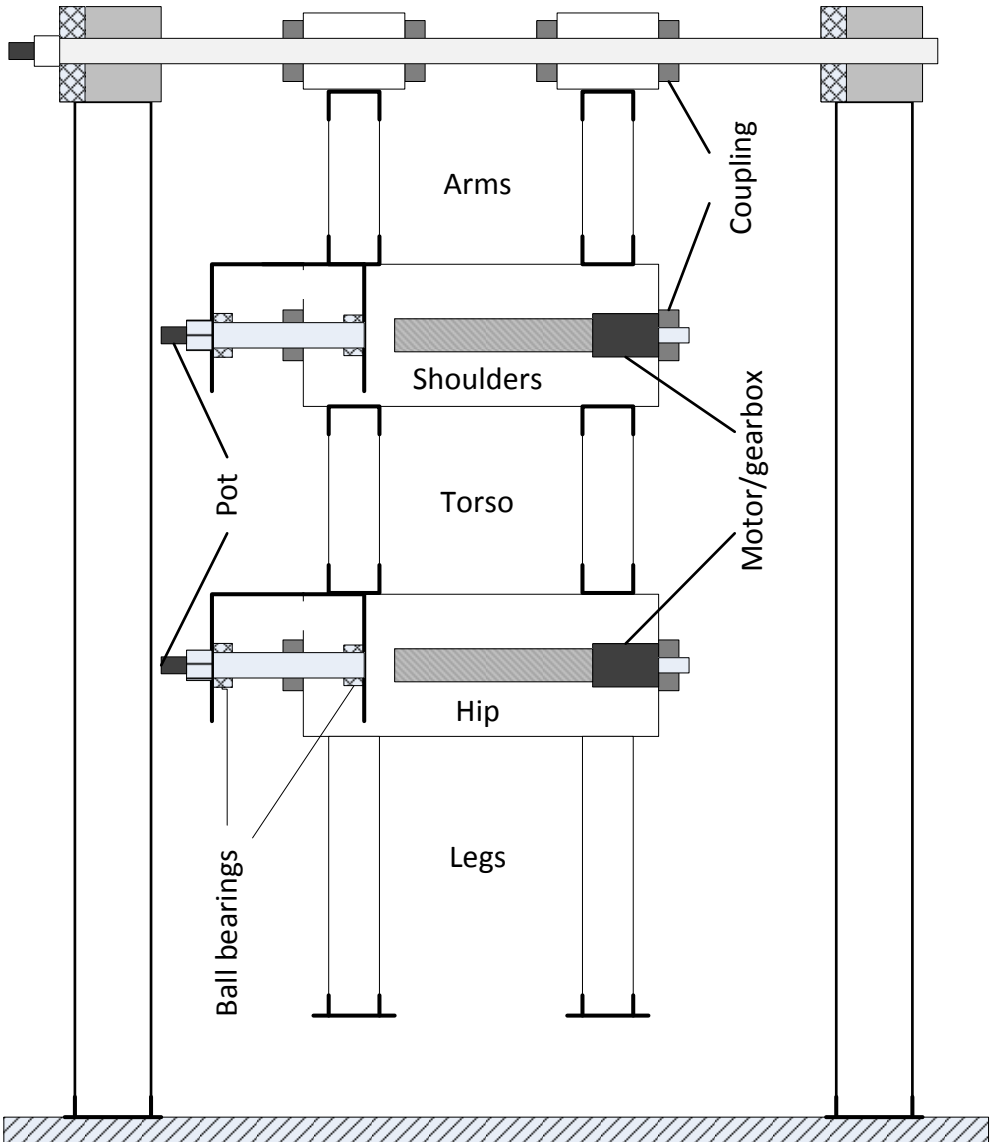
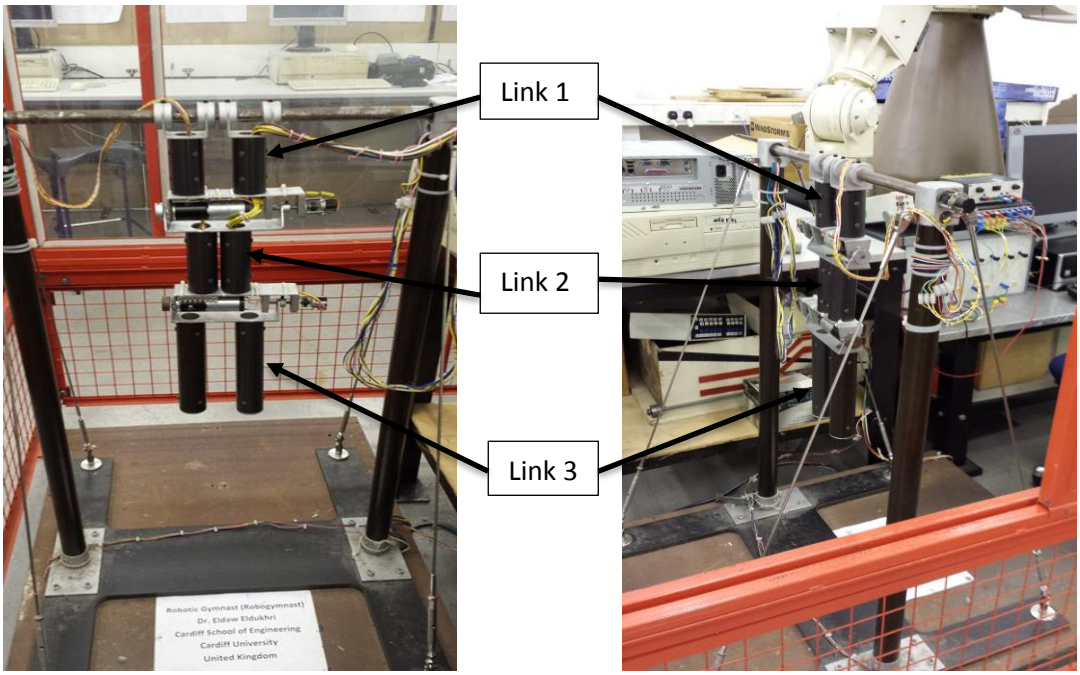


Figure 3.1: Robogymnast System Diagram (Kamil 2015)



(a) **(b)**

Figure 3.2: Robogymnast (a) Front view (b) Side view

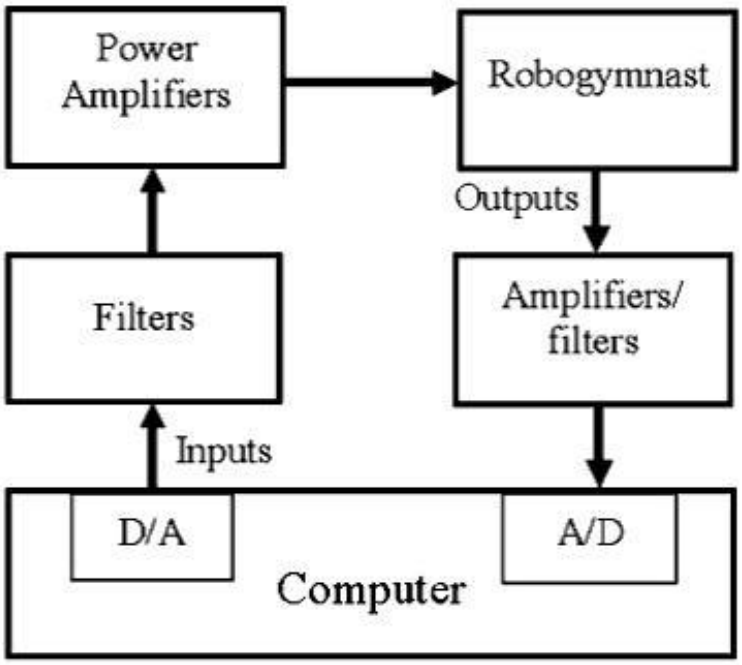


Figure 3.3: Block diagram representation of the Robogymnast system (Eldukhri and Pham 2010)

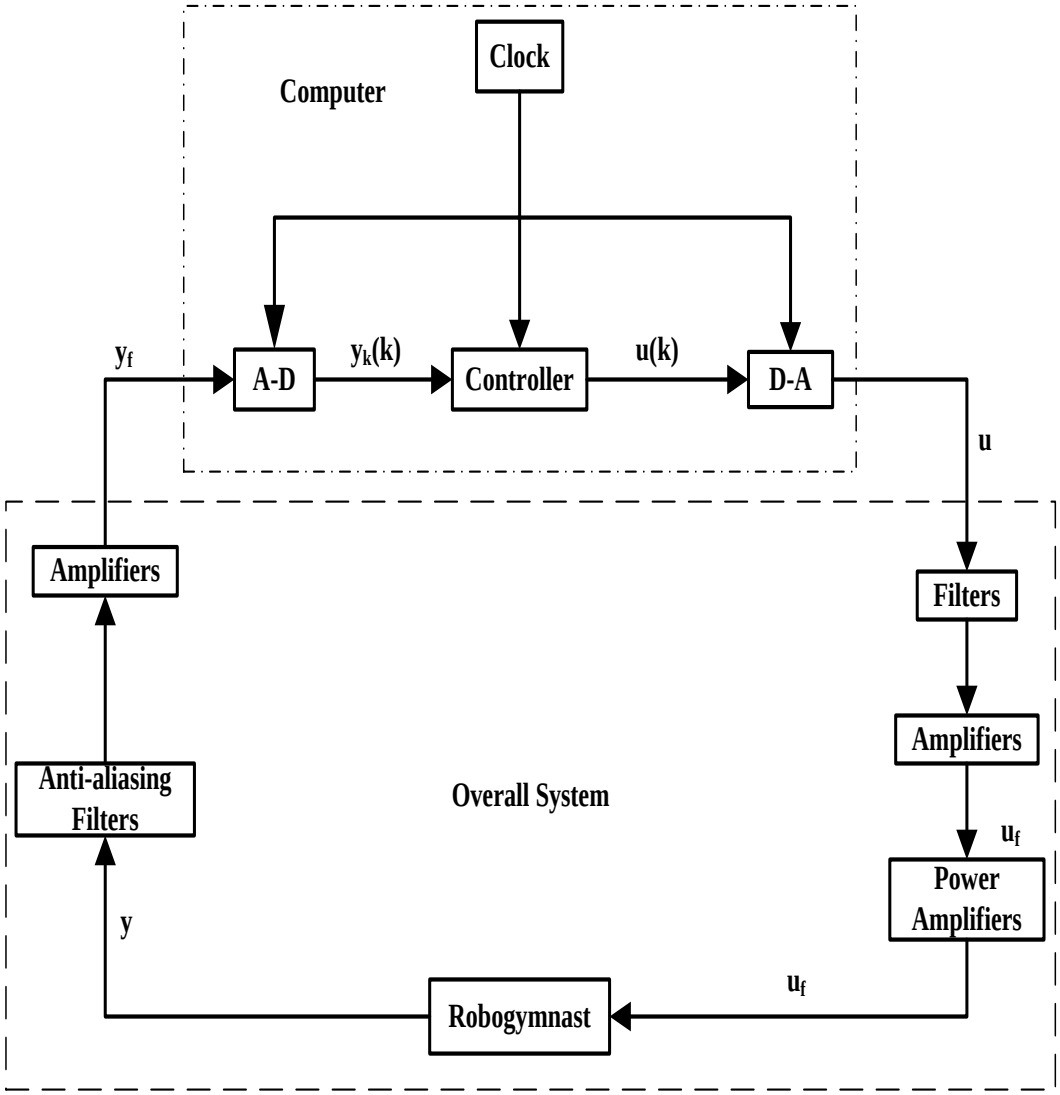


Figure 3.4: Block diagram representation of the experimental apparatus (Kamil 2015)

The Robogymnast's sensors (potentiometers) send analogue signals to the anti-aliasing filters. The analogue signals consist of two types of information. The first is the sensor readings, which are considered as controllable disturbance, and the second type are the uncontrollable disturbances. The uncontrollable disturbances are high frequency signals, while the sensor readings are low frequency signals (Kamil 2015). The anti-aliasing filter is tasked with reducing the effects of the disturbances. The filtered signals are then sent to a signal amplifier to be amplified. The amplified analogue signals are then sent to an ADLINK DAQ-2501 AD/DA convertor to be converted to digital signals. The ADLINK AD/DA convertor has a resolution of 12 bits for analogue input and 14 bits for analogue output. It has a conversion time of 1 microsecond and a settling time of less than 3 microseconds. From the AD/DA convertor, the signals are then sent to the controller. The controller is a computer (PC) that contains a C++ program. The controller program contains (Kamil 2015):

- A state feedback controller
- A discrete integrator
- A reduced order observer
- Offset adjustments in the control outputs.

- Scaling factors and sensor gains for the conversion of input signals from volts to radians.

The controller uses the data obtained from the input channels for control action calculations. It then sends the control action signals to an AD/DA to be converted to analogue signals. The control action signals go through filters and amplifiers before being sent to a power amplifier. The power amplifier amplifies the control action signals and sends them to the actuators (motor 1 and motor 2).

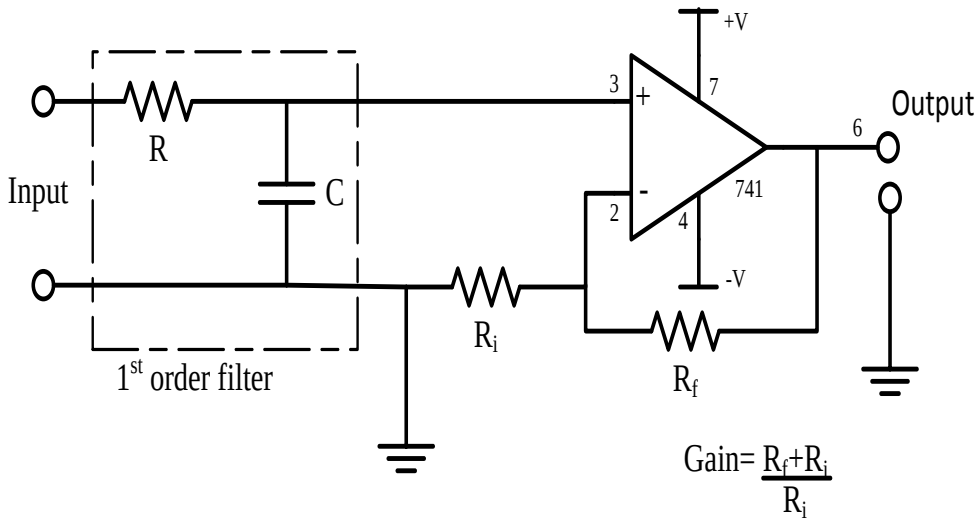


Figure 3. 5: Circuit diagram of 1st order filter in series with operational amplifier (Kamil 2015)

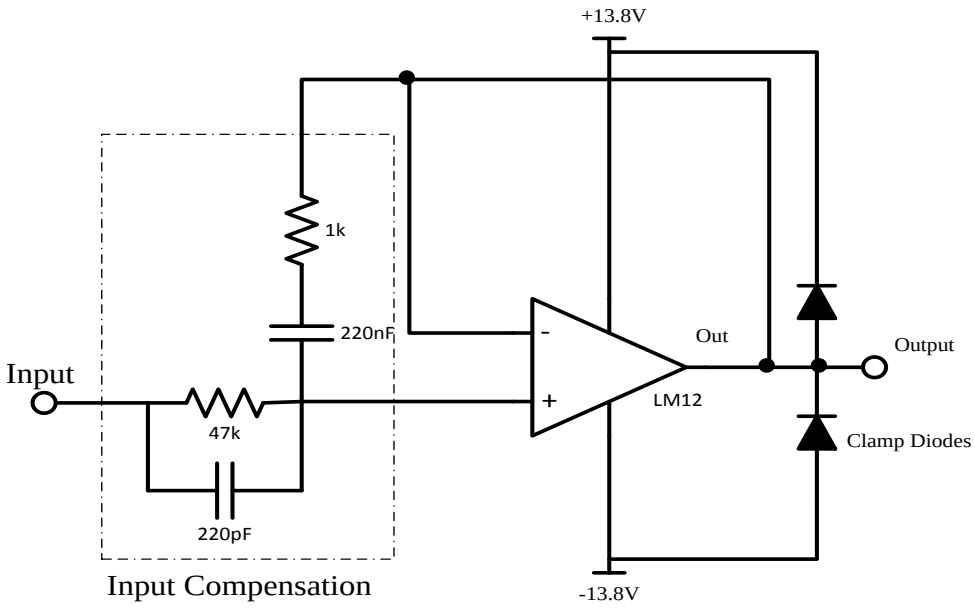


Figure 3.6: Circuit diagram of the power amplifier (Kamil 2015)

3.3 Mathematical Model

For modelling purposes, the Robogymnast is regarded as a triple link pendulum in a stable equilibrium configuration (Eldukhri and Pham 2010), as seen in Figure 3.7. The standard method for deriving dynamical equations of multi-rigid systems uses the Euler-Lagrange formula. This method involves only the derivatives of time, speed and position. The most important part of the Lagrangian equation is obtaining the kinetic and potential energy of the entire system (Gmiterko and Grossman 2009). In this section, the Robogymnast is

regarded as being in the downward (stable) position. The model of the Robogymnast in an inverted (unstable) position is discussed in Chapter 6.

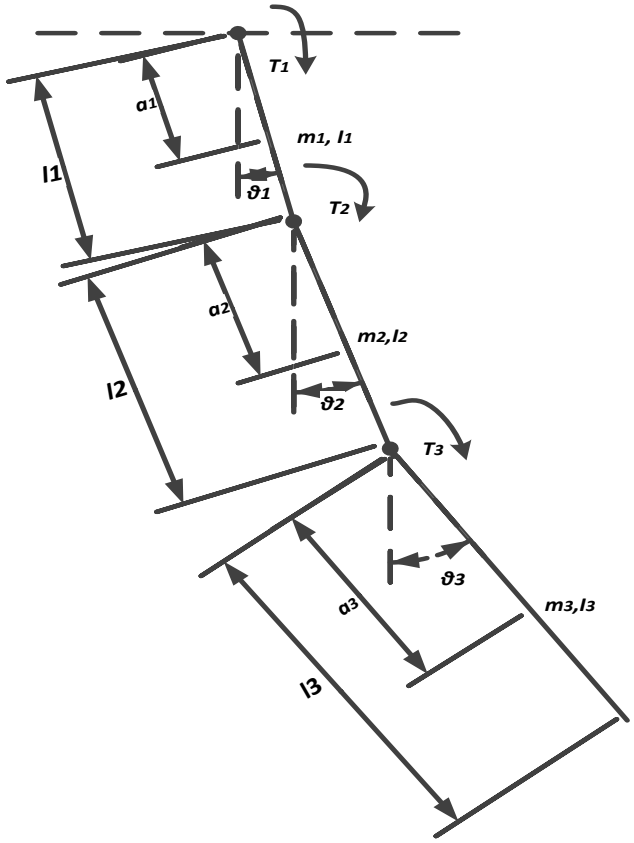


Figure 3.7: Schematic representation of Robogymnast

The mathematical model is derived using the Lagrange equation provide as equation (3.1)

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}_i}\right) - \frac{\partial K}{\partial \theta_i} + \frac{\partial D}{\partial \dot{\theta}_i} + \frac{\partial P}{\partial \theta_i} = T_i \quad i = 1,2,3 \tag{3.1}$$

where K is the kinetic energy, P is the potential energy and D is the dissipation energy. The angle of the *ith* link, measured with reference to the vertical line, is represented by θ_i , while T_i is the torque associated with it. The variables of the equation can be broken down to the form of equations (3.2) to (3.4):

$$K = \frac{1}{2}\sum_{i=1}^{3}\left\{ I_i\theta_i + m_i\left[\frac{d}{dt}\left(\sum_{k=i-3}^{i-1} l_k \sin\theta_k + a_i \sin\theta_i\right)\right]^2 + \left[\frac{d}{dt}\left(\sum_{k=i-3}^{i-1} l_k \cos\theta_k + a_i \cos\theta_i\right)\right]^2\right\} \tag{3.2}$$

$$P = \sum_{i=1}^{3} m_i g\left(a_i \cos\theta_i + \sum_{k=i-3}^{i-1} l_k \cos\theta_k\right) \tag{3.3}$$

$$D = \frac{1}{2}\sum_{i=1}^{3}\left(C_i\left(\dot{\theta}_i - \theta_0\right)^2\right) \tag{3.4}$$

Since joint 1 has no actuator, the torque applied to it is effected by motors on joint 2 (Tm_1) and joint 3 (Tm_2), where $T_1 = -Tm_1$, $T_2 = Tm_1 - Tm_2$, $T_3 = Tm_2$

The torque given by the motor is represented by

$$Tm_1 = G_1u_1 - Ip_1\left(\ddot{\theta}_2 - \ddot{\theta}_1\right) - Cp_1\left(\dot{\theta}_2 - \dot{\theta}_1\right) \tag{3.5}$$

$$Tm_2 = G_2u_2 - Ip_2\left(\ddot{\theta}_2 - \ddot{\theta}_1\right) - Cp_2(\dot{\theta}_2 - \dot{\theta}_1) \tag{3.6}$$

Solving equation (3.1) for θ_1 and linearizing around the point $\theta_1 = \theta_2 = \theta_3 \approx 0$.

$$L_1 = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}_1}\right) - \frac{\partial K}{\partial \theta_1} + \frac{\partial D}{\partial \dot{\theta}_1} + \frac{\partial P}{\partial \theta_1} - T_1$$

$$= \left[I_1 + m_1 {a_1}^2 + 2m_2 {l_1}^2 + 2m_3 {l_1}^2 + Ip_1\right]\ddot{\theta}_1 + [-C_2 - Cp_1]\dot{\theta}_2 + [-m_1 a_1 g - m_2 l_1 g - m_3 l_1 g]\theta_1 + G_1 u_1 \quad (3.7)$$

Solving equation (3.1) for θ_2

$$L_2 = [m_2 l_1 a_2 + m_3 l_1 l_2 - Ip_1]\ddot{\theta}_1 + \left[I_2 + m_2 {a_2}^2 + m_3 {l_2}^2 + Ip_1 + Ip_2\right]\ddot{\theta}_2 + [-Ip_2 + m_3 l_2 a_3]\ddot{\theta}_3 + [-C_2 - Cp_1]\dot{\theta}_1 + [C_2 + C_3 + Cp_1 + Cp_2]\dot{\theta}_2 + [-C_3 - Cp_2]\dot{\theta}_3 + [-m_2 a_2 - m_3 l_2]g\theta_2 - G_1 u_1 + G_2 u_2 \quad (3.8)$$

Solving equation (3.1) for θ_3

$$L_3 = [m_3 l_1 a_3 - Ip_2]\ddot{\theta}_1 + [m_3 l_2 a_3 + Ip_2]\ddot{\theta}_2 + [I_3 m_3 {a_3}^2]\ddot{\theta}_3 + [Cp_2]\dot{\theta}_1 + [-Cp_2 - C_3]\dot{\theta}_2 + [C_3 + Cp_2]\dot{\theta}_3 + [-m_3 a_3 g]\theta_3 - G_2 u_2 \quad (3.9)$$

After rearranging the equations (3.7), (3.8) and (3.9), equation (3.10) is obtained.

$$\widetilde{M}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\end{bmatrix} + \widetilde{N}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} + \tilde{P}\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{bmatrix} + \widetilde{H}\begin{bmatrix}u_1\\ u_2\end{bmatrix} = 0 \quad (3.10)$$

Where

$$\widetilde{M} = \begin{bmatrix} J_1 + Ip_1 & l_1M_2 - Ip_1 & l_1M_3 \\ l_1M_2 - Ip_1 & J_2 + Ip_1 + Ip_2 & l_2M_3 - Ip_2 \\ l_1M_3 & l_2M_3 - Ip_2 & J_3 + Ip_2 \end{bmatrix}$$

$$\widetilde{N} = \begin{bmatrix} C_1 + C_2 + Cp_1 & -C_2 - Cp_1 & 0 \\ -C_2 - Cp_1 & C_2 + C_3 + Cp_1 + Cp_2 & -C_3 - Cp_2 \\ 0 & -Cp_2 - C_3 & C_3 + Cp_2 \end{bmatrix}$$

$$\tilde{P} = \begin{bmatrix} -M_1g & 0 & 0 \\ 0 & -M_2g & 0 \\ 0 & 0 & -M_3g \end{bmatrix}, \quad \widetilde{H} = \begin{bmatrix} G_1 & 0 \\ -G_1 & G_2 \\ 0 & -G_2 \end{bmatrix}$$

and

$$M_1 = m_1a_1 + (m_2 + m_3)l_1, M_2 = m_2a_2 + m_3l_2,$$

$$M_3 = m_3a_3, J_1 = I_1 + m_1{a_1}^2 + (m_2 + m_3){l_1}^2$$

$$J_2 = I_2 + m_2{a_2}^2 + m_3{l_2}^2, J_3 = I_3 + m_3{a_3}^2$$

Parameter values given in Table 3.1 and Table 3.2 are then accordingly inserted into the equations.

Table 3.1: Parameters of the Robogymnast (Eldukhri and Pham 2010)

Link 1	Link 2	Link 3
l_1(m) = 0.155	l_2(m) = 0.180	l_3(m) = 0.242
a_1(m) = 0.0426	a_2(m) = 0.138	a_3(m) = 0.065
m_1(kg) = 2.625	m_2(kg) = 0.933	m_3(kg) = 0.375
I_1(kgm^2) = 0.014	I_2(kgm^2) = 0.018	I_3(kgm^2) = 0.002
C_1(Nms) = 0.0172	C_2(Nms) = 0.0272	C_3(Nms) = 0.035

Table 3.2: Motor parameters (Eldukhri and Pham 2010)

Motor 1	Motor 2
Ip_1(kgm^2) = 0.0358	Ip_2(kgm^2) = 0.0358
Cp_1(Nms) = 7.73	Cp_2(Nms) = 7.73
G_1(Nm/V) = 1.333	G_2(Nm/V) = 0.625
k_1 = 246:1	k_2 = 110.6:1

In order to arrange the equation in terms of relative angles (q) matrix W is introduced, where

$$W = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \quad \text{and} \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

thus

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 - \theta_1 \\ \theta_3 - \theta_2 \end{bmatrix} = W\theta$$

Equation (3.10) is then written as

$$\widetilde{M}W^{-1}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \widetilde{N}W^{-1}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} + \tilde{P}W^{-1}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \widetilde{H}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad (3.11)$$

Solving equation (3.11) for $[\ddot{q}_1 \quad \ddot{q}_2 \quad \ddot{q}_3]^T$ gives:

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} = -W\widetilde{M}^{-1}\widetilde{N}W^{-1}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} - W\widetilde{M}^{-1}\tilde{P}W^{-1}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} - W\widetilde{M}^{-1}\widetilde{H}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (3.12)$$

Assuming that $x = [q\ \dot{q}]^T$, the state-space modelling is then obtained from equation (3.12) as

$$\dot{x} = \begin{bmatrix} 0_3 & I_3 \\ -W\tilde{M}^{-1}\tilde{P}W^{-1} & -W\tilde{M}^{-1}\tilde{N}W^{-1} \end{bmatrix} x + \begin{bmatrix} 0_{3x2} \\ -W\tilde{M}^{-1}\tilde{H} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = Ax + Bu \tag{3.13}$$

$$y = [I_3 \quad 0_3]x = Cx \tag{3.14}$$

where

$$0_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 0_{3x2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and $\boldsymbol{y = q}$ is the output vector.

After substituting the parameters with the values given in Tables 3.1 and 3.2, a numerical model of the Robogymnast is obtained using Matlab® M-files where

$$A = \begin{bmatrix} 0_3 & I_3 \\ A_{21} & A_{22} \end{bmatrix}, A_{21} = \begin{bmatrix} -36.42 & -0.35 & 0.21 \\ 13.10 & -22.06 & -223 \\ 2.14 & -1.50 & -5.68 \end{bmatrix}$$

$$A_{22} = \begin{bmatrix} -0.20 & 88.38 & 9.17 \\ 0.20 & -168.29 & 7.70 \\ 0.02 & 7.69 & -201.45 \end{bmatrix}, B = \begin{bmatrix} 0_{3x2} \\ -15.19 & -0.74 \\ 28.92 & -0.62 \\ -1.32 & 16.21 \end{bmatrix}$$

The A, B and C matrices are then converted to discrete time using Matlab® with a sampling time of *t*=2.45ms and the matrix Ad is obtained.

$$Ad = \begin{bmatrix} 0.99 & -2.43e^{-3} & -2.35e^{-4} & 2.49e^{-2} & 1.01e^{-2} & 1.19e^{-3} \\ 1.49e^{-3} & 0.99 & -2.72e^{-4} & 3.78e^{-5} & 5.87e^{-3} & 2.15e^{-4} \\ 2.55e^{-4} & -2.22e^{-4} & 0.99 & 5.29e^{-6} & 2.15e^{-4} & 4.94e^{-3} \\ -0.77 & -0.23 & -2.39e^{-2} & 0.99 & 0.52 & 6.36e^{-2} \\ 7.59e^{-2} & -0.13 & -1.43e^{-2} & 2.63e^{-3} & 1.55e^{-2} & 2.00e^{-3} \\ 1.32e^{-2} & -1.21e^{-2} & -2.85e^{-2} & 3.96e^{-4} & 2.05e^{-3} & 6.54e^{-3} \end{bmatrix}$$

$$Bd = \begin{bmatrix} -1.73e^{-3} & -9.66e^{-5} \\ -3.28e^{-3} & -1.75e^{-5} \\ -3.73e^{-5} & 1.61e^{-3} \\ -8.88e^{-2} & -5.14e^{-3} \\ 0.17 & -1.83e^{-4} \\ 3.91e^{-4} & 7.99e^{-2} \end{bmatrix}, \quad Cd = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The matrices A, B and C in equations (3.13) and (3.14) are then replaced with matrices *Ad*, *Bd* and *Cd* respectively to obtain discrete time equations (3.15) and (3.16):

$$x_{(k+1)} = Adx_k + Bdu_k \tag{3.15}$$

$$y_{(k+1)} = Cdx_{(k+1)} \tag{3.16}$$

3.4 Summary

This chapter has provided a description of the Robogymnast's system and its mathematical modelling. A detail description of the entire Robogymnast system and its components was given. The system's process flow was described and illustrated. The derivation of the mathematical equation based on the Euler-Lagrange approach was also demonstrated. A linearized equation of motion and a state-space equation of the Robogymnast in a downward position were produced from the derivation. The mathematical equation and the state-space equation are needed in order to observe and study the system's behaviour with different types of controllers before implementing the controllers on the actual system. It will be utilised in Chapter 4, in which the swing-up control of the Robogymnast will be discussed.

CHAPTER 4

Swing-Up Control of the Triple Link Pendulum

4.1 Introduction

Various motion controls have been implemented on inverted pendulums, such as swinging, swing-up and inverted balancing (Åström and Furuta 2000; Xin and Kaneda 2007a; Park et al. 2011; Yoshida 1999; Lee et al. 2015; Kharola et al. 2016; Xin and Yamasaki 2012; Eom and Chwa 2015; Xin and Kaneda 2001; Rubi et al. 2002; Cheng et al. 2013; Kamil et al. 2012; Eldukhri and Pham 2010). In this chapter, Invasive Weed Optimization (IWO) is used to tune the parameters of the swing-up controller developed by Eldukhri and Pham (2010). The main goal of this chapter is to select the optimal control parameters to achieve the fastest swing-up motion for the Robogymnast.

The remainder of the chapter is organized as follows. Section 4.2 introduces the swing-up control of the triple link pendulum and the equations related to its control signals. Section 4.3 discusses Invasive Weed Optimization (IWO), explaining its process. Section 4.4 describes the implementation of the IWO in tuning the control signal parameters of the swing-up control. Section 4.5

presents the IWO results. The control parameters are then implemented on the system and the simulation and experiment results are also presented in this section. Section 4.6 provides the discussion and conclusion of the results. A summary of the chapter is given in section 4.7.

4.2 Swing-up control

The swing-up motion of the Robogymnast (as seen in Figure 4.1) is a sequence of motions in which the Robogymnast swings from a stable pendant configuration to an inverted unstable configuration.

Figure 4.1 Robogymnast in mid-swing

This is achieved by controlling the parameters of the input voltages to the two motors (Eldukhri and Pham 2010). The equations that govern the input voltages are

$$u_1 = A_1\alpha_1 \sin\phi_1 \tag{4.1}$$

$$u_2 = A_2\alpha_2 \sin\phi_2 \tag{4.2}$$

Where u_1 and u_2 are the input voltages for motor 1 and motor 2. The voltages are controlled by adjusting the following parameters

$$\alpha_1(n+1) = \alpha_1(n) + \Delta\alpha_1 \tag{4.3}$$

$$\alpha_2(n+1) = \alpha_2(n) + \Delta\alpha_2 \tag{4.4}$$

$$\phi_1(n+1) = \phi_1(n) + \left(\eta/\delta_1\right) \tag{4.5}$$

$$\phi_2(n+1) = \phi_2(n) + \left(\eta/\delta_2\right) \tag{4.6}$$

$$\delta_1(n+1) = \delta_1(n) + \Delta\delta_1 \tag{4.7}$$

$$\delta_2(n+1) = \delta_2(n) + \Delta\delta_2 \tag{4.8}$$

$\Delta\alpha_1$ and $\Delta\alpha_2$ are the increments of amplitudes. $\Delta\delta_1$ and $\Delta\delta_2$ are the increments/decrements of the frequencies. Because u_1 and u_2 are sinusoidal cycle inputs (multiple of sampling intervals T_s depending on the value of δ_1 and δ_2), ϕ_1 and ϕ_2 vary between 0 and 2π with a step increment of η/δ_1 and η/δ_2 respectively. At the end of each duty cycle, α_1, α_2, δ_1 and δ_2 are

increased by$\Delta\alpha_1$, $\Delta\alpha_2$, $\Delta\delta_1$ and $\Delta\delta_2$ respectively. Voltages u_1 and u_2 have been limited to be between -10V and 10V in order to avoid damaging the motors. The value of constant A_1 is fixed at 3.4 and A_2 is fixed at 2.5. The values of α_1, α_2, δ_1 and δ_2 are initially set at 1, while the value of constant η is set at 0.3142 (Eldukhri and Pham 2010). The dynamic behaviour of the Robogymnast during the swing-up motion was simulated using a MATLAB® program developed by the author using the discrete state space equations (equations (3.15) and (3.16)).

In previous work, Eldukhri and Pham (2010) achieved swing-up motion of the Robogymnast by varying the amplitudes and frequencies of u_1 and u_2 using a single parameter δ whose periodic increment $\Delta\delta$ was obtained through trial and error. Kamil et al. (2012) separated the increments of α and δ into Δα and Δδ respectively. The Bees Algorithm was then employed to find the optimum values of Δα and Δδ.

In this chapter, input signals u_1 and u_2 are each assigned their own α and δ parameters. The IWO is then used to select the optimum values of $\Delta\alpha_1$,$\Delta\alpha_2$, $\Delta\delta_1$ and $\Delta\delta_2$.

4.3 Invasive Weed Optimization Algorithm

Invasive Weed Optimization (lWO), first designed and developed by Mehrabian and Lucas, is a novel numerical stochastic optimization algorithm inspired by the colonization of invasive weeds (Madivada Hymavathi and Rao 2012). The robustness and seeding characteristics of weeds has been incorporated to form a swarming optimization method that is simple, flexible and effective. IWO has some distinctive properties in comparison with traditional numerical search algorithms like reproduction, spatial dispersal and competitive exclusion (Mehrabian and Lucas 2006). The procedures required in order to implement IWO in an optimization algorithm are as follows:

1- Randomly generate a finite population of seeds from the set of feasible solutions (initializing population).
2- Calculate the fitness of the population. Every seed will then reproduce based on its fitness (reproduction). In this case, the number of seeds produced is directly proportional to its fitness level.
3- The new seeds are then randomly distributed over the search area and grow into new plants (spatial dispersal) The mean of distribution is equal to the location of the parent plant, but the standard deviation (SD), σ_{iter},

will be reduced from a specified initial value, $\sigma_{initial}$, to the final value, σ_{final}, according to equation 4.7 (Ghalenoei et al. 2009).

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n} \left(\sigma_{initial} - \sigma_{final}\right) + \sigma_{final} \quad (4.7)$$

4- The process is continued until the maximum population is reached, where the lower fitness seeds are truncated (competitive exclusion). The process is continued until maximum iteration is reached.

The flowchart of the IWO process is shown in Figure 4.2.

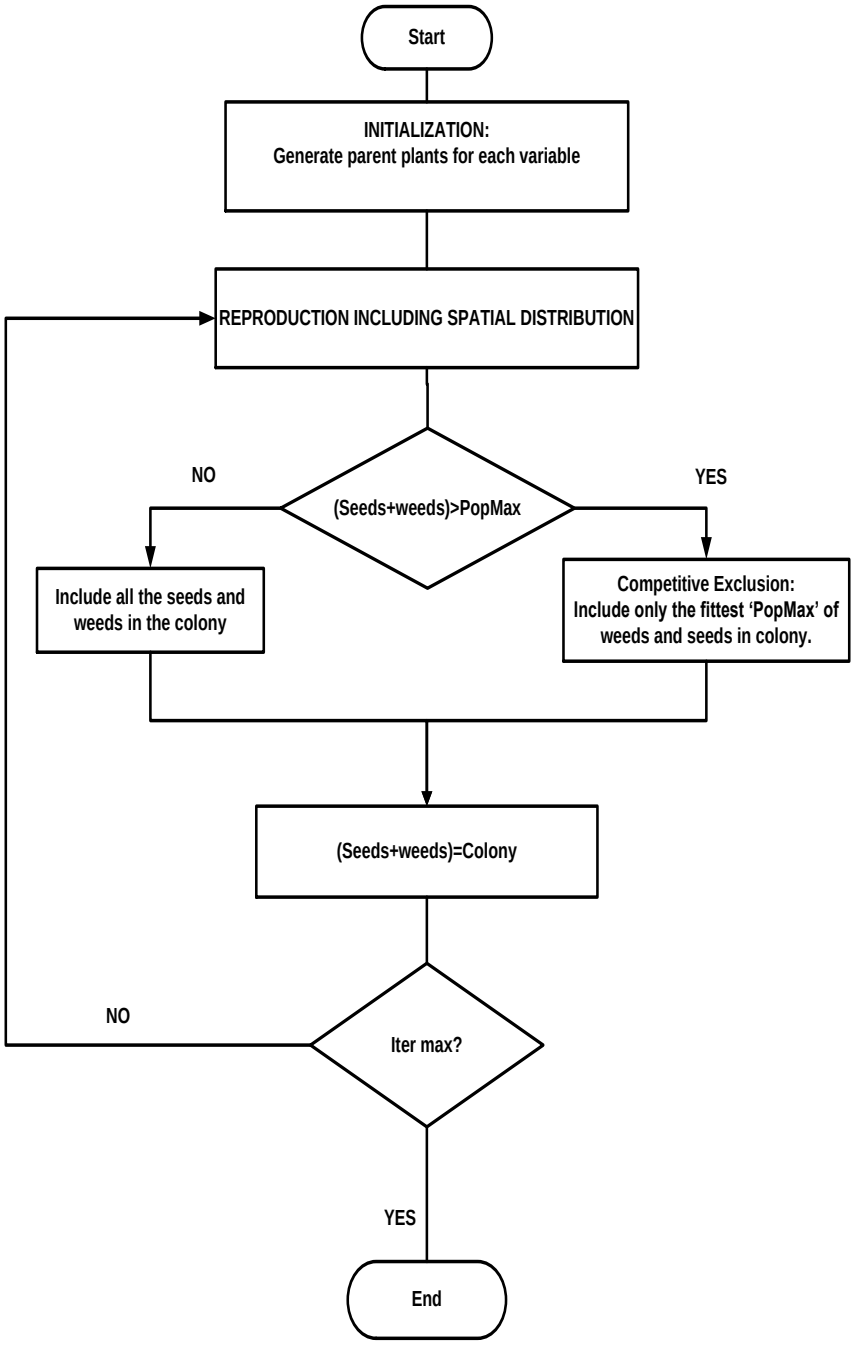


Figure 4.2: IWO Flow Chart (Madivada Hymavathi and Rao 2012)

4.4 Tuning the swing–up control parameters using IWO

IWO was used to investigate the optimum values of the variables for the swing-up of the Robogymnast due to its simplicity and flexibility. The parameters of the IWO were set as in Table 4.1. The variables investigated are $\Delta\alpha_1$ and $\Delta\alpha_2$, which are the increments of amplitudes, while $\Delta\delta_1$ and $\Delta\delta_2$ are the increments/decrements of frequency. The search range was obtained through trial and error by starting the search with the widest possible range and determining through observation where the optimum value for each variable is most likely to be found. The procedure is repeated until an acceptable range is obtained. This is done to speed up the search process of the IWO and to ensure that the optimum values are obtained.

Table 4.1: IWO Parameters

Variable	Value	Description
Number of initial plants (p_{init})	10	Number of randomly chosen values from the solution space.
Minimum number of seeds (S_{Min})	0	Minimum population of solutions
Maximum number of seeds (S_{Max})	500	Maximum population of solutions
Initial value of standard deviation ($\sigma_{initial}$)	0.04	Standard deviation used for spatial distribution of plants.
Final value of standard deviation (σ_{final})	0.01	Final standard deviation used for spatial distribution of plants.
Maximum number of iteration ($Iter_{max}$)	5	Number of iterations
Nonlinear Modulation Index (n)	0.001	-
Search range	$0<\Delta\alpha_1<0.7$ $0<\Delta\alpha_2<0.2$ $5.0<\Delta\delta_1<6.0$ $5.0<\Delta\delta_2<6.0$	Search range used based on trial and error.

i. Compute fitness of each plant by determining the time taken for the Robogymnast to swing 180° using the discrete state space Equations 3.15 and 3.16.

ii. Compute maximum and minimum fitness of colony.

iii. For each individual plant in colony ($p \in W$)

iv. Compute the number of seeds for p, corresponding to its fitness.

v. Randomly select the seeds from the feasible solutions around the parent plant (p) in a neighbourhood with spatial distribution based on standard deviation obtained from equation 4.7 and mean (μ) equal to zero.

vi. Add the generated seeds to the solution set, W

vii. Sort the population p in descending order of their fitness.

viii. If population>PopulationMax, truncate weeds with smaller fitness until: Population = PopulationMax.

ix. Continue with next iteration.

x. Repeat step ii until the maximum number of iterations.

Figure 4.3: Pseudo-code for IWO

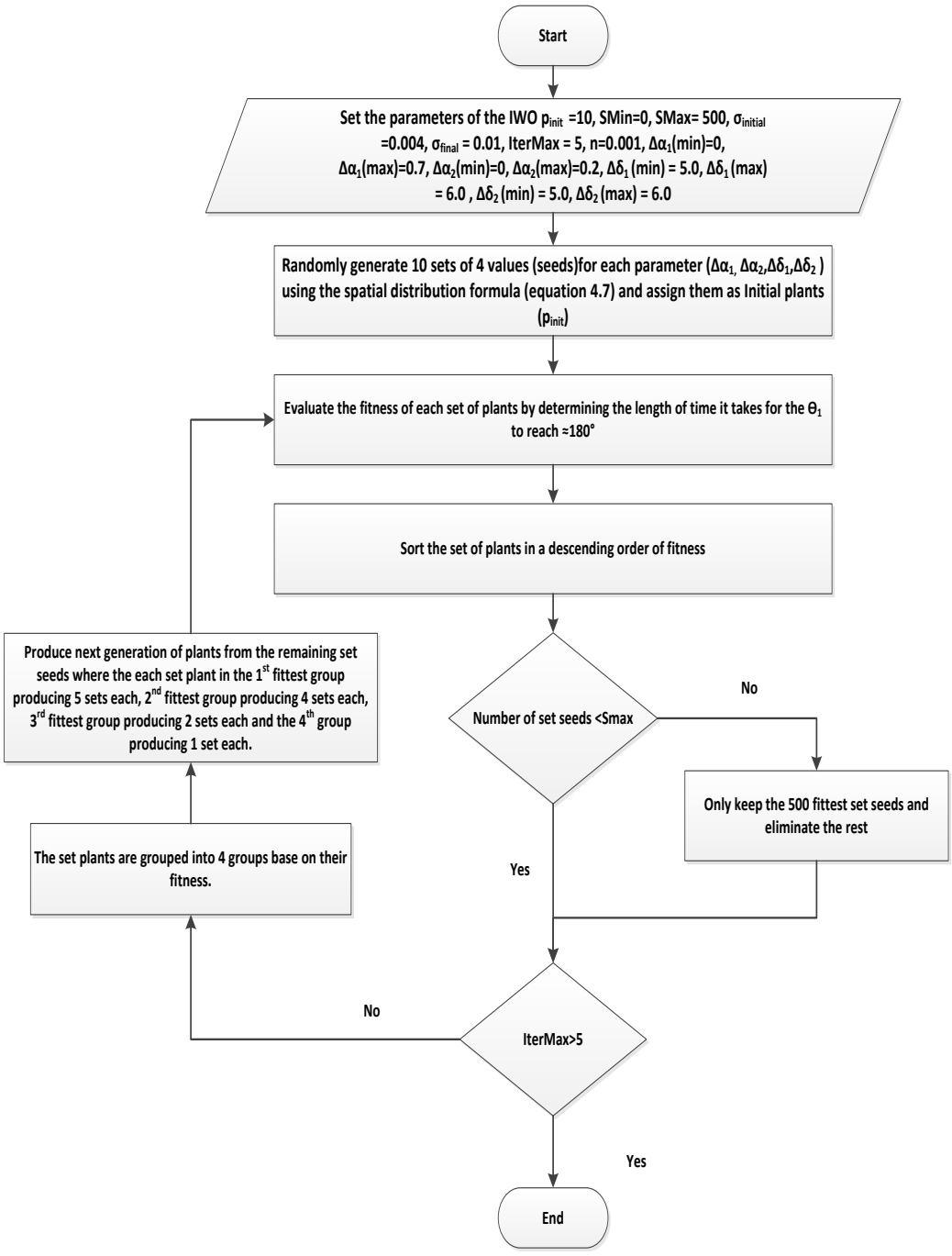


Figure 4.4: Flowchart for Invasive Weed Optimization Algorithm

4.5 Results

This section gives a review of the results obtained from the IWO search. The values are then applied to simulations and experiments and their results studied.

4.5.1 IWO Results

The parameters in Table 4.1 were applied to the IWO algorithm and used to search for the optimum values of $\Delta\alpha_1$, $\Delta\alpha_2$, $\Delta\delta_1$ and $\Delta\delta_2$. The results of the IWO algorithm shown in Table 4.2 were obtained by using an error of 0.55%. The values obtained were then applied to the MATLAB® program to simulate the dynamic behaviour of the Robogymnast in order to verify the IWO result. The system was simulated using the top four results [1]:

Set 1: $\Delta\alpha_1$=0.6924V, $\Delta\alpha_2$= 0.1966V, $\Delta\delta_1$=5.1984 rad^{-1}, $\Delta\delta_2$= 5.1129 rad^{-1}

Set 2: $\Delta\alpha_1$=0.6872V, $\Delta\alpha_2$= 0.1726V, $\Delta\delta_1$=5.4428 rad^{-1}, $\Delta\delta_2$= 5.9191 rad^{-1}

Set 3: $\Delta\alpha_1$=0.6616V, $\Delta\alpha_2$= 0.1699V, $\Delta\delta_1$=5.5194 rad^{-1}, $\Delta\delta_2$= 5.9701 rad^{-1}

Set 4: $\Delta\alpha_1$=0.6635V, $\Delta\alpha_2$= 0.1827V, $\Delta\delta_1$=5.5506 rad^{-1}, $\Delta\delta_2$= 5.9349 rad^{-1}

Simulation results in Figure 4.5 to 4.8 shows successful swing-up of the Robogymnast to an inverted configuration using the values obtained from the IWO, and these simulation results conform to IWO results.

[1] The top four results are selected based on success in the experiment.

Table 4.2: IWO Results

$\Delta\alpha_1$ (V)	$\Delta\alpha_2$ (V)	$\Delta\delta_1$ (rad^{-1})	$\Delta\delta_2$ (rad^{-1})	Angular Position of Robogymnast (θ_1 Deg)	Duration to reach the upright position(s)
0.6924	0.1966	5.1984	5.1129	-179.003	128.500
0.6833	0.1919	5.2013	5.1705	-179.018	128.525
0.6872	0.1726	5.4428	5.9191	-179.032	134.375
0.6627	0.1707	5.5217	5.9179	-179.192	135.975
0.6615	0.1646	5.5168	5.9925	-179.059	135.975
0.6593	0.1644	5.5170	5.9872	-179.084	135.975
0.6524	0.1808	5.5171	5.9668	-179.078	135.975
0.6616	0.1699	5.5194	5.9701	-179.063	136.00
0.6662	0.1548	5.5470	5.9579	-179.125	136.525
0.6596	0.1492	5.5463	5.9585	-179.098	136.525
0.6648	0.1663	5.5501	5.9432	-179.217	136.525
0.6635	0.1827	5.5506	5.9349	-179.244	136.525
0.5468	0.0157	5.1284	5.1673	-179.2190	158.050
0.5780	0.0011	5.1276	5.1721	-179.1363	158.050

4.5.2 Simulation Results

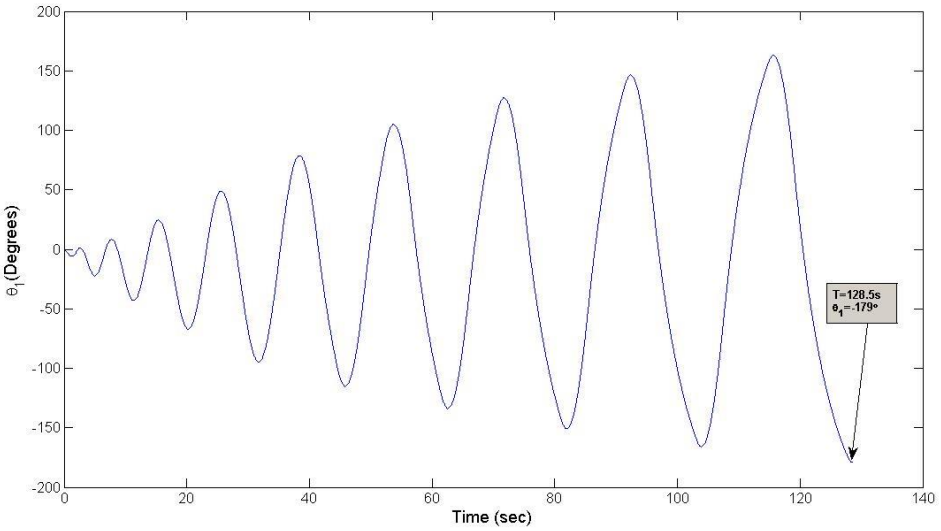


Figure 4.5: Simulated angular position Θ_1 for Set 1

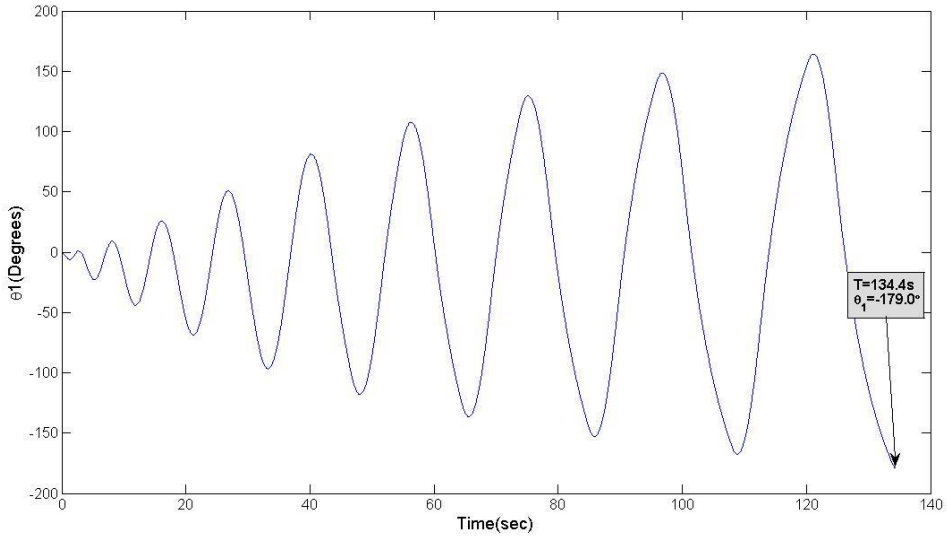


Figure 4.6: Simulated angular position Θ_1 for Set 2

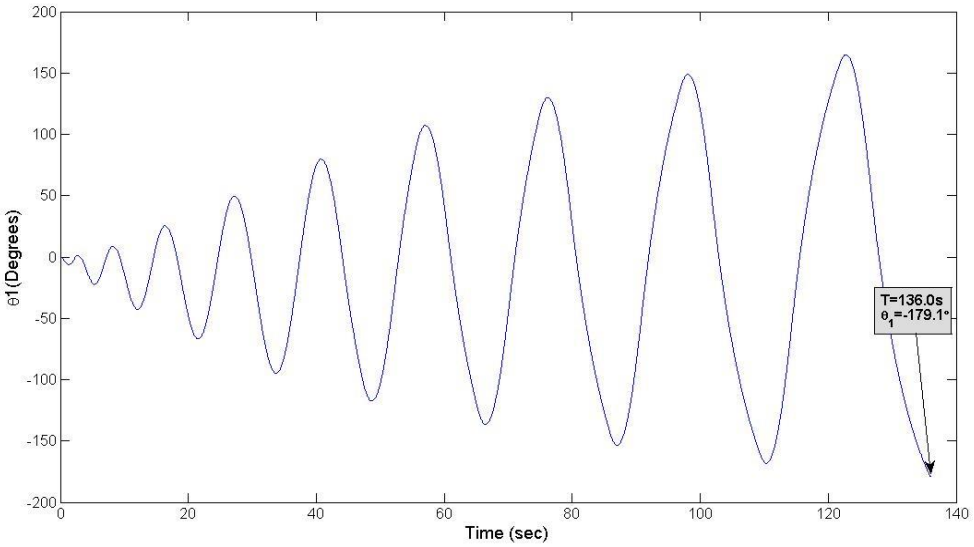


Figure 4.7: Simulated angular position Θ_1 for Set 3

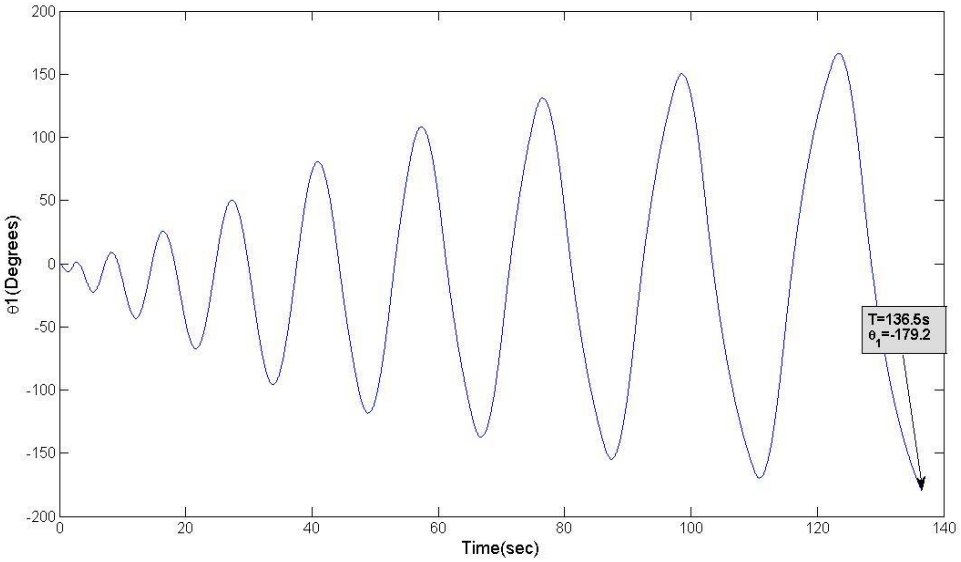


Figure 4.8: Simulated angular position Θ_1 for Set 4

The fastest swing-up motion was achieved at using Set 1 parameters. The Robogymnast reached an angle of -179° in 128.5 seconds. All the swing-up motions displays similar pattern, where the swinging starts with high frequencies and ends with smaller frequencies but larger amplitudes.

4.5.3 Experiment Results

Values of the parameters obtained using IWO were applied to the actual system. The experiments conducted show that the Robogymnast will only be able to obtain a smooth swing-up motion if the motor voltages u_1 and u_2 have the same frequency, contrary to the simulation results. If the voltages are not in phase, the Robogymnast will not be able to achieve the natural frequency required for smooth swing-up motion. Thus, in order to obtain a satisfactory result, the assumption that $\Delta\delta_2 = \Delta\delta_1$ had to be made. The larger $\Delta\delta$, the slower the frequency of the sinusoidal function applied for the two motors. In the real system, it has been observed that the system will perform better if the sinusoidal signals applied to the motors start at a relatively fast frequency. The value of $\Delta\delta$ obtained from the simulation results was divided by 100 in order to achieve smooth motion and to avoid damaging the robot's motor/gearbox structures caused by the inherent backlash in the gearboxes. Figures 4.9 to 4.12 illustrates the performance the Robogymnast during experiments.

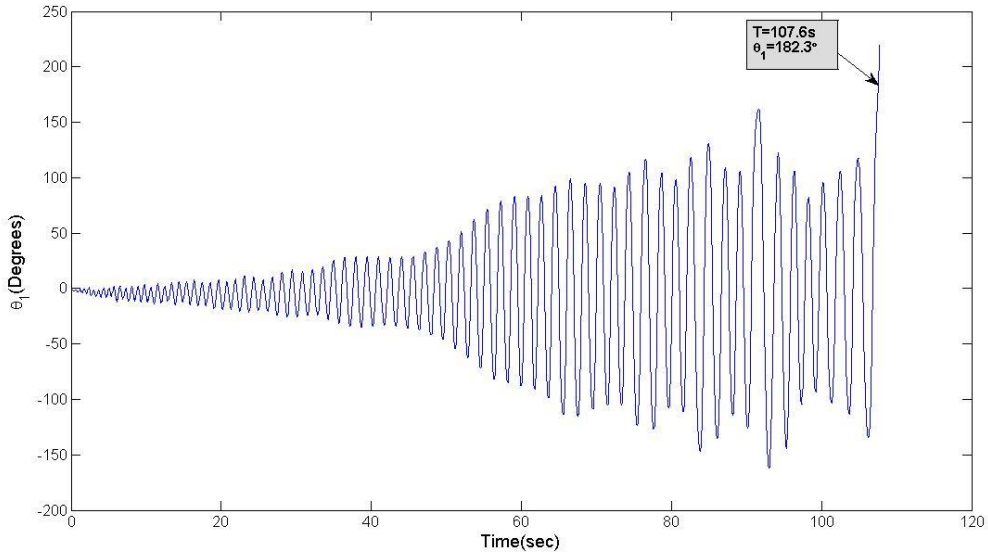


Figure 4.9: Measured angular position Θ_1 for Set 1.

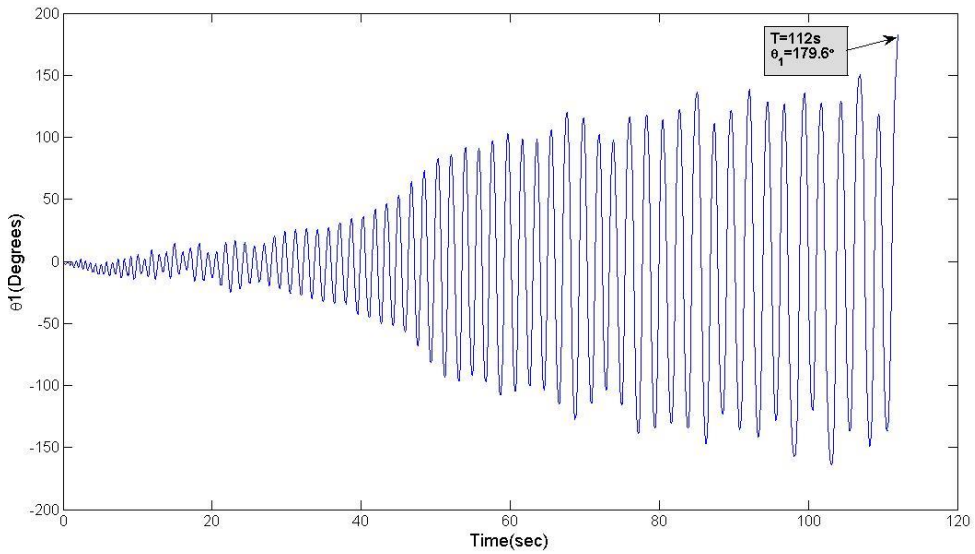


Figure 4.10: Measured angular position Θ_1 for Set 2

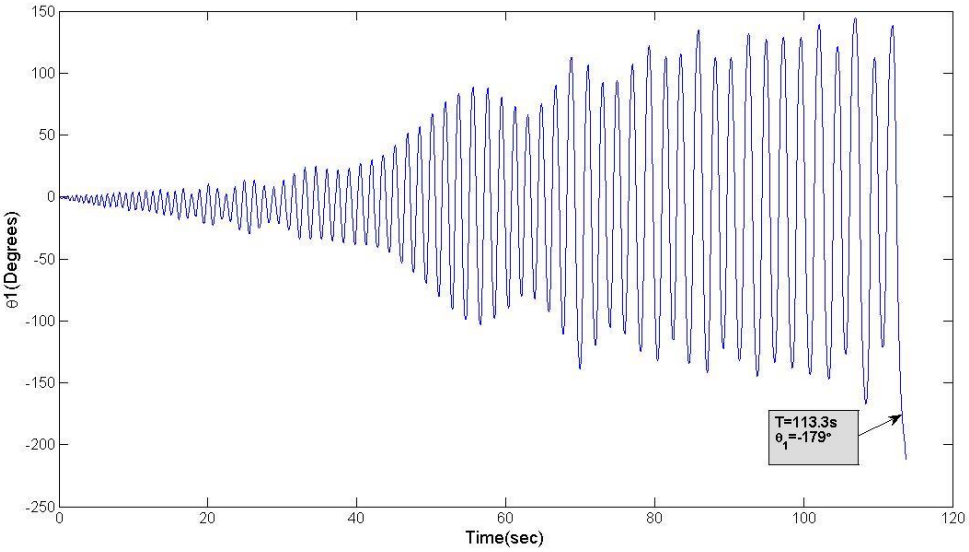


Figure 4.11: Measured angular position Θ_1 for Set 3

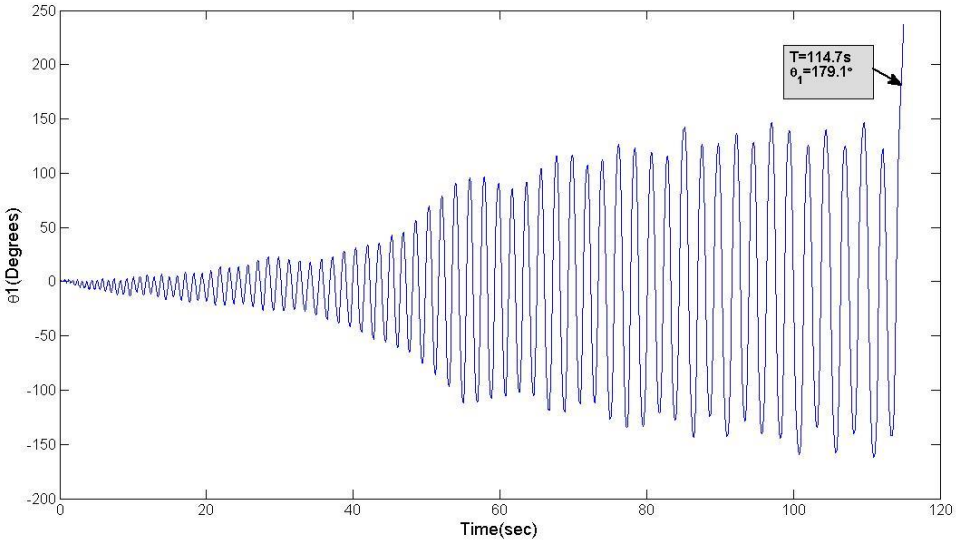


Figure 4.12: Measured angular position Θ_1 for Set 4

4.6 Discussion and Conclusion

The simulation results obtained show that IWO could be used to find the optimal variables required to swing the Robogymnast more efficiently. The results show that the amplitude of u_1 must be higher than u_2, while the difference between $\Delta\delta_1$ and $\Delta\delta 2$ ranges from 0.6% to 8.05% of each other. In all the results, u_2 does not exceed the value of 6V, while u_1 will reach the maximum value of 10V in about 20 seconds. This shows that the value of u_2 does not affect the swing of the Robogymnast as much as u_1. The results also show that the higher the value of $\Delta\delta_1$ and $\Delta\delta 2$, the faster the frequency of the swing decreases and the faster the Robogymnast will swing up to 180°. However, if the time taken to reach the upright position is relatively short, it may cause damage to the motor/gearbox structures (Kamil et al. 2012). Thus, a compromise must be made in order to obtain an optimized swing-up movement without damaging the Robogymnast. From observation, it would appear that $\Delta\alpha_1$, $\Delta\alpha_2$ and $\Delta\delta$ of 0.6872V, 0.1726V and 0.05543 rad^{-1} respectively will give the smoothest swing-up motion. The experimental results vary when compared to the simulation result, as shown in Table 4.3. This may be caused by external factors such as friction, inertia and backlash.

Table 4.3: Simulation Results vs. Experimental Results

$\Delta\alpha_1$ (V)	$\Delta\alpha_2$ (V)	$\Delta\delta_1^*$ (rad^{-1})	$\Delta\delta_2^*$ (rad^{-1})	Duration to reach the upright position (seconds)	
				Simulation	Experiment
0.6924	0.1966	5.1984	5.1129	128.5	107.6
0.6872	0.1726	5.4428	5.9191	134.4	112.0
0.6616	0.1699	5.5194	5.9701	136.0	113.3
0.6635	0.1827	5.5506	5.9349	136.5	114.7

*Value of $\Delta\delta$ is divided by 100 when applied in the experiment

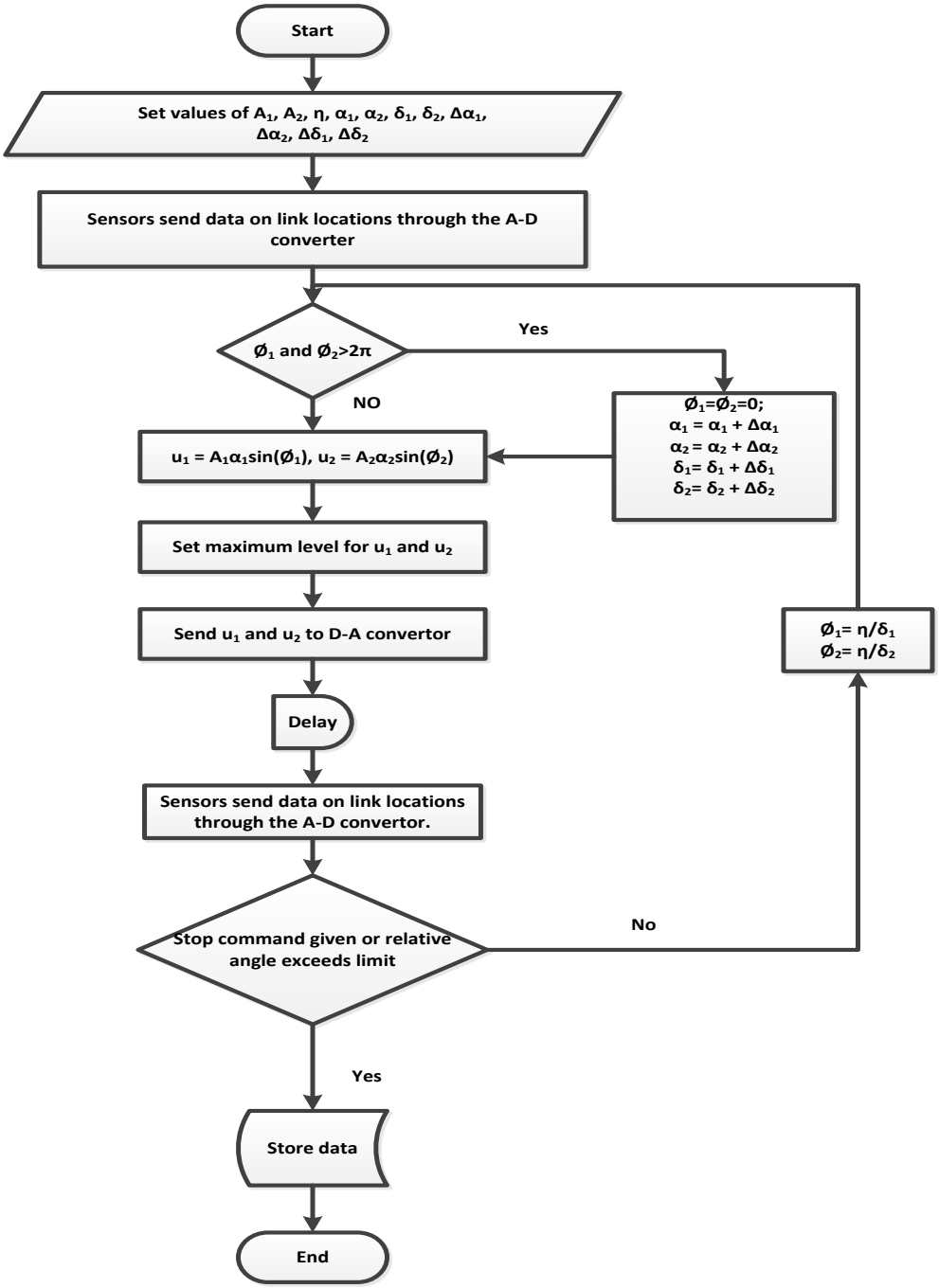


Figure 4.13: Flowchart of Robogymnast swing-up sequence

4.7 Summary

In this chapter, Invasive Weed Optimization (IWO) was used to investigate the optimum values of the control parameters for the swing-up of the Robogymnast developed by Eldukhri and Pham (2010). Kamil et al. (2012) independently manipulated the amplitudes and the frequencies of the control signal. They also optimized the parameters ($\Delta\alpha$ and $\Delta\delta$) of the two motor control signals (u_1 and u_2) using the swarm-based Bees Algorithm (BA). In this chapter, two parameters ($\Delta\alpha_1$ and $\Delta\delta_1$) were assigned to the control signal u_1 and another two parameters ($\Delta\alpha_2$ and $\Delta\delta_2$) were assigned to the control signal u_2. IWO was used to optimize the swing-up motion of the robot by determining the optimum values of parameters that control the input sinusoidal voltage of the two motors. The values obtained from IWO were then applied to both simulation and experiment. Results showed that the swing-up of the Robogymnast from the stable downwards position to the inverted configuration was successfully accomplished. In the following chapter, the Artificial Neural Network Model of the Robogymnast will be discussed.

CHAPTER 5

Artificial Neural Network Modelling of the Robogymnast

5.1 Introduction

A model is a precise representation of a system's dynamics used to answer questions via analysis and simulation (Aström and Murray 2010). A mathematical model is a mathematical representation of a system (Spong et al. 2006). Mathematical modelling has long been essential in the study and design of dynamical systems. It provides an approximation of real-world conditions. It is also economical, as it provides the means of optimizing a design before actually building it. However, a mathematical model becomes less accurate as its complexity increases. This is because modelling is a process of simplification and deduction. Simplification involves loss of information about a situation (Schrodt and Johnson 2004). The system being studied here is a complex multi-link under-actuated mechanism which requires a complex mathematical model that takes into consideration a great deal of information. Under-actuated mechanisms provide a

useful test bed for the evaluation and comparison of different control techniques (Eldukhri and Pham 2010). Eldhukri and Pham succeeded in swinging the Robogymnast from a stable pendant position to an inverted unstable configuration (Eldukhri and Pham 2010). This was further improved through the optimization of the control parameters by implementing the Bees Algorithm (BA: Kamil et al. 2012) and Invasive Weed Optimization (IWO) in Chapter 4. However, all the previous studies require tuning of the parameters in order to apply them to the real system. This shows that the mathematical model, though useful, is not sufficient when it comes to modelling the system. Proponents of neural networks claim that their versatility and robustness makes them suitable for various applications, such as modelling and control. The neural network is commonly employed for nonlinear modelling of a system. Neural networks possess various attractive features such as massive parallelism, distributed representation and computation, generalization ability, adaptability and inherent contextual information processing (Jain et al. 1996). Toha and Tokhi (2008) designed a Multi-Layer Perceptron neural network (MLP) model and an Elman recurrent Neural Network (ENN) model of a Twin Rotor Multi-input multi-output System (TRMS). The models were trained with the Levenberg-Marquardt (LM) method using experimental data to characterize the dynamic behaviour of the system. Both models yield very similar accurate results with the ENN model, providing slightly better prediction of the system's behaviour. Gao et al. (1996) designed a modified ENN model of a dynamic system with random outputs and compared it with the conventional ENN. The modified ENN

consists of extra adjustable weights between the neurons of the context layer and the output layer, similar to that of a Jordan recurrent network (Pham and Karaboga 1999). The modified ENN performed comparably to the conventional ENN but required only 121 iterations to converge compared to 603 iterations for the conventional model, thus making the training process faster. Zhang (2003) proposed a time series forecasting mechanism using a hybrid autoregressive integrated moving average (ARIMA) and ANN model. The model uses ARIMA to handle the linear parts of the time series and ANN to handle the nonlinear parts of the time series. Results prove that ANNs are flexible computing frameworks for modelling a broad range of nonlinear problems and can approximate a large class of functions with a high degree of accuracy. This chapter proposes an Elman neural network model of the Robogymnast and compares it with the mathematical model. The Elman neural network is a recurrent neural network model created by Jeffrey L. Elman (Elman 1990). It can be trained using various methods such as the standard back-propagation learning algorithm (Pham and Liu 1996; Pham and Karaboga 1999).

This chapter proceeds as follows. Section 5.2 introduces the Elman Neural Network and its application in modelling the Robogymnast. Section 5.3 discusses the activation function and the justification for its selection. Section 5.4 then briefly explains the back-propagation algorithm. The training of the ENN model is explained in section 5.5 with section 5.6 providing the results. In section 5.7, the

results are further discussed and conclusions are drawn. The final section provides a summary of the chapter.

5.2 Elman Neural Networks

The Elman Neural Network (ENN) model shown in Figure 5.1 is similar to the feed-forward network in Figure 2.2, but has an extra layer called the context layer. The neurons in the context layer are used only to memorize the previous activations of the hidden units and can be considered to function as a one-step delay. The input layer consists of two neurons, which will receive the input voltages to the two motors of the robot gymnast, where u_1 is the input voltage for motor 1 and u_2 is the input voltage for motor 2. The hidden layer consists of six neurons which will produce six outputs that will represent the state vector of the robot gymnast. The six states are the relative angles θ_1, θ_2, θ_3, and their respective velocities $\dot{\theta}_1$, $\dot{\theta}_2$ and $\dot{\theta}_3$. The output layer consists of three neurons representing the three output angles of the system.

The ENN model can be represented using equations (5.1) and (5.2).

$$X(k) = W_{cx}X(k-1) + W_{ih}U(k-1) \tag{5.1}$$

$$Y(k) = W_{ho}X(k) \tag{5.2}$$

Where X is the output of the neurons in the hidden layer, Y is the output of the neurons in the output layer and U is the input for the model. W_{cx}, W_{ih}, W_{ho} are the

weight matrices for the context layer, the input layer and the output layer respectively. It is important to highlight that the matrix W_{cx} refers to the weights from the context layer to the hidden layer. The outputs from the hidden layer to the context layer are unweighted. The ENN model was selected as the model for the Robogymnast because the position of its context layer allows the previous values of the state vector to be stored and reused as inputs for the next state vectors. This factor makes the behaviour of the ENN similar to that of the state space equation. It can be observed that Equation 5.1 and Equation 5.2 are similar to the discrete time equations (Equation 3.15 and Equation 3.16) where the *Ad*, *Bd* and *Cd* matrices are replaced by W_{cx}, W_{ih} and W_{ho} respectively. This makes it much easier to transfer the mathematical model to the ENN model without requiring any changes to the states.

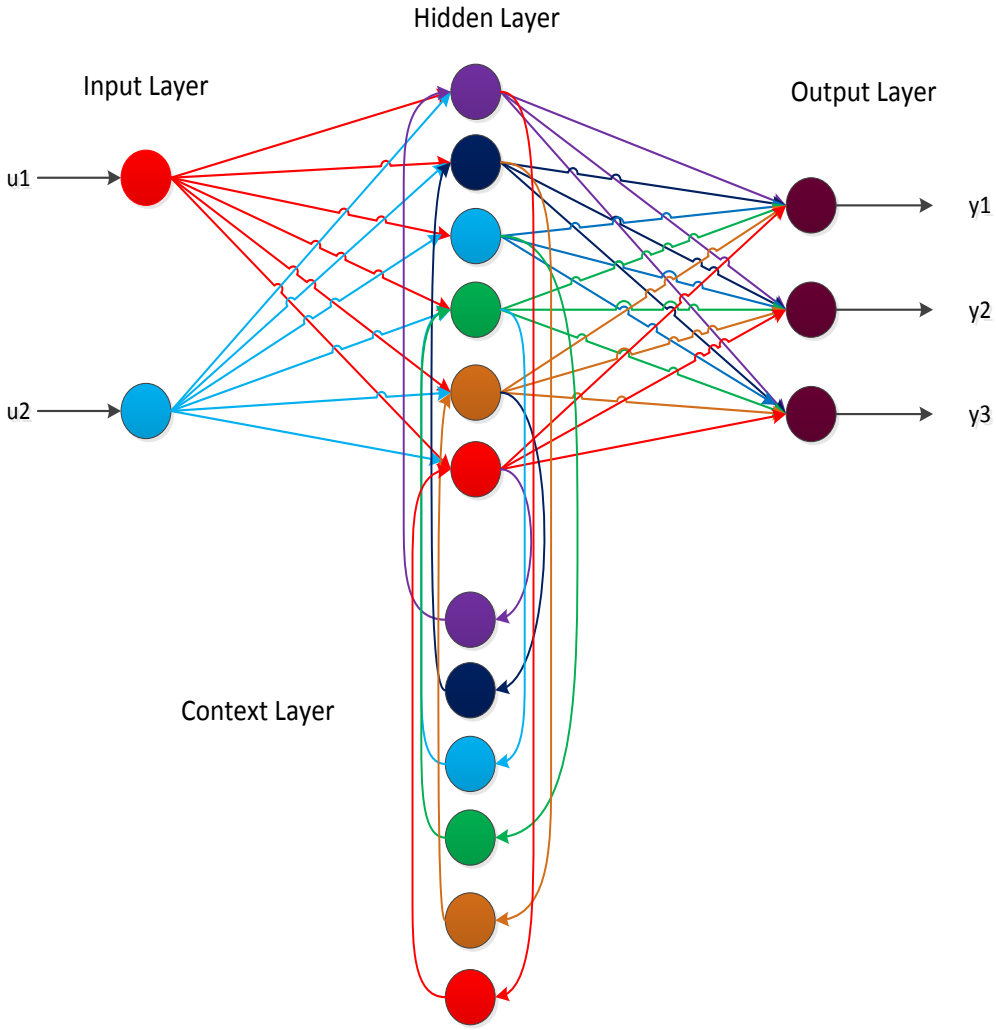


Figure 5.1: Elman Neural Network Diagram of Robogymnast

5.3 Activation Function

An activation function is responsible for activating the neuron's output. Many activation functions used in ANNs produce a continuous value rather than a discrete value (Youssef and Aly 2013). Two of the most popular activation functions used are the logistic activation function, more popularly referred to as the sigmoid function, and the identity activation function (linear activation function) (Jones 2004).

- The Logistic Activation (Sigmoid) Function

$$f(net_j) = \frac{1}{1 + e^{-z}}$$

- The Identity Activation (linear) Function

$$f(z) = z$$

Where z is the value of the input to the neurons.

Both types of activation function were implemented on the ENN model. However, the identity activation function appears to produce better results, while the logistic activation function experiences premature saturation as the input of the ENN model becomes non-linear. Due to its linearity, the identity activation function also requires less computation compared to the logistic activation function. This is a huge advantage when it comes to training large amounts of data. These two factors

are deemed to make it the best candidate for the activation function of the ENN model.

5.4 Back-Propagation Algorithm

The ENN modelling was trained using the Back-Propagation (BP) algorithm. The BP algorithm is based on the generalized delta rule proposed in 1985 by the PDP research group headed by Dave Rumelhart, based at Stanford University, California, U.S.A (Sharma et al. 2012). Before the BP can be used, it requires target patterns or signals, as it is a supervised learning algorithm. Training patterns are obtained from the samples of the types of inputs to be given to the multilayer neural network and their answers are identified by the user. The configuration for training a neural network using the BP algorithm is shown in Figure 5.2, in which the training is done offline.

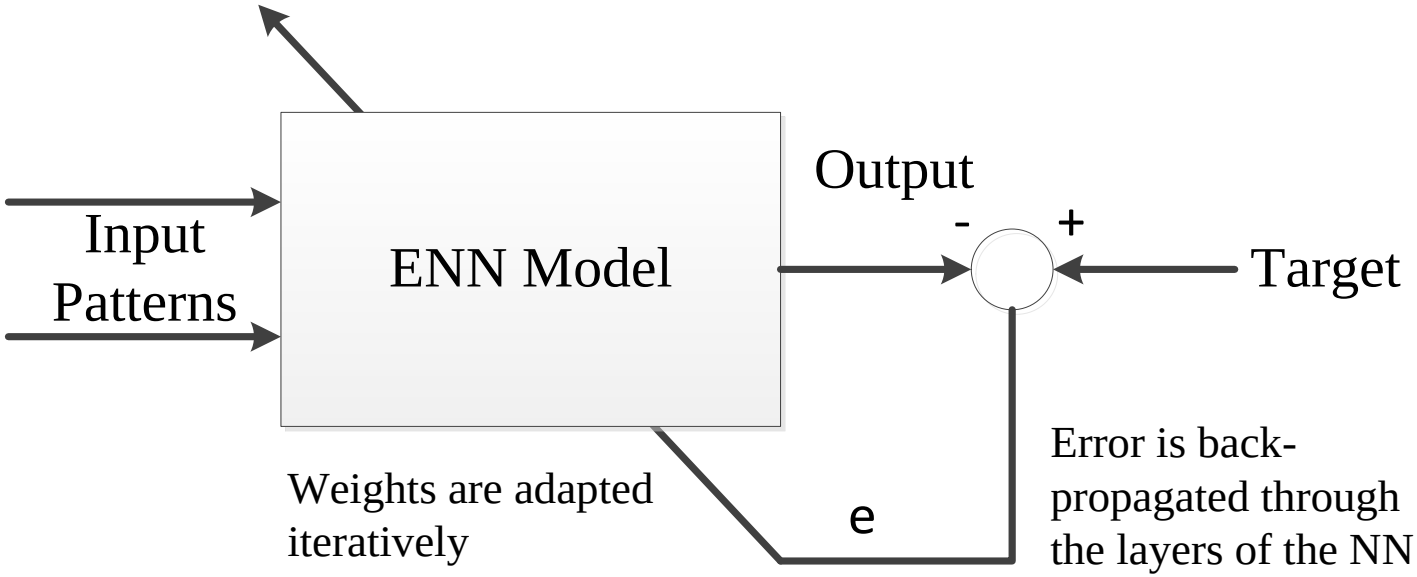


Figure 5.2: Back-propagation Configuration

The objective is to minimize the error between the target and actual output and to find ΔW (increment of weights). The error is calculated for every iteration and is back-propagated through the layers of the ENN to adapt the weights. Equations (5.3) and (5.4) are used for the back-propagation of weight adjustment of the ENN model (Pham and Liu 1995).

$$\Delta W_{ih} = \eta(y_r(k) - y_{ENN}(k)){W_{ho}}^T U(k) \quad (5.3)$$

$$\Delta W_{cx} = \eta(y_r(k) - y_{ENN}(k)){W_{ho}}^T X^T(k-1) \quad (5.4)$$

Where ΔW_{ih} , ΔW_{cx} are the weight increments for W_{ih}, W_{cx} and η is the learning rate of the learning process. y_r and y_{ENN} are the training data output and ENN output. Once the maximum number of iterations has been reached, the training is stopped, and the neural network is reconfigured in the recall mode to solve the task. W_{ho} is not adjusted and remains as the following to maintain the homogeneity of the output:

$$W_{ho} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

5.5 Training the ENN Model

The ENN model was trained using data from the swing-up control of the Robogymnast obtained from the experiments conducted in Chapter 4 and inputs generated by the simulation program. The back-propagation training of the model was done using a Matlab® program created by the author using the parameters in Table 5.1. To make the learning process faster, the values of A_d and B_d are taken as initial values of W_{cx} and W_{ih} respectively.

Back-propagation learning of ENN model algorithm:

1. Read the ENN parameters (u_1, u_2, y_1, y_2, y_3), plant parameters (number of inputs, number of layers, number of neurons, number of outputs), and change in load value.
2. Initialize the ENN weight matrices with initial values.
3. Set the initial state vector of the plant and the desired target vector.
4. Set the number of iterations.
5. Execute the feed-forward propagation for the neural network.
6. Find the error of the plant output.
7. Execute the back-propagation for the neural network.
8. Apply new weight increments to the currents weights.
9. Check whether the maximum number of iterations has been reached.
10. If the maximum number of iterations has not been reached, repeat step 5 to step 9; else end program.

Table 5. 1: Parameters of Back-Propagation Training of ENN Model

Parameters of Neural Network Modelling	Values	Description
Number of inputs	2	Input voltages u_1 and u_1
Number of outputs	3	Output angle $y_1.y_2, y_3$
Number of layers	4	1 input layer, 1 hidden layer, 1context layer, 1 output layer
Number of neurons in hidden layer	6	6 neurons representing the 6 variables in the state vector.
Number of neurons in context layer	6	The context layer acts as a step delay for the state vector.
Number of samples	20000	All data is taken from a range of 0-20000 data sets.
Learning rate η	0.01	Training parameter that controls the size of weight and bias changes during learning.
Number of Iterations	100	The training is repeated perform until the maximum number of iterations is achieved.

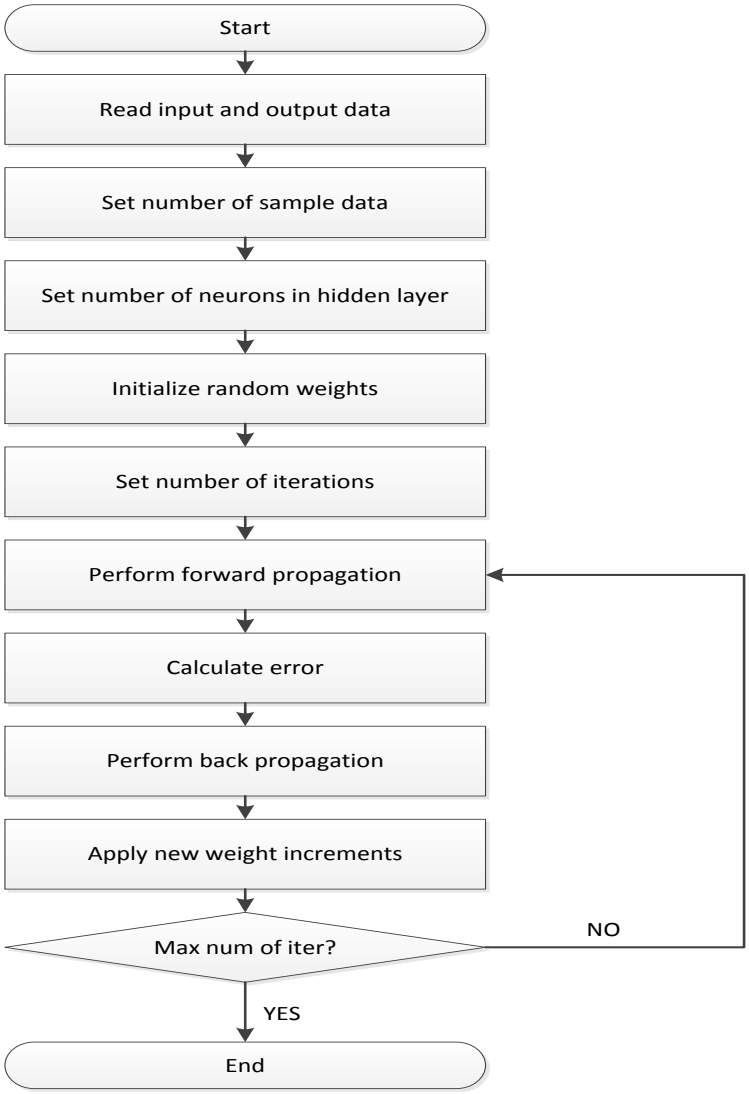


Figure 5.3: Flow Chart of Back-Propagation Training for Robogymnast

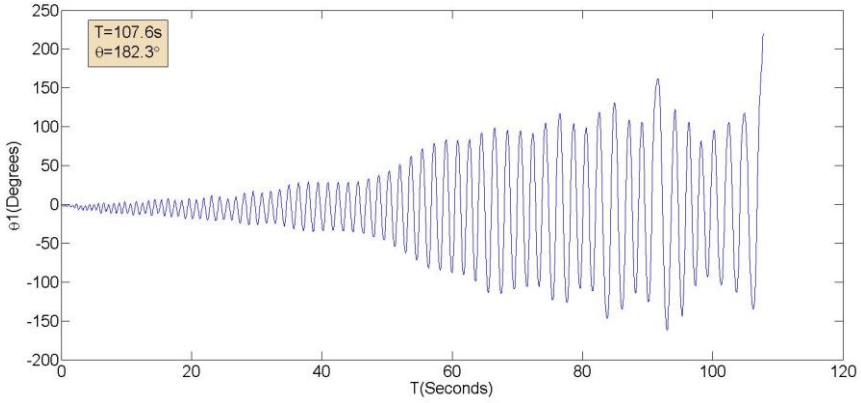
5.6 Results

The training process yields the following weights:

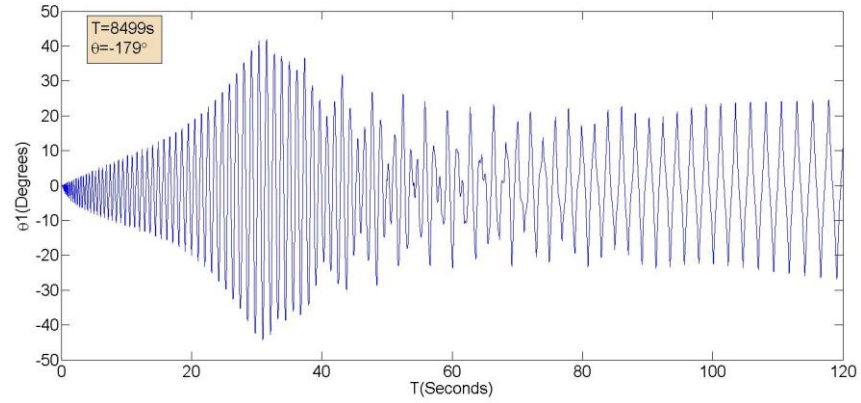
$$W_{cx} = \begin{bmatrix} 0.99 & -2.43e^{-3} & -2.34e^{-4} & 2.49e^{-2} & 1.01e^{-2} & 1.20e^{-3} \\ 1.49e^{-3} & 0.99 & -2.72e^{-4} & 3.78e^{-5} & 5.88e^{-3} & 2.15e^{-4} \\ 2.55e^{-4} & -2.22e^{-4} & 1.00 & 5.30e^{-6} & 2.15e^{-4} & 4.95e^{-3} \\ -0.77 & -0.23 & -2.39e^{-2} & 0.99 & 0.52 & 6.37e^{-2} \\ 7.59e^{-2} & -0.13 & -1.43e^{-2} & 2.64e^{-3} & 1.55e^{-2} & 2.01e^{-3} \\ 1.32e^{-2} & -1.21e^{-2} & -2.85e^{-2} & 3.97e^{-4} & 2.05e^{-3} & 6.55e^{-3} \end{bmatrix}$$

$$W_{ih} = \begin{bmatrix} -2.91e^{-3} & -1.62e^{-4} \\ -5.51e^{-3} & -2.94e^{-5} \\ -6.26e^{-5} & 2.71e^{-3} \\ -0.15 & -8.61e^{-3} \\ 0.283 & -3.08e^{-4} \\ -6.55e^{-4} & 0.13 \end{bmatrix}$$

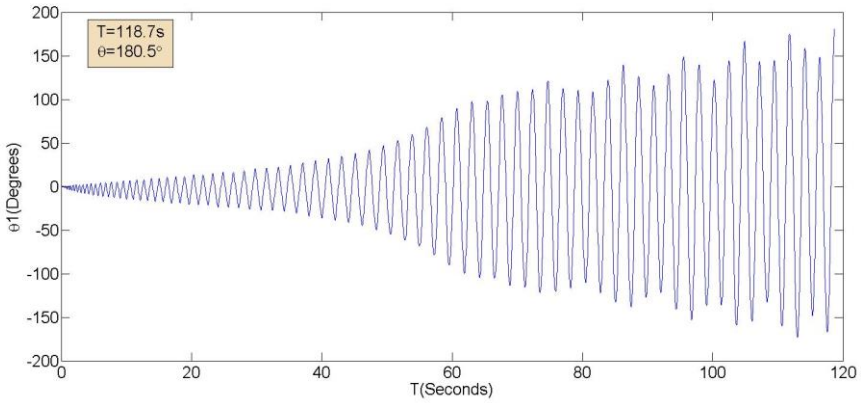
The weights were then applied in Equations 5.1 and 5.2 with various inputs and their outputs analysed and compared with the experimental and the mathematical model outputs.



(a)

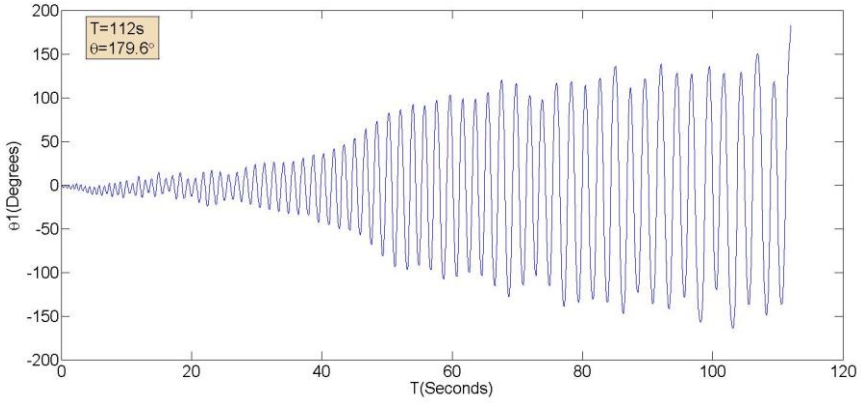


(b)

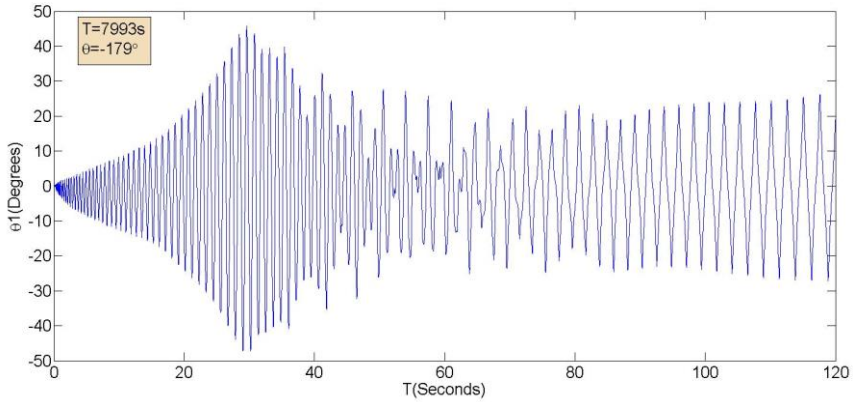


(c)

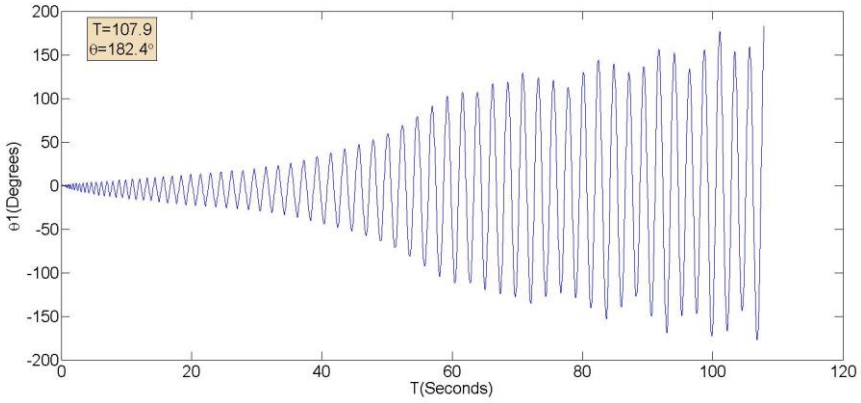
Figure 5.4: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6924, $\Delta\alpha_2$= 0.1966, $\Delta\delta$=0.051129 for (a) Experimental; (b) Mathematical Model; (c) ENN Model



(a)

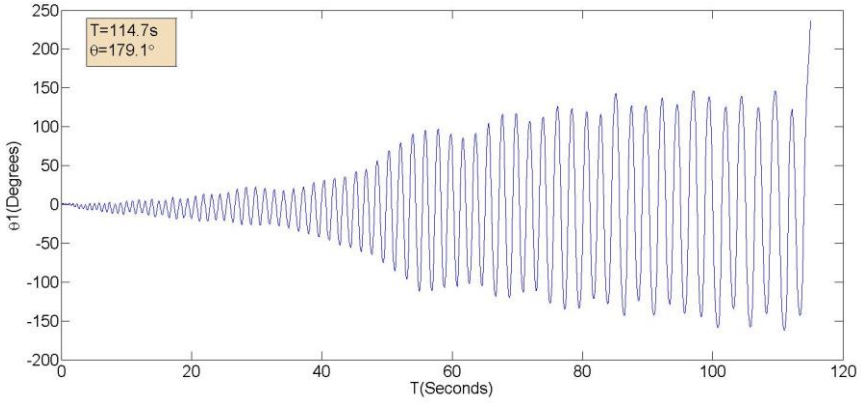


(b)

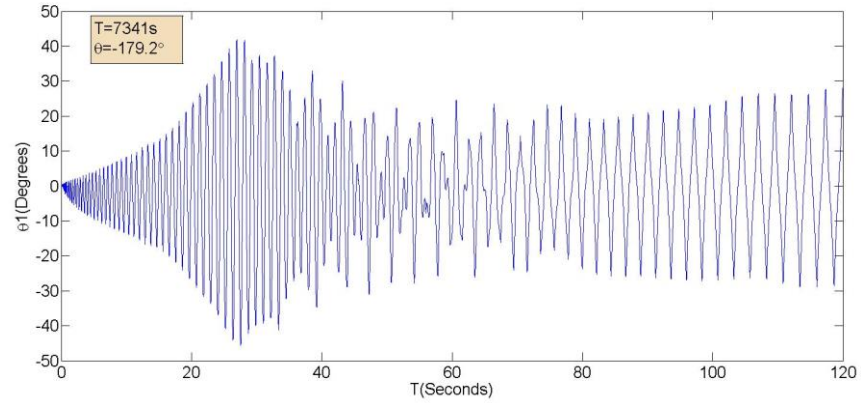


(c)

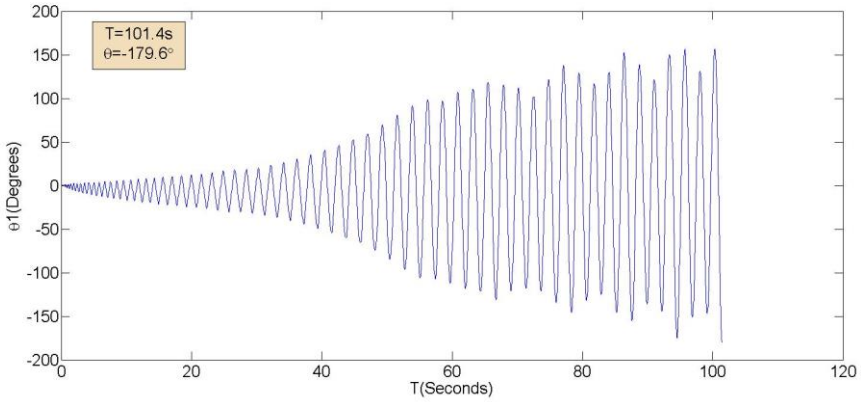
Figure 5.5: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6616, $\Delta\alpha_2$= 0.1699, $\Delta\delta$=0.05512 for (a) Experimental; (b) Mathematical Model; (c) ENN Model



(a)



(b)



(c)

Figure 5.6: Measured angular position Θ_1 at $\Delta\alpha_1$=0.6635, $\Delta\alpha_2$= 0.1827, $\Delta\delta$=0.05935 for (a) Experimental; (b) Mathematical Model; (c) ENN Model

The figures show that the ENN model displays swing-up characteristics that are comparable to the swing-up characteristics obtained from experimenting with the real system. For example, Figure 5.4 shows that the time taken for the ENN model to swing up to approximately 180° is 118.7 seconds, which is closer to the time taken by the experiment to achieve the same task, at 107.6 seconds, compared to 8499 seconds for the mathematical model. This comparison can be seen throughout Figure 5.5 and Figure 5.6.

Table 5.2 presents the Root Mean Square (RMS) error and the Mean Absolute (MA) error of both models compared with the actual experimental data. The ENN model (ENNM) obtains smaller values for both errors when compared to the Mathematical model (MM).

Table 5.2: Error Comparison

Control Signals			Root Mean Square Error (%)		Mean Absolute Error (%)	
$\Delta\alpha_1$ (V)	$\Delta\alpha_2$ (V)	$\Delta\delta$ (rad^{-1})	MM	ENNM	MM	ENNM
0.6924	0.1966	0.051129	57.4418	35.5183	41.3808	23.5414
0.6872	0.1726	0.054430	64.4605	42.4171	48.024	28.7436
0.6616	0.1699	0.05512	60.6531	40.6222	44.436	27.6083
0.6635	0.1827	0.05935	65.3856	32.3343	48.1372	21.9021

5.7 Discussion and Conclusion

The results show that the Robogymnast Elman neural network model provides a better representation of the actual system compared to the mathematical model. The ENN model is tested by comparing the system response output of the swing-up control motion of the Robot gymnast. The same input control parameters ($\Delta\alpha_1$,$\Delta\alpha_2$,$\Delta\delta$) are applied to the input voltages of both the mathematical model and the ENN model. The output angle of the first link θ_1 is then compared with that of the actual experimental output. The output of the ENN closely resembles the experimental output in terms of shape and amplitude. The difficulty of training the modelling is caused by the non-linearity of the system. The non-linearity of the system is caused by the input voltage being limited at ±10V, as shown in Figure 5.7. As the input voltages become limited at 10V, the swing angle Θ_1 still needs to increase to 180°. This makes the system non-linear, as the increment of Θ_1 is no longer proportional to the amplitude of the input voltages. The system's response is now more dependent on the change in the input signal frequency and the natural inertia of the system.

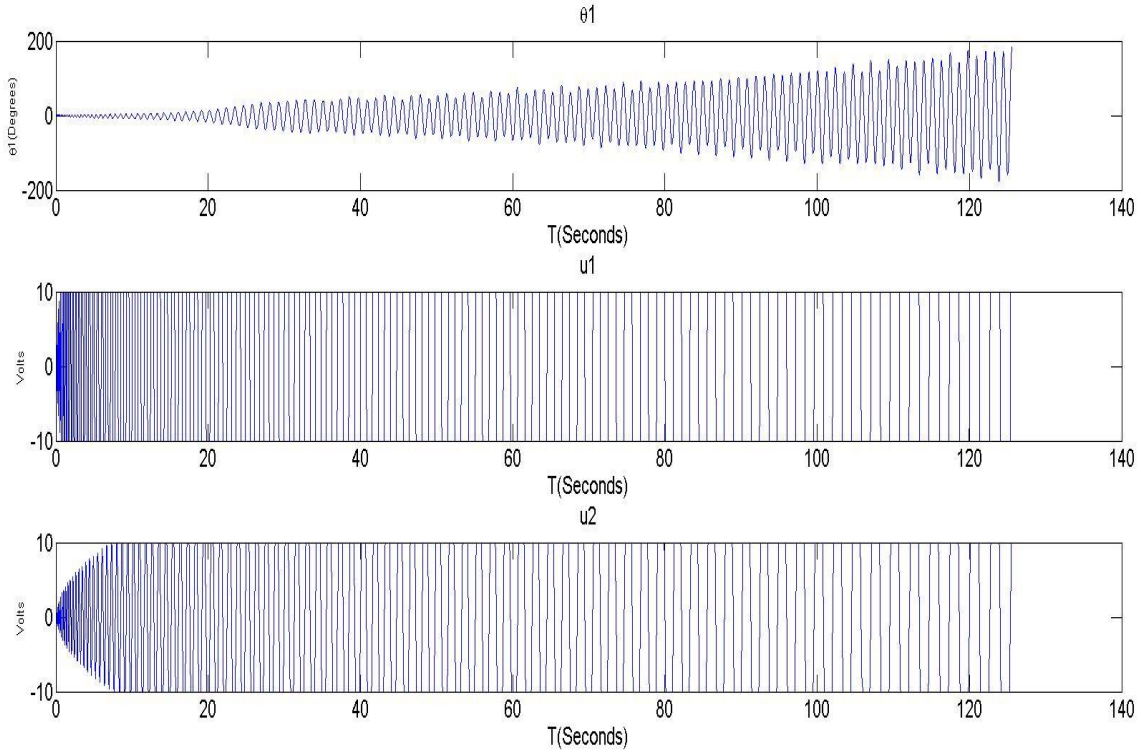


Figure 5.7: Output of Θ_1 with its input voltages u_1 and u_2

While the mathematical model is useful for studying the behaviour of the system, it does not provide an accurate representation of the actual behaviour of the system in its environment. The mathematical model does not take into consideration external factors such as air resistance and friction caused by wear. In order to successfully achieve the swing-up, the system needs to operate at its natural frequency. To do so requires the motors to begin rotating back and forth at high frequencies. At high frequencies, it becomes more difficult to calculate the system's response, as any slight disturbance will cause the system to behave differently. The ENN model was trained using data from the actual experiment, thus taking into consideration the effects that external factors might have on the system's behaviour. The ENN model provides a simple but useful alternative to the mathematical model and in this case improves on it.

5.8 Summary

A mathematical model is a representation of a system using mathematical equations and symbols. It is often used to describe a system and to study its behaviour. However, most mathematical models provide a useful but inaccurate representation of the actual system's response. A neural network model would provide a more accurate representation of the actual system's response because training is done using actual experimental data. This study focuses on modelling the response of the

Robogymnast. Due to the restrictions encountered by the mathematical model caused by the complex nature and nonlinearity of the Robogymnast, a novel approach of modelling the Robogymnast using neural networks was proposed in this chapter. A multi-layered Elman neural network model was used to represent the system. Inputs were applied to both the mathematical model and the neural network model and their outputs were compared and analyzed. In the following chapter, the controller design for inverted balancing of the Robogymnast will be discussed.

CHAPTER 6

Upright Balancing of the Robogymnast

6.1 Introduction

A number of researchers have studied the problem of stabilising inverted underactuated pendulums (Spong and Block 1995; Gawthrop and Wang 2006; Grossimon et al. 1996; Awtar et al. 2002). The balancing of a triple inverted pendulum is an important problem in robotics because it mimics the human body and its balancing mechanisms (Kamil et al. 2012). Brown and Passino (1997) developed intelligent controllers for balancing the acrobot by combining classical control, fuzzy and adaptive fuzzy controllers which swing, catch and balance the acrobot in an inverted position. A successful direct fuzzy balancing controller was then designed by emulating the action of the LQR. Wang et al. (2014) employed an improved Artificial Bee Colony (ABC) algorithm to optimize the performance of the LQR. The optimized LQR was then used to balance a circular-rail double inverted pendulum. Simulation results proved that the improved ABC has outperformed the original ABC, as the LQR controller with improved ABC achieved a much shorter settling time. Kamil et al. (2014) designed a Discrete-time Linear Quadratic Regulator (DLQR) to balance the Robogymnast. The DLQR

controller used is similar to the conventional LQR but with an 8-by-8 Q matrix instead of the usual 6-by-6 Q matrix typical of a triple link pendulum controller. The extra dimensions allow the DLQR to incorporate the angular accelerations of the first two links in determining the gain of the controller. This chapter presents two applications of IWO used to optimize the 6-by-6 Q-matrix of the LQR controller. The output of the optimization process is then tested and its performance analysed.

Section 6.2 explains the modifications that have to be made to the ENN model discussed in Chapter 5 in order to represent the Robogymnast in the upright position. Section 6.3 discusses the Linear Quadratic Regulator and its equations. Section 6.4 demonstrates the application of the IWO in LQR controller design. Section 6.5 presents the application of the cost function (J) as the fitness criterion. In Section 6.6, the application of settling time (T_{st}) as the fitness criterion is discussed. The discussion and conclusion are given in Section 6.7 and Section 6.8 provides a summary of the chapter.

6.2 Model of the Robogymnast in the upright position

In this chapter, the Robogymnast is regarded as a triple link pendulum in an unstable upright configuration, as shown in Figure 6.1. In order to represent the system in this configuration, the matrix W_{cx} in Equation 5.1 is expressed as follows:

$$W_{cx} = \begin{bmatrix} 0.99 & -2.43e^{-3} & -2.34e^{-4} & 2.49e^{-2} & 1.01e^{-2} & 1.20e^{-3} \\ 1.49e^{-3} & 0.99 & -2.72e^{-4} & 3.78e^{-5} & 5.88e^{-3} & 2.15e^{-4} \\ 2.55e^{-4} & -2.22e^{-4} & 1.00 & 5.30e^{-6} & 2.15e^{-4} & 4.95e^{-3} \\ \mathbf{0.77} & \mathbf{0.23} & \mathbf{2.39e^{-2}} & 0.99 & 0.52 & 6.37e^{-2} \\ \mathbf{-7.59e^{-2}} & \mathbf{0.13} & \mathbf{1.43e^{-2}} & 2.64e^{-3} & 1.55e^{-2} & 2.01e^{-3} \\ \mathbf{-1.32e^{-2}} & \mathbf{1.21e^{-2}} & \mathbf{2.85e^{-2}} & 3.97e^{-4} & 2.05e^{-3} & 6.55e^{-3} \end{bmatrix}$$

Where the elements highlighted in the box have their polarities reversed compared to Equation 5.1.

Matrices W_{ho} and W_{ih} remained unchanged as in Chapter 5.

6.3 Linear Quadratic Regulator

The linear quadratic regulator (LQR) is a well-known design technique that provides practical feedback gains. It is a multivariable controller, as it can control displacement of the angles of the triple inverted pendulum at the same time (Sehgal and Tiwari 2012). Extensive research in the control field has shown on multiple occasions that LQR is well suited for inverted pendulum stabilization (Lee and

Perkins 2008). The objective of LQR is to find the minimum value of the following cost function:

$$J = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \quad (6.1)$$

Where u(t) is unconstrained, Q is required to be a symmetric, positive semi-definite matrix and R is required to be a symmetric positive definite matrix. x represents the states of the system and u represents the control signals.

For LQR, the input will be as follows:

$$u(t) = -Fx(t) \quad (6.2)$$

where F is the gain matrix required by the LQR. By applying Equation 6.2 into the state space equation, the following equation will emerge:

$$\dot{x} = (A - BF)x \quad (6.3)$$

To obtain the value of F, the following equation is then applied:

$$F = R^{-1}B^T P \quad (6.4)$$

Using the Algebraic Riccati Equation below, the value of P can be obtained:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \quad (6.5)$$

The value of F can then be obtained from Equation 6.4.

In order to implement an LQR controller, one must select suitable weighing matrices. For the Robogymnast, the value of Q will penalize the states, while the value of R will penalize the inputs. For this reason, the elements of the Q matrix were selected to be much larger than the elements of the R matrix.

6.4 Application of IWO in LQR controller design

The IWO is applied to find the global optimal solution for the LQR controller in order to minimize the settling time and voltage required for the Robogymnast to go from an unbalanced inverted configuration to a balanced upright configuration. Q and R are set as diagonal matrices:

$$Q = \begin{bmatrix} Q1 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q2 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q3 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q4 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q5 & 0 \\ 0 & 0 & 0 & 0 & 0 & Q6 \end{bmatrix}; \qquad R = \begin{bmatrix} R1 & 0 \\ 0 & R2 \end{bmatrix}$$

For the optimization process, the parameters R1 and R2 of the LQR controller are set at 1 and the values of Q are to be optimized. This is because, for this application, more weight is put on the control of the states than the inputs. In order to ensure that the Q matrix is a symmetric, positive semi-definite matrix, Q is set as:

$$Q = Q_{seeds} x\, Q_{seeds}^{T} \qquad (6.6)$$

Where Q_{seeds} is a diagonal matrix consisting of IWO seeds (S1, S2, S3, S4, S5, S6) and Q_{seeds}^{T} is its transposed matrix:

$$Q_{seeds} = \begin{bmatrix} S1 & 0 & 0 & 0 & 0 & 0 \\ 0 & S2 & 0 & 0 & 0 & 0 \\ 0 & 0 & S3 & 0 & 0 & 0 \\ 0 & 0 & 0 & S4 & 0 & 0 \\ 0 & 0 & 0 & 0 & S5 & 0 \\ 0 & 0 & 0 & 0 & 0 & S6 \end{bmatrix}$$

The optimization is applied for an initial deflection of absolute angles θ_1=3°, θ_2=3°, θ_3=3°. This is the estimated maximum deflection angle that the Robogymnast can make before the system becomes incapable of bringing it back to a balanced upright configuration. The objective of the controller is to obtain a relative angle of $q_1 \leq 0.001$ rad, $q_2 \leq 0.001$ rad and $q_3 \leq 0.001$ rad. Where:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 - \theta_1 \\ \theta_3 - \theta_2 \end{bmatrix} \qquad (6.7)$$

6.5 LQR controller designed using cost function (J) as the fitness criterion

This section presents descriptions and analysis of the proposed optimized LQR controller using the IWO and cost function J as the fitness criterion. Optimization is achieved by finding the minimum value of J. This is the fitness criterion used by most previous researchers (Asadi et al. 2016; Souza and Bigot 2016.). As seen in Equation 6.1, J is dependent on the sum of the states and control signal multiplied by their respective weights. This shows that the smaller the value of J, the more efficient the LQR controller will be. The parameters of the IWO procedure are as shown in Table 6.1.

Table 6. 1: IWO parameters with J as the fitness criterion

Variable	Value	Description
Number of initial plants (p_{init})	5	Number of randomly chosen values from the solution space.
Minimum number of seed sets (S_{min})	1	Minimum population of solutions
Maximum number of seed sets (S_{max})	500	Maximum population of solutions
Initial value of standard deviation ($\sigma_{initial}$)	0.1	Standard deviation used for spatial distribution of plants.
Final value of standard deviation (σ_{final})	0.01	Final standard deviation used for spatial distribution of plants.
Maximum number of iterations ($Iter_{max}$)	10	Number of iterations
Nonlinear Modulation Index	0.01	-
Target angle	q_1<0.001 rad q_2<0.001 rad q_3<0.001 rad	The angle where time is recorded and used as the fitness criterion
Search range	0-3000	Search range used based on trial and error.

A seed set is a combination of six seeds that make up S1, S2, S3, S4, S5 and S6. The number of maximum seed sets is 500. This is to ensure that the number of seeds is not so large as to slow the search time. The maximum number of iterations is set as 10. After a number of trials, it is found that a larger number of iterations would not contribute any improvement to the search process. The target angle is set at 0.001 rad, which is close enough to be considered stable and inverted. The search range is set at 0-3000 based on trial and error. It is found that the output of the LQR is more dependent on the ratio of the diagonal values of the Q matrix with respect to each other rather than the magnitude of each individual Q value.

Table 6.2: IWO Results using the cost function J as the fitness criterion

Fitness Rank	S1	S2	S3	S4	S5	S6	Settling time, T_{st} (s)	J^*
1	500.677	150.253	500.310	250.035	50.040	0.000	8.88	1008.639
2	150.307	100.439	250.916	300.783	150.435	200.882	26.35	1760.094
3	50.061	500.107	400.436	250.645	150.673	100.191	6.38	2242.47
4	500.348	150.587	300.658	500.002	100.174	150.002	16.80	3157.548
5	200.348	400.587	350.658	300.002	150.174	450.002	11.75	3302.033
6	250.348	300.587	150.658	450.002	100.174	450.002	25.38	3793.35
7	50.061	200.107	200.436	250.645	400.673	100.191	27.38	4612.892
8	150.061	350.107	250.436	100.645	400.673	350.191	15.68	5455.54
9	500.307	150.439	200.916	300.783	400.435	300.882	22.58	5495.975
10	990.407	297.172	990.041	494.901	99.013	391.785	9.300	5501.419

* Fitness Criterion

Table 6.2 shows the top ten best seed sets obtained from a population of 500. The minimum J obtained is 1008.639. All values obtained are well within the search range previously set during the optimization process. It can be seen that the time for the Robogymnast to achieve a stable inverted configuration is not proportional to the values of J. The next subsection will present the simulation results when the values obtained from IWO results were applied to the LQR controller of the Robogymnast.

6.5.1 Simulation results of LQR designed using IWO with cost function J as the fitness criterion

The fittest seeds, which are S1=500.677, S2=150.253, S3=500.310, S4=250.035, S5=50.040 and S6=0.000, are selected for analysis. Using Equation 6.6, the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 2.5068e^5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2258e^5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.5031e^5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6252e^5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0250e^5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -0.4430e^3 & -0.1873e^3 & -0.0358e^3 & 0.0803e^3 & 0.0423e^3 & 0.0053e^3 \\ -1.3437e^3 & -0.5900e^3 & 0.1729e^3 & 0.2528e^3 & 0.1330e^3 & 0.0153e^3 \end{matrix}$$

In order to verify the effectiveness of the IWO algorithm, the parameters obtained are applied to a Matlab® program created by the author. The results are then compared

for three different configurations (Figure 6.1) in order to ensure that the optimization can be implemented in various configurations.

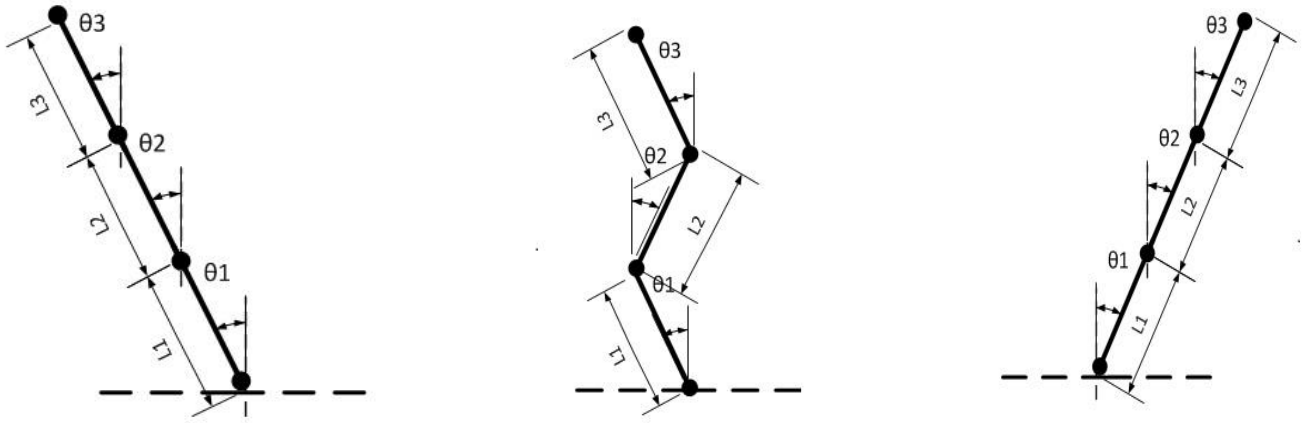


Figure 6.1: Configurations of Robogymnast (a) θ_1=-3°; θ_2=-3°; θ_3=-3°, (b) θ_1=-3°; θ_2=3°; θ_3=-3°, (c) θ_1=3°; θ_2=3°; θ_3=3°

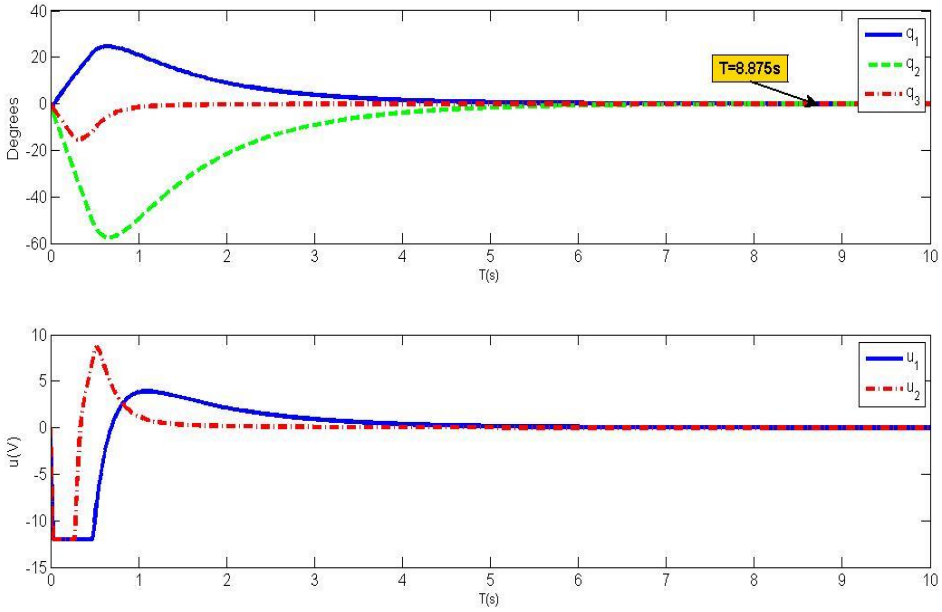


Figure 6.2: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

Figure 6.2 illustrates the controlled system response and the voltages when the Robogymnast is in the upright position with the initial absolute angular position equal to [-3°, -3°, -3°] (Figure 6.1(a)). It is clear that the designed controller was able to stabilise the system and converge to the set values. The maximum voltage for motor 1 (u_1) and motor 2 (u_2) are both shown as -12V. It can be seen that the time taken to reach a stable upright position is 8.875 seconds.

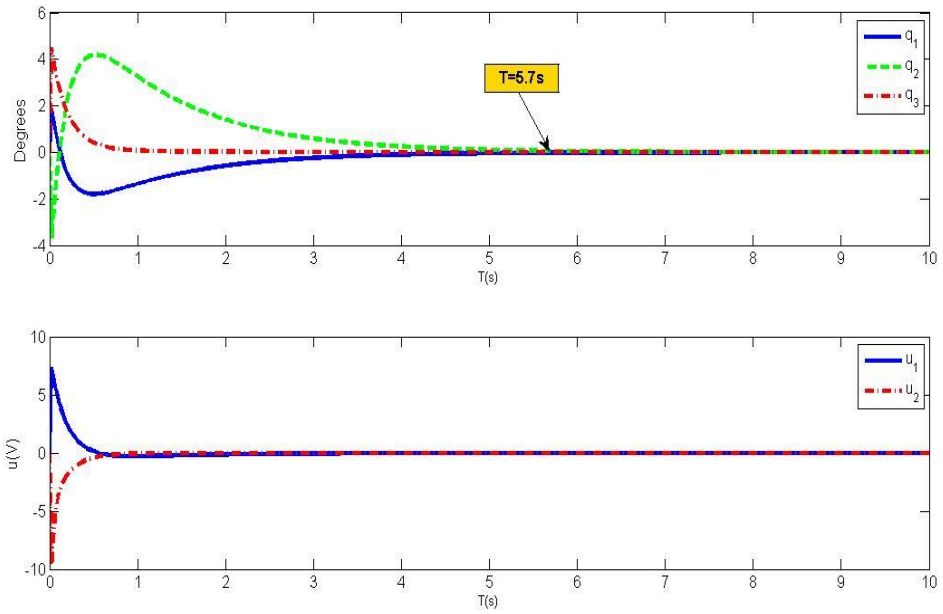


Figure 6.3: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°

The system response when the Robogymnast is in the upright position equal to [-3°, 3°, -3°] (Figure 6.1(b)) is displayed in Figure 6.3. The time taken for the system to stabilise is 5.7 seconds, while the maximum control actions of motor 1(u_1) and motor 2 (u_2) are 7.32V and -9.54V respectively.

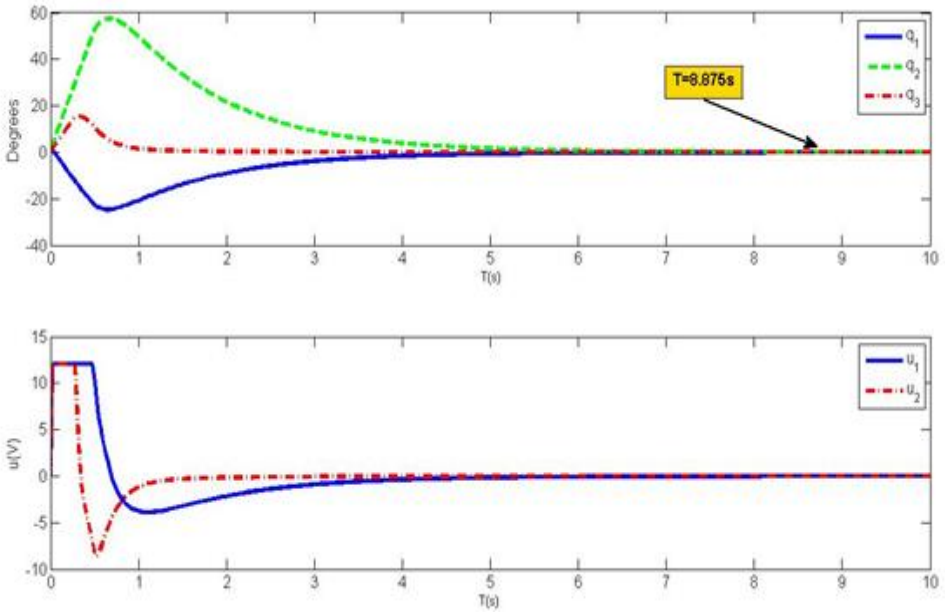


Figure 6.4: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3°

Figure 6.4 shows the reaction time and control effort when the initial absolute angular position of the Robogymnast is equal to [3°, 3°, 3°] (Figure 6.1(c)). It can be seen that the response is similar to Figure 6.2 but in the opposite direction.

To further verify the effectiveness of the designed LQR controller, an external disturbance of 0.05 rad was applied to each of the Robogymnast links one at a time and its reaction was observed. The disturbance was applied about two seconds after the controller attempted to stabilize the system from an initial absolute angular position equal to [1.5°, 1.5°, 1.5°]. The objective of this test was to determine the robustness of the LQR controller using the parameters obtained using IWO.

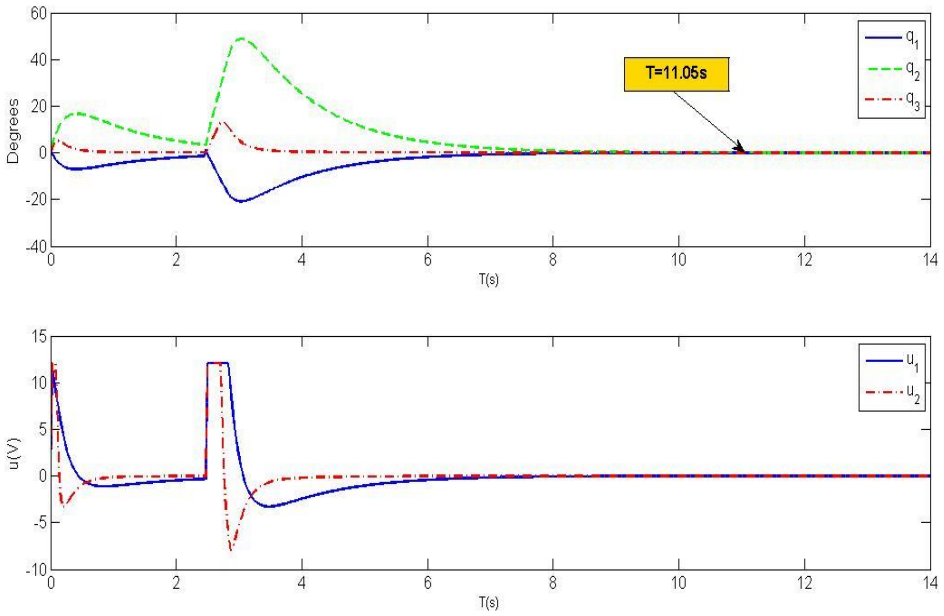


Figure 6.5: Disturbance to Link 1

From Figure 6.5, it can be seen that the system experiences a large displacement when a disturbance is applied to the first link. However, despite this, the controller is still able to balance the Robogymnast successfully.

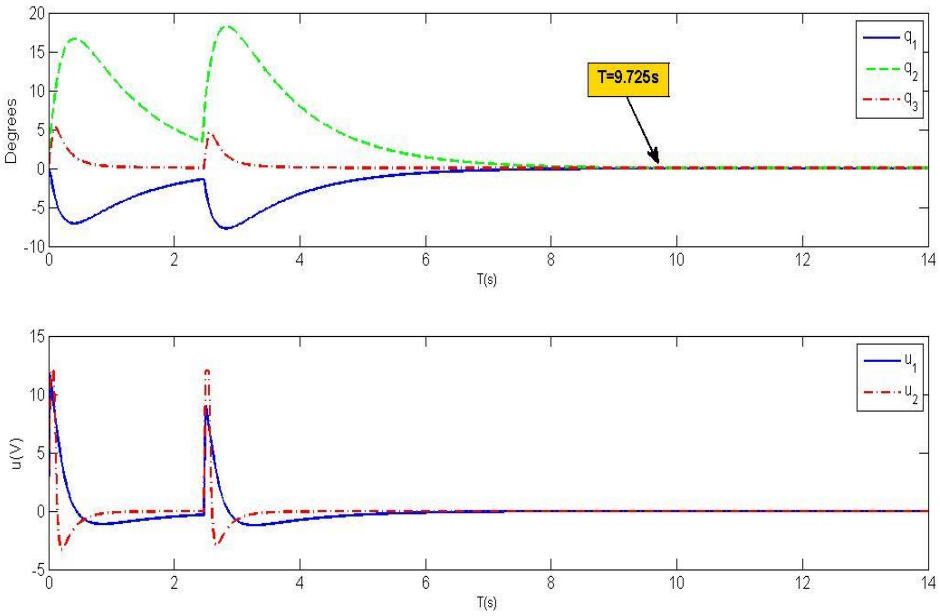


Figure 6.6: Disturbance to Link 2

Figure 6.6 illustrates the reaction of the system when a disturbance is applied to the second link. The displacement caused by the disturbance is smaller compared to Figure 6.5. The voltage requirements for both motors are also visibly smaller.

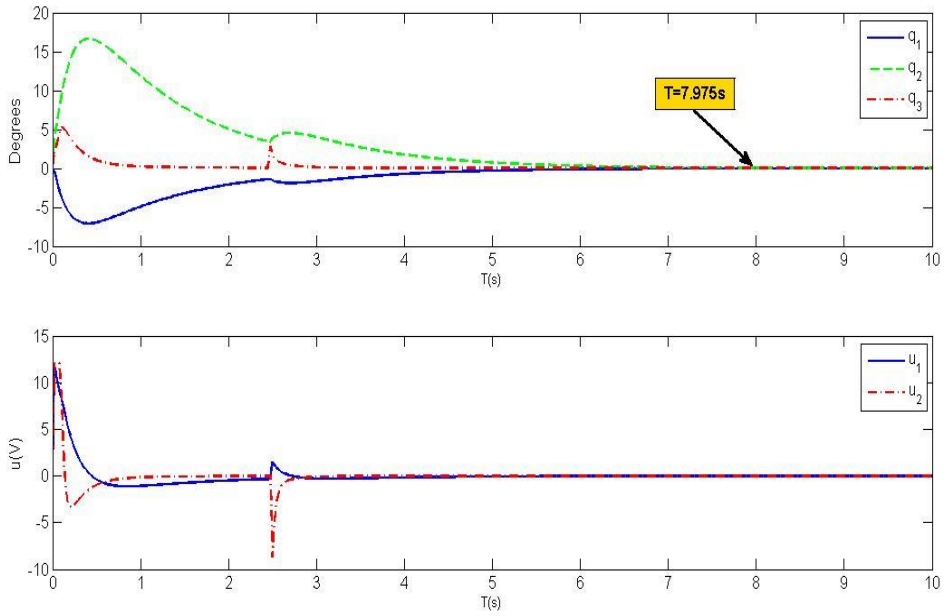


Figure 6.7: Disturbance to Link 3

Figure 6.7 represents the reaction of the system when a disturbance is applied to the third link. The displacement in this figure is far less severe when compared to Figure 6.5 and Figure 6.6. It can also be seen that when the disturbance is applied, u_2 is significantly larger than u_1. This indicates that most of the work is done by motor 2.

6.6 LQR controller designed using time (T_{st}) as the fitness criterion

The result in Table 6.2 shows that the value of the cost function J is not proportional to the settling time (T_{st}) of the Robogymnast. This section proposes an LQR controller where the diagonal values of the Q matrix are selected using IWO with T as the fitness criterion. The optimized parameters of the LQR controller are selected based on the minimum T_{st} value. The IWO parameters used in this procedure are as in Table 6.1.

Table 6.3 shows the top ten best seed sets obtained from a population of 500. The fastest settling time obtained was 5.10 seconds, while the slowest was 5.83 seconds, within a population of 500 seeds. All values obtained are within the search range previously set during the optimization process. Similar to the results shown in Table 6.2, the time for the Robogymnast to achieve a stable inverted configuration is not proportional to the values of J. The next subsection will present the simulation results when the values obtained from the IWO results were applied to the LQR controller of the Robogymnast.

Table 6.3: IWO Results using Time (T_{st}) as the fitness criterion

Fitness Rank	S1	S2	S3	S4	S5	S6	Settling time, T_{st} (s)‡	J
1	1292.593	745.061	995.334	399.673	149.737	247.772	5.10	672.493
2	1786.046	1090.479	1191.917	548.108	298.172	495.097	5.50	1557.780
3	899.211	649.188	849.557	300.782	200.146	99.645	5.55	423.515
4	1937.840	1537.127	1341.448	598.776	496.416	594.183	5.60	2546.645
5	1387.852	1088.966	1045.013	496.238	346.738	148.509	5.68	1149.552
6	1243.427	1493.460	1642.765	648.759	399.046	647.979	5.70	2367.704
7	1927.856	2327.026	1336.946	894.068	743.335	346.987	5.71	4463.318
8	1735.460	1936.290	1839.579	845.573	546.656	697.183	5.73	3759.826
9	1941.855	1839.636	1642.401	847.866	547.446	496.372	5.75	3197.372
10	1991.059	2134.856	2036.027	995.476	596.649	791.592	5.83	4771.564

‡ Fitness Criterion

6.6.1 Simulation results of LQR designed using IWO with T_{st} as the fitness criterion

The fittest seeds, which are S1 = 1292.593, S2 = 745.061, S3 = 399.673, S4 = 149.737, S5 = 149.737 and S6 = 247.772, are selected for analysis. Using Equation 6.6, the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 16.708e^5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.551e^5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.907e^5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.597e^5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.224e^5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.614e^5 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -0.5790e^3 & -0.2504e^3 & -0.0306e^3 & 0.1057e^3 & 0.0556e^3 & 0.0070e^3 \\ -0.1201e^3 & -0.055e^3 & 0.0234e^3 & 0.0222e^3 & 0.0117e^3 & 0.0014e^3 \end{matrix}$$

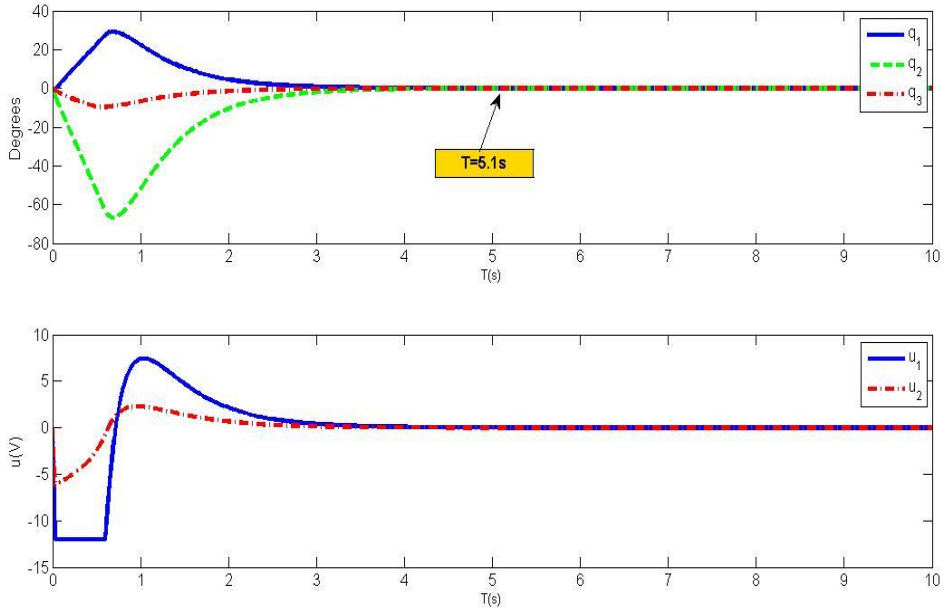


Figure 6.8: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

Figure 6.8 clearly illustrates the controller's ability to stabilise the Robogymnast when it is in the upright position with the initial absolute angular position equal to [-3°, -3°, -3°] (Figure 6.1 (a)). The maximum voltage for motor 1 (u_1) is 12 volts, and for motor 2 (u_2) is 6.290 volts. It can be seen that the time taken to reach a stable upright position is 5.1 seconds.

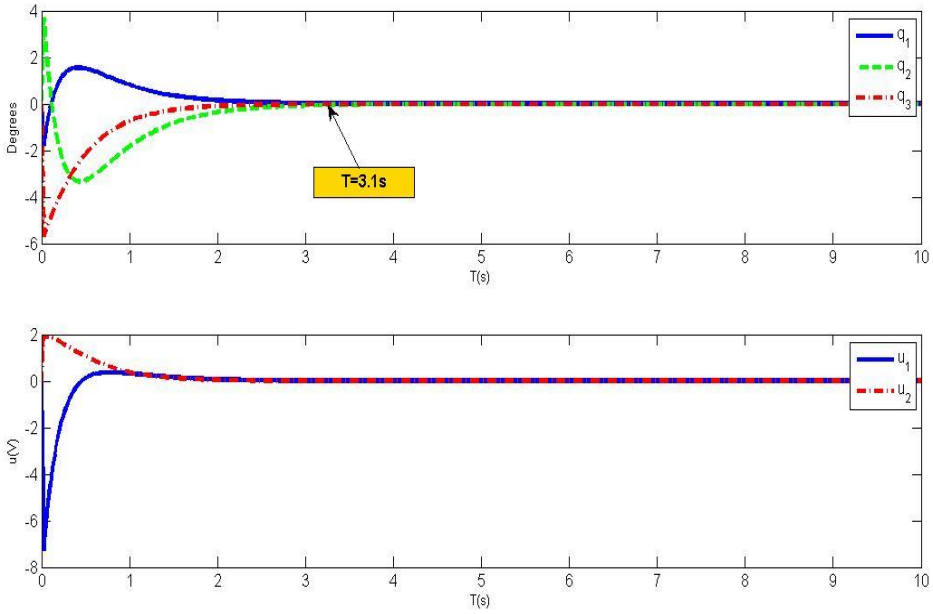


Figure 6.9: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°

Figure 6.9 shows the response of the system when the initial absolute angular position is equal to [-3°, 3°, -3°] (Figure 6.1(b)). The time taken for the system to stabilize is 3.10 seconds. The maximum voltage is -7.296 volts for motor 1 (u_1) and 1.927 volts for motor 2 (u_2).

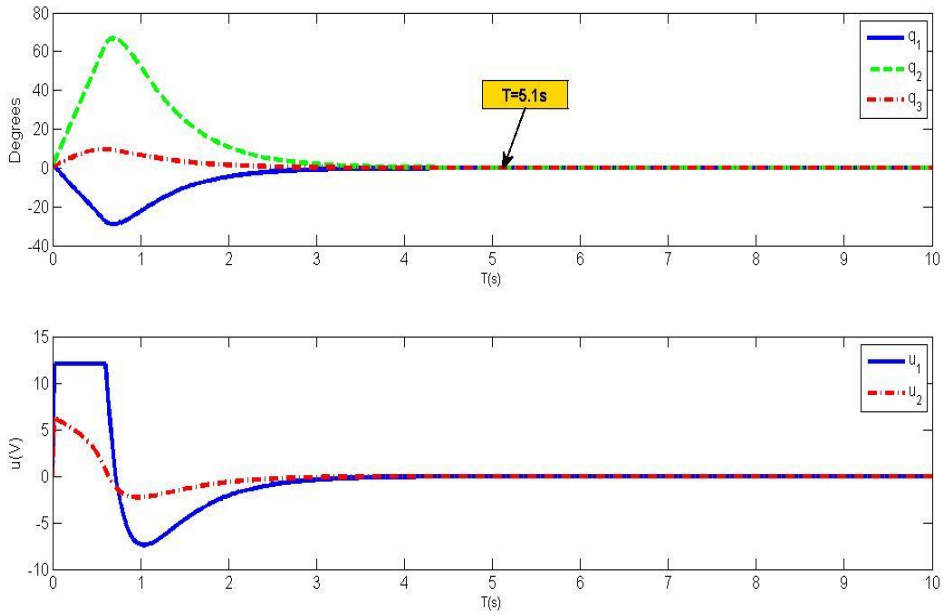


Figure 6.10: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3°

Finally, Figure 6.10 displays the reaction of the system when the initial absolute angular position is equal to [3°, 3°, 3°] (Figure 6.1(c)). The reaction is similar to Figure 6.8 but in the opposite direction.

As in section 6.5, an external disturbance was applied to each of the Robogymnast links one at a time and its reaction observed. The disturbance is applied about two seconds after the controller attempts to stabilize the system from an initial absolute angular position equal to [1.5°, 1.5°,1.5°].

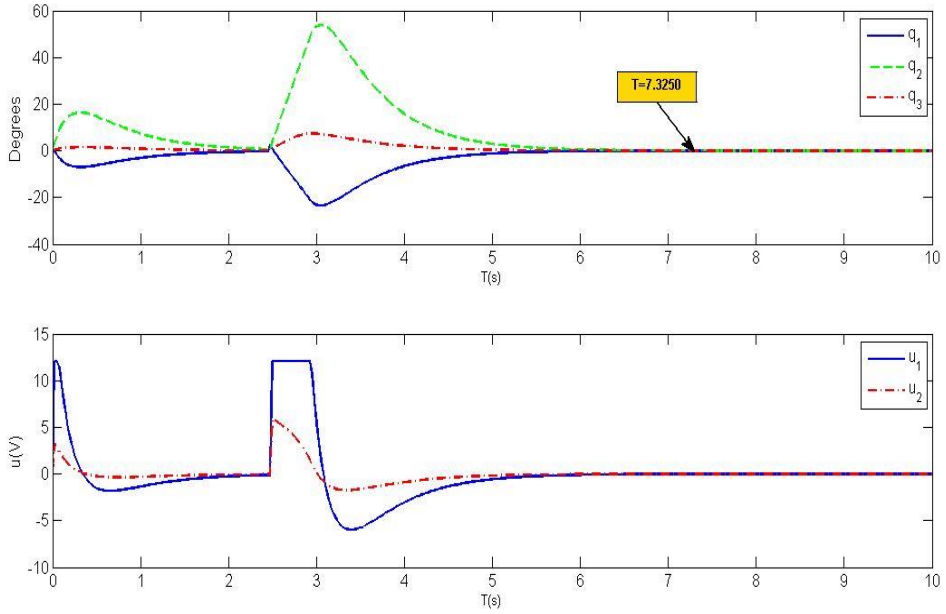


Figure 6.11: Disturbance to Link 1

Figure 6.11 illustrates that when a disturbance is applied to the first link, the controller quickly reacts to counter the displacement. The figure also reveals that u_1 is larger than u_2, indicating that most of the work is done by motor 1.

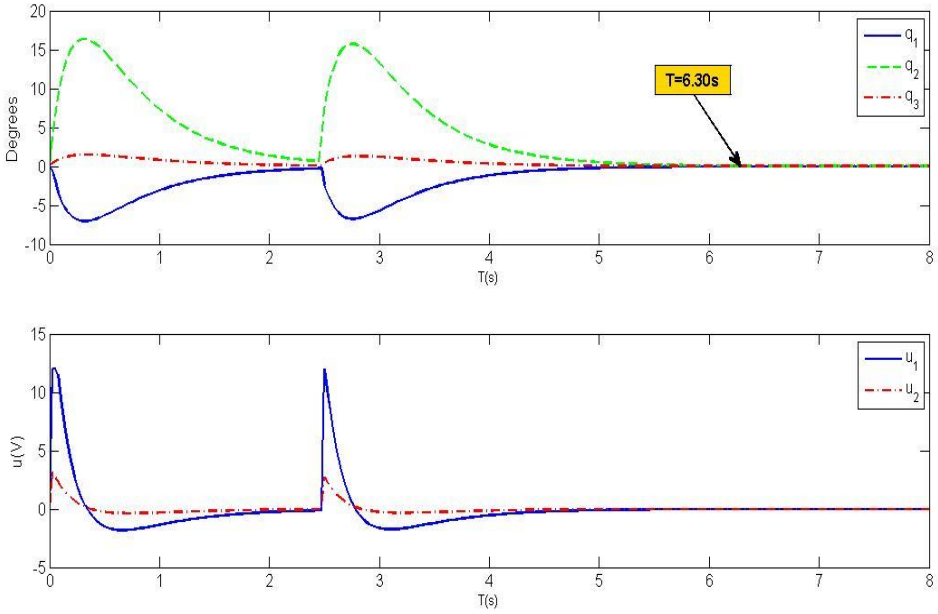


Figure 6.12: Disturbance to Link 2

Figure 6.12 shows the reaction of the system when a disturbance is applied to the second link. The displacement is not as severe as in Figure 6.11. The maximum voltage applied to motor 1 is still 12 volts, but its peak duration is about 0.4 seconds shorter than in Figure 6.11.

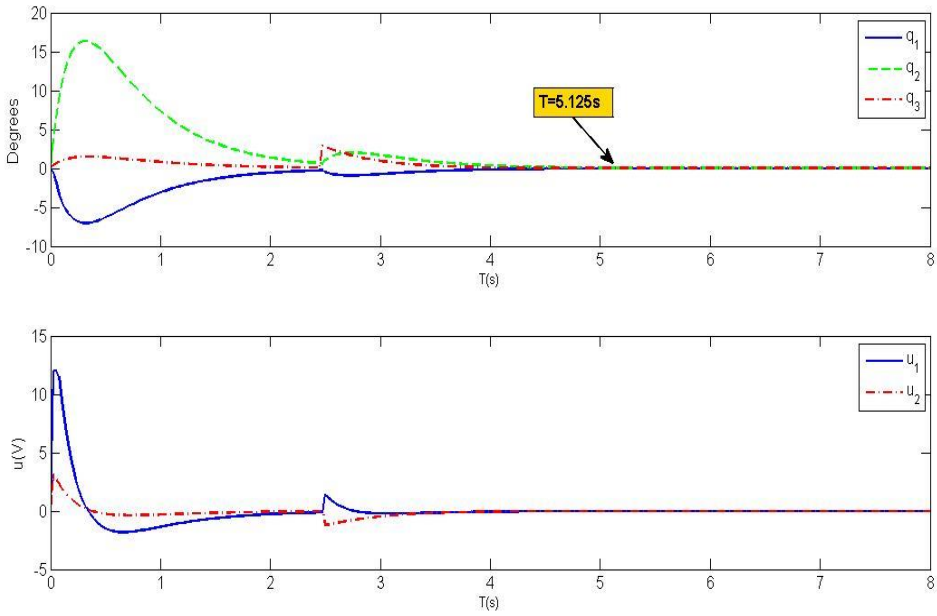


Figure 6.13: Disturbance to Link 3

Figure 6.13 shows the effect that a disturbance on the third link has on the entire system. The displacement is small and the voltage requirements are minimal.

6.7 Discussion and conclusion

In the case of inverted balancing of the Robogymnast, it would appear that the LQR controller designed using parameters obtained by both methods can successfully bring the Robogymnast to an inverted and stable configuration. Using IWO with time (T_{st}) as the fitness criterion yields parameters that lead to a controller (LQRT) with a faster reaction time compared to the controller with parameters obtained

using IWO with J as the fitness criterion (LQRJ). Another distinguishable difference is that the required voltage for motor 2 (u_2) is lower for LQRT. This condition is consistent throughout all of the three configurations used in the simulation. The results also reveal that most of the work is done by motor 1, thus resulting in a higher voltage u_1. In order to further analyse the performance of both controllers, more tests had to be done. Table 6.4 compares the performance of the two controllers in different initial angular configurations.

Table 6. 4: Comparison of the performance of the LQRJ and LQRT in different initial angular configurations.

Deflection Angle	Controller	Jsum	Tmax(s)	$u_1$max(V)	$u_2$max(V)	Description/Purpose
θ_1=1°; θ_2=1°; θ_3=1	LQRJ	605640	6.700	7.732	12.000	To examine controller's reaction at low deflection angles.
	LQRT	2725800	3.625	10.105	2.097	
θ_1=1°; θ_2=-1°; θ_3=1°	LQRJ	36148	4.425	2.441	3.180	To examine controller's reaction at low deflection angles.
	LQRT	144650	2.500	2.432	0.642	
θ_1=3°; θ_2=3°; θ_3=3°	LQRJ	9997600	8.875	12.000	12.000	Figure 6.4 and 6.10
	LQRT	67249000	5.100	12.000	6.290	
θ_1=3°; θ_2=-3°; θ_3=3°	LQRJ	325330	5.700	7.322	9.541	Opposite of Figure 6.3 and 6.9
	LQRT	1301800	3.100	7.296	1.927	
θ_1=3.1°; θ_2=3.1°; θ_3=3.1°	LQRJ	12677000	9.125	12.000	12.000	To examine the controller's reaction when slightly higher deflection angle is applied.
	LQRT	98407000	5.350	12.000	6.499	
θ_1=0°; θ_2=4°; θ_3=5°	LQRJ	2020200	7.550	12.000	12.000	To examine the controller's reaction when higher deflection angles are applied to link 2 and link 3.
	LQRT	10114000	4.100	12.000	3.432	
θ_1=4°; θ_2=0°; θ_3=0°	LQRJ	3482200	7.800	12.000	12.000	To examine the controller's reaction when higher deflection angles are applied to link 1.
	LQRT	16846000	4.325	12.000	4.5471	
θ_1=5.45°; θ_2=0°; θ_3=0°	LQRJ	15839000	9.375	12.000	12.000	To test maximum deflection angle the controller can recover from.
	LQRT	201070000	5.600	12.000	6.195	
θ_1=5.65°; θ_2=0°; θ_3=0	LQRJ	46030000	10.700	12.000	12.000	To test maximum deflection angle the controller can recover from
	LQRT	Inf	Inf	12.000	12.000	
θ_1=5.7°; θ_2=0°; θ_3=0°	LQRJ	Inf	Inf	12.000	12.000	To test maximum deflection angle the controller can recover from.
	LQRT	Inf	Inf	12.000	12.000	

Table 6.4 reveals that controller LQRT performs consistently faster (43%-46% faster). However, controller LQRJ is capable of stabilizing the Robogymnast with a larger initial absolute angular position, as seen in Table 6.4. Voltage u_1 for LQRJ is lower when all three initial absolute angular positions have the same polarity and LQRT requires significantly less voltage for motor 2. J is smaller due to the smaller state space values, while T_{st} has large state values, but these values settle in a shorter amount of time. Finally, both controllers show that J and T_{st} are not always proportional to each other.

6.8 Summary

The purpose of this chapter was to determine the best fitness criterion to be used when designing the LQR controller for the Robogymnast. The first fitness criterion is the J cost function. The second fitness criterion is the settling time (T_{st}) required to stabilize the Robogymnast. The fitness criteria were employed on the IWO and used to obtain optimum diagonal values of the Q matrix. Using the Q parameters obtained, two LQR controllers were designed and tested using simulation. According to the simulations, the Robogymnast is able to recover its balance from an initial unbalanced configuration. The controllers also prove to be robust enough to perform even when a slight disturbance is applied. The LQRT controllers has a faster reaction time, but LQRJ is capable of recovering from a larger angular position. LQRT also provides better efficiency in motor 2, thus reducing the

required voltage when balancing the Robogymnast. In the next chapter, a hybrid J and T_{st} fitness criterion is proposed to get the best of both controllers.

CHAPTER 7

Multi-Objective Invasive Weed Optimization of the LQR Controller

7.1 Introduction

Multi-objective optimization has become an important part of optimization activities. Many real-world optimization problems are naturally posed as nonlinear programming problems having multiple conflicting objectives. A multi-objective optimization problem deals with more than one objective function (Deb 2014). Liu et al. (2008) designed an approach for weighting matrices for LQR based on a multi-objective evolution algorithm. The algorithm uses the J function and pole placement as the objective function. Simulation results show that a shorter adjustment time and smaller amplitude value deviating from the steady state are achieved using the proposed approach. An optimal design of LQR weighting matrices based on

intelligent optimization methods such as Genetic Algorithms (GA), PSO, Differential Evolution (DE) and the Imperialist Competitive Algorithm (ICA) to solve the optimization problem of LQR for a robot manipulator was proposed by Ghoreishi et al. (2011). All results were compared by combining criteria like speed of response, the closed-loop pole locations and maximum level of control effort into an objective function to find the best weighting matrices in the LQR controller. An optimal trade-off design for a fractional order (FO)-PID controller is proposed with a Linear Quadratic Regulator (LQR)-based technique using two conflicting time domain objectives (Das et al. 2015). The research deals with problems such as choosing optimal weights and time delays in the LQR formulation. Khalaf et al. (2015) utilized Multi-Objective Invasive Weed Optimization (MOIWO) to design an impedance controller for a prosthesis test robot. The criteria for this optimization problem are the required amount of force and motion tracking. Simulation results showed that the solutions that were designed for motion tracking performed this task perfectly but failed to reproduce the desired forces, while the solution that was designed for force tracking deviated from the desired motion in order to produce the desired force.

In this chapter, the diagonal values of the LQR Q matrix are selected using modified IWO algorithms. The first technique is the Weighted Criteria Method IWO (WCMIWO), which combines the values of the cost function (J) and settling time (T_{st}) into a single fitness criterion with the help of weights. The second technique

is the Fuzzy Logic IWO Hybrid (FLIWOH), which analyses the values of J and T_{st}. These two values are then evaluated and assigned a membership value, which will subsequently be used as the fitness criterion. The performance of the two techniques are then compared and analysed. The performance of the resulting controllers will also be analysed with and without disturbance applied to the system. The criteria that will be used to evaluate the controllers are the settling time, input voltages, the maximum angular deflection from which they can recover and ability to maintain an upright position with disturbance applied to the system. The chapter is organized as follows. In Section 7.2, Multi-Objective Optimization (MOO) and the various types of MOO methods are briefly discussed. The following section describes the WCMIWO and its results. This is followed by the description of the FLIWOH and its results. In Section 7.5, the previous methods are repeated with disturbance applied to the system. The findings and results are discussed in section 7.6. Finally, a summary of the chapter is provided in Section 7.7.

7.2 Weighted Criteria Method Invasive Weed Optimization

The WCMIWO technique uses both J and T_{st} in determining the fitness of each set of seeds. The fitness criterion JT is calculated using equation (7.3):

$$JT = \left(W_J \, x \, J\right) + \left(W_T \, x \, T_{st}\right) \tag{7.3}$$

where W_J and W_T are the multiplied weights of J and T_{st} respectively, whose values are selected through trial and error. The weights are necessary due to J being significantly larger than T_{st}, to ensure that J does not dominate the resulting fitness criterion JT. The set seeds are arranged in ascending order with the smallest value of JT as the fittest set seeds. Table 7.1 shows the parameters of the WCMIWO process. The number of maximum seed sets is again selected to be 500. The maximum number of iterations is set at 10. The target angle is again set at 0.001 rad, which is close enough to be considered stable and inverted. Table 7.2 shows the top ten best seed sets obtained from a population of 500. The minimum J value obtained is $100.183e^5$ and the fastest time is 6.35 seconds.

Table 7.1: WCMIWO parameters

Variable	Value	Description
Number of initial plants (p_{init})	5	Number of randomly chosen values from the solution space.
Minimum number of seed sets (S_{min})	1	Minimum population of solutions
Maximum number of seed sets (S_{max})	500	Maximum population of solutions
Initial value of standard deviation ($\sigma_{initial}$)	50	Standard deviation used for spatial distribution of plants.
Final value of standard deviation (σ_{final})	0.5	Final standard deviation used for spatial distribution of plants.
Maximum number of iterations ($Iter_{max}$)	10	Number of iterations
Nonlinear Modulation Index	0.01	-
Weight of J (W_J)	$1e^{-6}$	Weightage of J
Weight of T (W_T)	10	Weightage of T_{st}
Target angle	q1<0.001 rad q2<0.001 rad q3<0.001 rad	The angle where time is recorded and used as the fitness criteria
Search range	0-1000	Search range used based on trial and error.

Table 7.2: WCMIWO Results

Fitness Rank	S1	S2	S3	S4	S5	S6	Duration to reach the upright position, T_{st} (s)	$J \times 10^5$	JT*
1	50.348	500.587	400.658	250.002	150.174	100.002	6.35	222.964	85.80
2	500.682	150.726	500.307	250.176	50.385	0.000	8.88	100.183	98.77
3	200.307	400.439	350.916	300.782	150.435	450.882	11.78	329.936	150.74
4	899.160	649.006	649.913	550.639	250.005	250.596	8.06	747.046	155.45
5	540.079	647.506	890.389	494.867	199.148	491.786	8.00	777.002	157.70
6	543.478	648.526	893.789	496.567	199.487	494.506	8.00	781.639	158.16
7	545.477	649.126	895.788	497.567	199.687	496.106	8.03	784.371	158.69
8	546.901	649.553	897.212	498.278	199.830	497.244	8.03	786.319	158.88
9	548.008	649.885	898.319	498.832	199.940	498.130	8.03	787.835	159.03
10	548.914	650.157	899.225	499.285	200.031	498.855	8.03	789.077	159.16

* Fitness Criterion

7.2.1 Simulation results of LQR designed using WCMIWO

The fittest seeds, which are S1=50.348, S2=500.587, S3=400.658, S4=250.002, S5=150.174 and S6=100.002, are selected for analysis. Using Equation 6.6, the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 2.5068e^5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.5059e^5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.6053e^5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6250e^5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2255e^5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1000e^5 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -0.5080e^3 & -0.2172e^3 & -0.0268e^3 & 0.0928e^3 & 0.0489e^3 & 0.0061e^3 \\ -0.4273e^3 & -0.1918e^3 & 0.0071e^3 & 0.0786e^3 & 0.0414e^3 & 0.0050e^3 \end{matrix}$$

In order to verify the effectiveness of the WCMIWO algorithm, the parameters obtained are applied to a Matlab® program created by the author. The results are then compared for three different configurations, as shown in Figure 6.1, to ensure that the optimization can be implemented in various configurations.

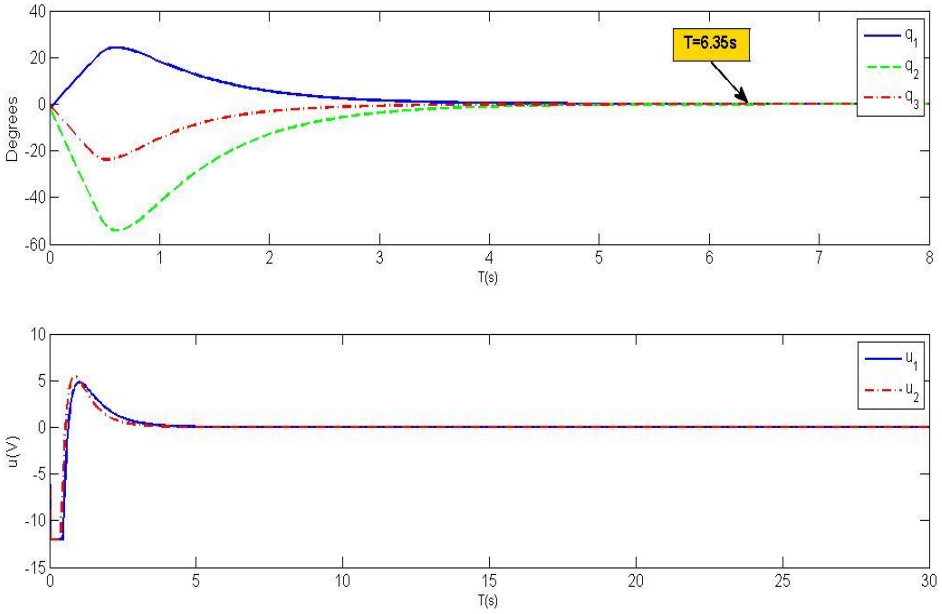


Figure 7.1: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

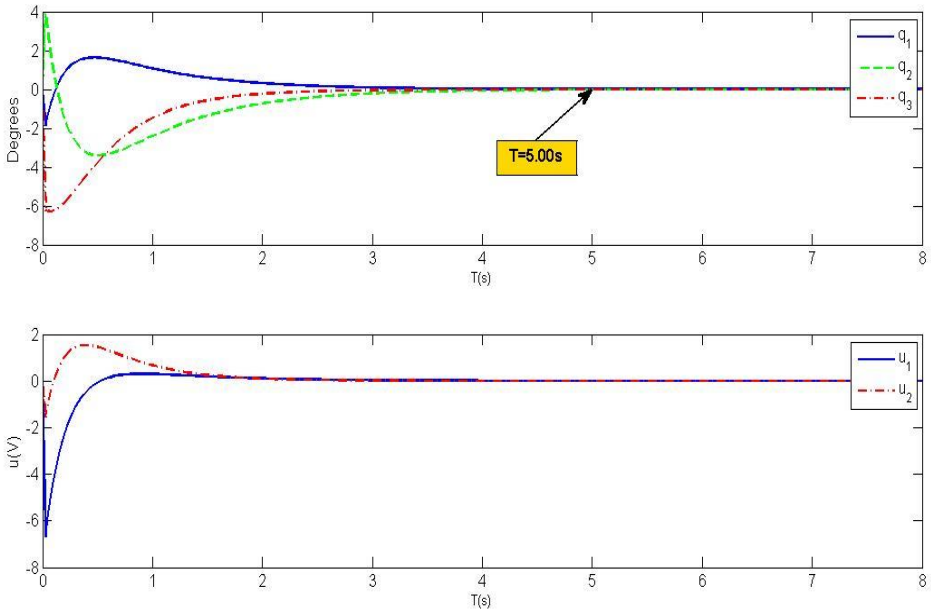


Figure 7.2: Simulation of LQR with initial deflection of θ_1=-3°; θ_2= 3°; θ_2 = -3°

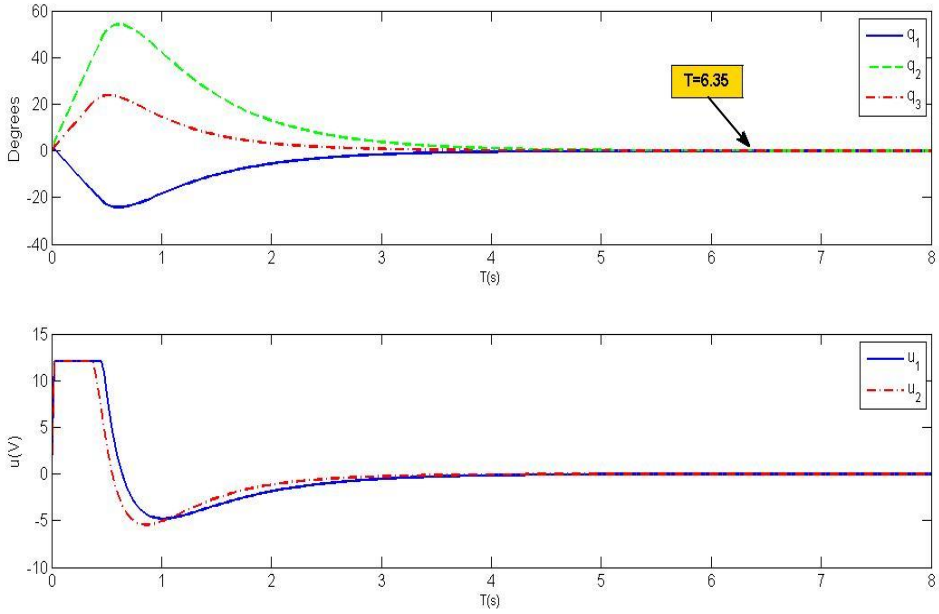


Figure 7.3: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3°

Figure 7.1 shows the system's response and reveals that the voltages required to bring the Robogymnast to a stable upright position with the initial absolute angular position equal to [-3°, -3°, -3°] are 12 volts for both motor 1 and motor 2. It can be seen that the time (T) taken to reach a stable upright position is 6.35 seconds. Figure 7.2 depicts the reaction of the Robogymnast as it attempts to stabilize itself from an initial absolute angular configuration equal to [-3°, 3°, -3°]. The maximum voltage required is 6.62 volts for motor 1 (u_1) and 1.55 volts for motor 2 (u_2). Figure 7.3 shows the response of the system when the initial absolute angular position is equal to [3°, 3°, 3°]. The time (T) taken for the system to stabilize is 6.35 seconds. The maximum voltage for motor 1 (u_1) and motor 2 (u_2) is 12 volts. An external disturbance of 0.05

rad was applied to each of the Robogymnast links one at a time and its reaction observed. The disturbance was applied about two seconds after the controller attempts to stabilize the system from an initial absolute angular position equal to [1.5°, 1.5°, 1.5°]. The objective of this test was to determine the robustness of the LQR controller using the parameters obtained using IWO.

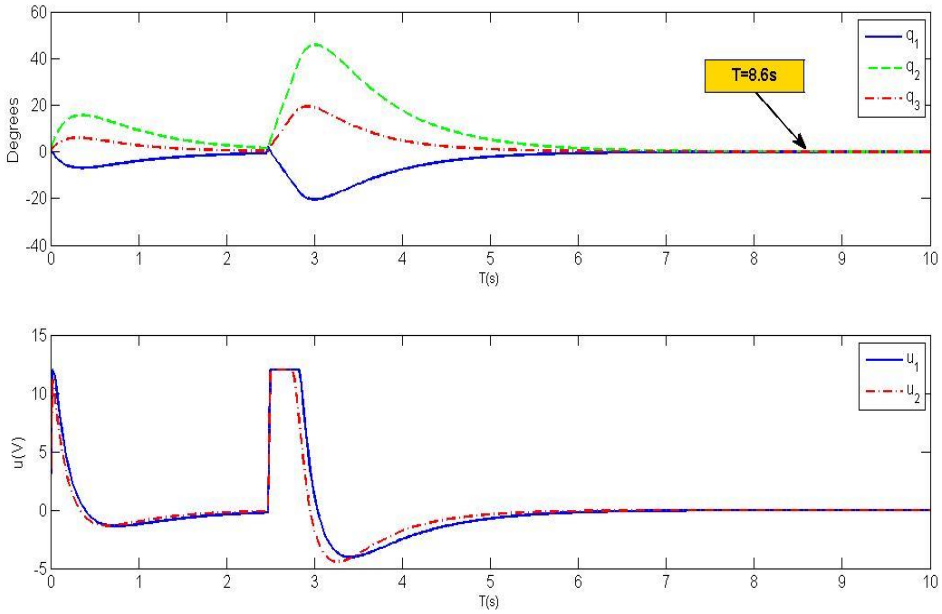


Figure 7.4: Disturbance to Link 1

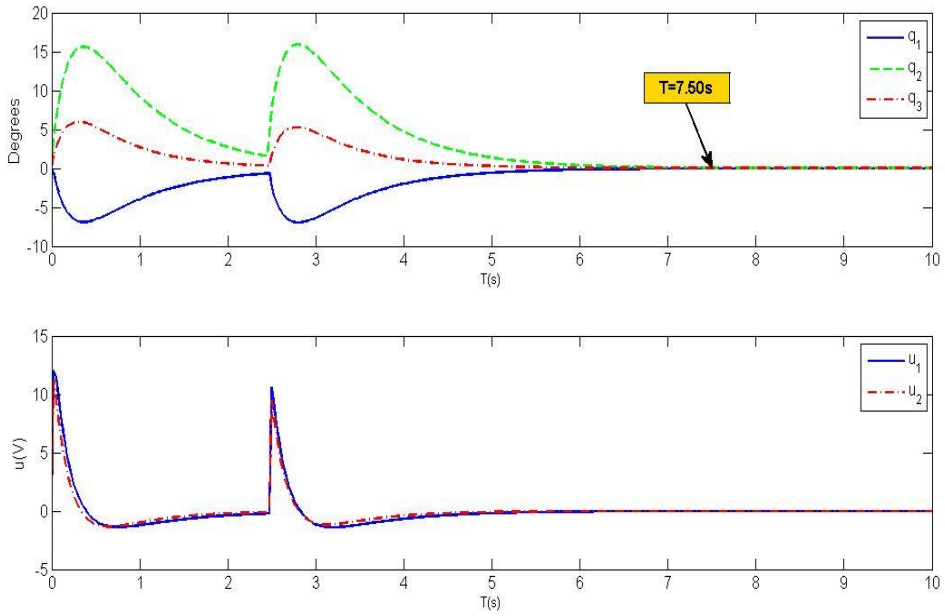


Figure 7.5: Disturbance to Link 2

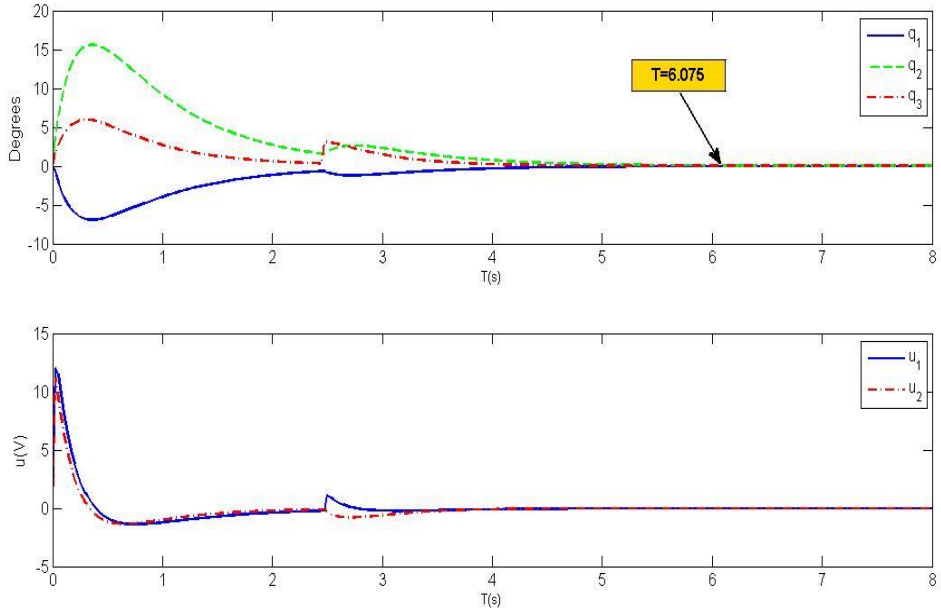


Figure 7.6: Disturbance to Link 3

Figure 7.4 illustrates that when a disturbance is applied to the first link, the controller quickly reacts to counter the displacement. The figure reveals that the amount of work done by both motors is more or less the same. From Figure 7.5, it can be seen that the system experiences significant displacement when a disturbance is applied to the second link. However, despite this, the controller was still able to balance the Robogymnast successfully. Figure 7.6 represents the reaction of the system when a disturbance is applied to the third link. The displacement in this figure is far less severe when compared to Figure 7.4 and Figure 7.5. It can also be seen that when the disturbance is applied, u_1 is larger than u_2. This indicates that most of the work is done by motor 1.

7.3 Fuzzy Logic Invasive Weed Optimization Hybrid

In this section, a multi-objective Fuzzy Logic Invasive Weed Optimization Hybrid (FLIWOH) technique is proposed. This technique uses a combination of the IWO and fuzzy logic. IWO is used for searching and for the generation of new seeds. Fuzzy logic is used to determine the fitness of the seeds by evaluating the fitness memberships of the J and T_{st} criteria. Figures 7.7 and 7.8 show the flowchart of the FLIWOH algorithm.

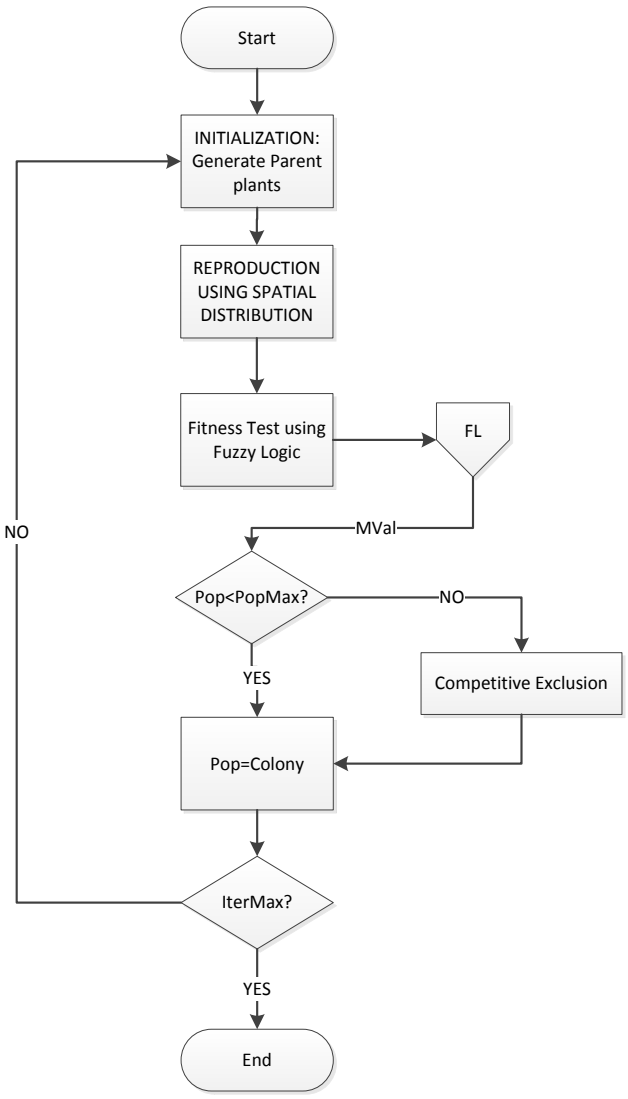


Figure 7.7: The main flowchart of the FLIWOH Algorithm

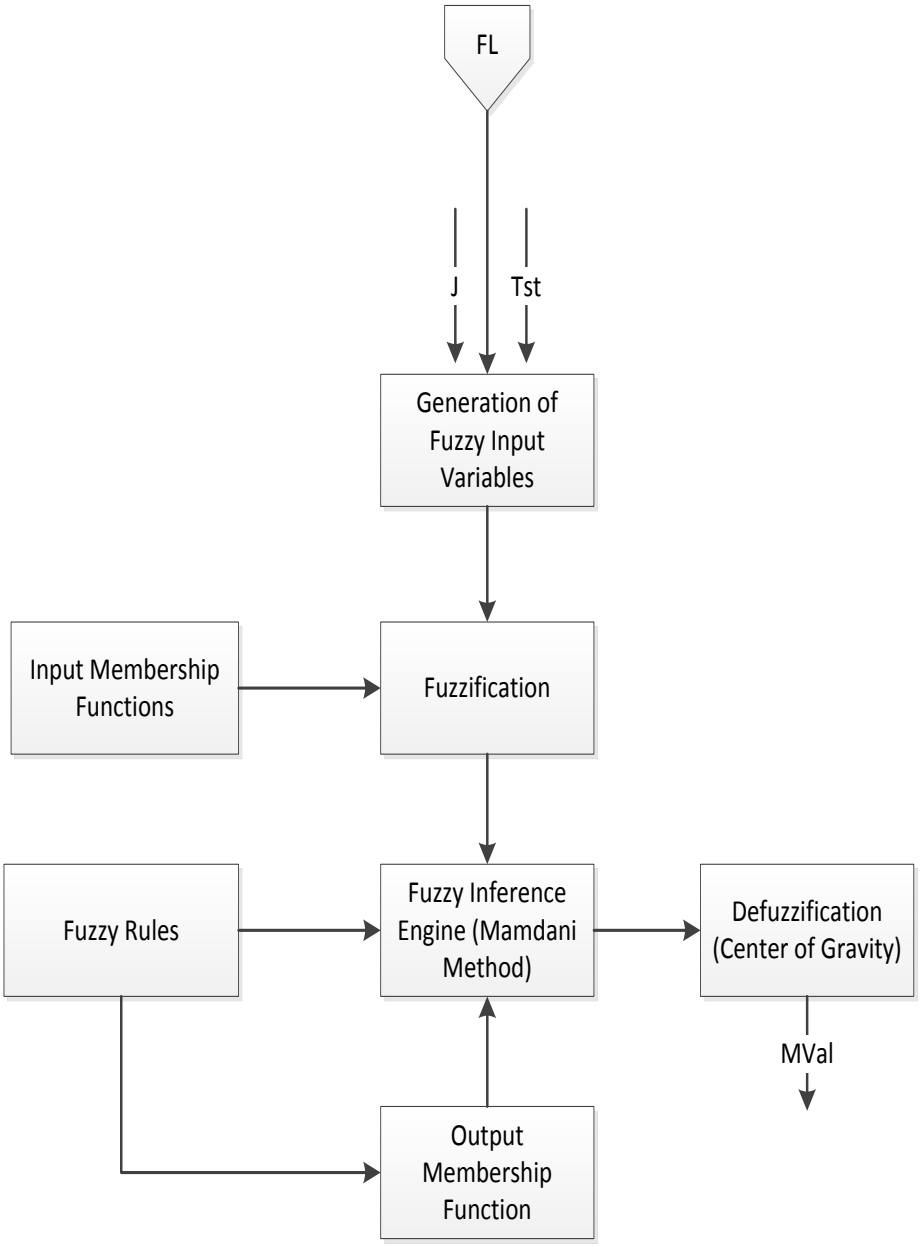


Figure 7.8: The flowchart of the Fuzzy Logic Algorithm

The fuzzy logic processor consists of two input variables and one output variable. Each of the input variables has three membership functions defined in the range of [0, 1] (Figure. 7.9). The output variable consists of three membership functions within the range of [0, 5]. Two well-known Fuzzy rule-based Inference Systems are the Mamdani fuzzy method and the Takagi-Sugeno (T-S) fuzzy method (Chai et al. 2009). The Mamdani method is selected as the fuzzy inference engine due to its expressive power, making it easy to formalize and interpret. Another advantage is that it can be used for both Multiple Input Single Output (MISO) and Multiple Input Multiple Output (MIMO) systems, whereas the T-S method can only be used in MISO systems (Hamam and Georganas 2008). This allows the Mamdani method to be used in future works when MIMO systems are required. The fuzzy logic rules in Table 7.4 are then applied and the output membership function generates the output membership value (MVal). The set seeds are then arranged in ascending order based on their MVal values, where the smaller the value of MVal, the fitter the set of seeds. The set seeds then go through the rest of the conventional IWO process.

Table 7.3: FLIWOH parameters

Variable	Value	Description
Number of initial plants (p_{init})	5	Number of randomly chosen values from the solution space.
Minimum number of seed sets (S_{min})	1	Minimum population of solutions
Maximum number of seed sets (S_{max})	500	Maximum population of solutions
Initial value of standard deviation ($\sigma_{initial}$)	50	Standard deviation used for spatial distribution of plants.
Final value of standard deviation (σ_{final})	0.5	Final standard deviation used for spatial distribution of plants.
Maximum number of iterations ($Iter_{max}$)	10	Number of iterations
Nonlinear Modulation Index	0.01	-
Target angle	q1<0.001 rad q2<0.001 rad q3<0.001 rad	The angle where time is recorded and used as the fitness criterion
Search range	0-1000	Search range used based on trial and error.

Table 7.4: Fuzzy Logic Rule

J \ T_{st}	LOW	AVG	HIGH
LOW	G	AVG	NG
AVG	AVG	AVG	NG
HIGH	AVG	NG	NG

Quality

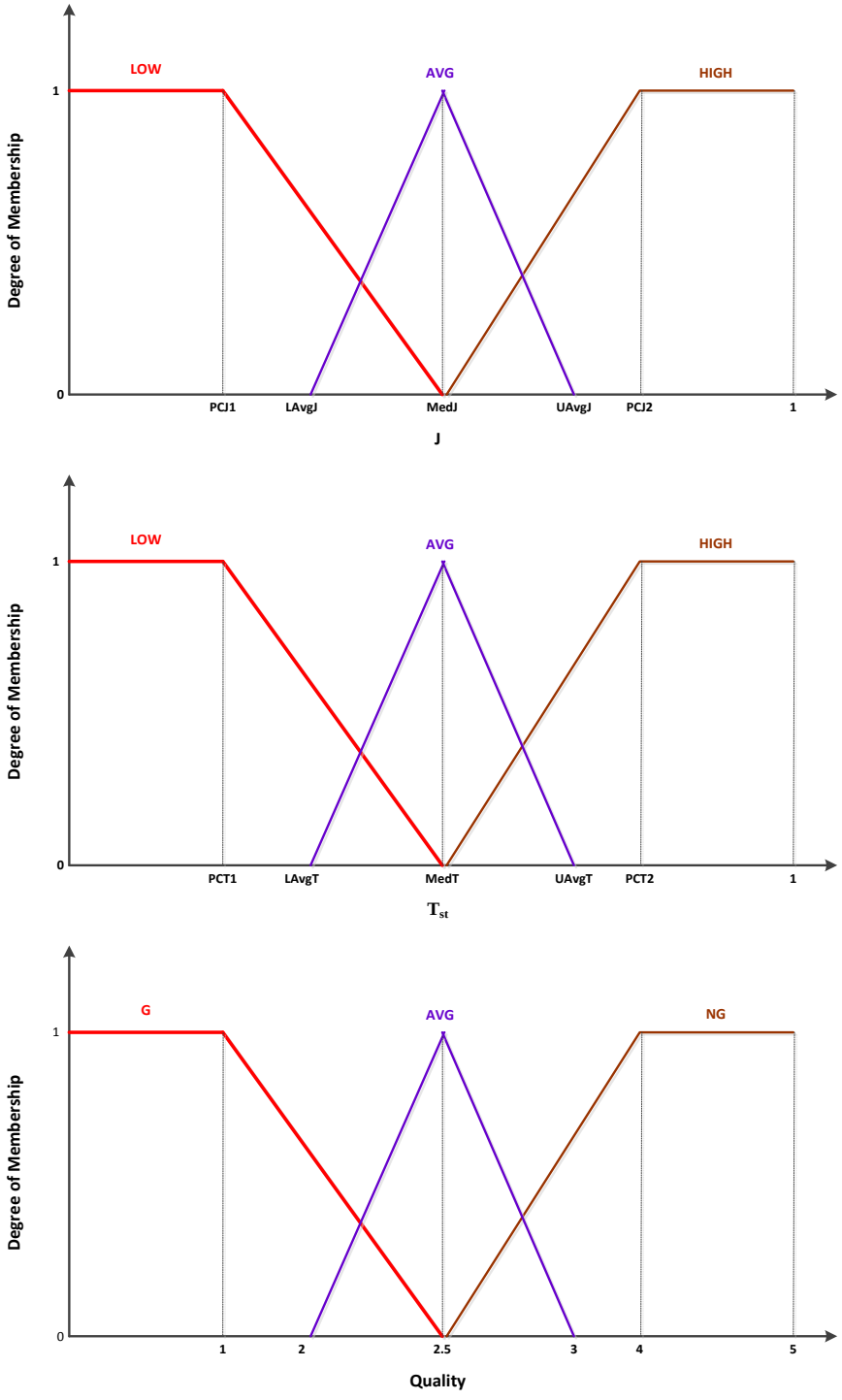


Figure 7.9: Fuzzy Logic Membership Functions

The parameters of the fuzzy logic membership functions are calculated as follows:

$$MedJ = median(J_{norm}) \quad (7.4)$$

$$UAvgJ = MedJ + (0.25)(MedJ) \quad (7.5)$$

$$LAvgJ = MedJ - (0.25)(MedJ) \quad (7.6)$$

$$PCJ1 = (0.25)(MedJ) \quad (7.7)$$

$$PCJ2 = 1 - PCJ1 \quad (7.8)$$

$$MedT = median(T_{norm}) \quad (7.9)$$

$$UAvgT = MedT + (0.25)(MedT) \quad (7.10)$$

$$LAvgT = MedT - (0.25)(MedT) \quad (7.11)$$

$$PCT1 = 0.25(MedT) \quad (7.12)$$

$$PCT2 = 1 - PCT1 \quad (7.13)$$

The range of each membership function had to be re-calculated for each iteration due to the changing range of the seeds' values. J_{norm} and T_{norm} are the normalized values of J and T_{st}. The values of J and T_{st} are normalized to ensure that their ranges fall between [0, 1]. Table 7.5 shows the top ten best seeds set obtained from a population of 500.

Table 7.5: FLIWOH Results

Fitness Rank	S1	S2	S3	S4	S5	S6	Duration to reach the upright position, Tst (s)	J x 10^5	MVal**
1	10.913	100.958	80.485	50.792	30.934	20.655	6.45	9.303	0.9152
2	199.492	179.689	200.470	100.651	70.476	89.536	7.15	48.715	0.9153
3	209.155	179.445	220.230	120.411	50.526	69.640	6.78	45.249	0.9153
4	198.819	119.370	130.565	60.785	50.326	29.608	6.43	19.457	0.9155
5	168.432	199.197	228.200	109.943	60.412	138.827	6.75	57.387	0.9157
6	208.685	199.507	258.648	110.460	50.711	120.046	6.20	54.100	0.9159
7	120.759	190.756	260.140	130.647	50.894	160.390	7.45	59.624	0.9230
8	120.402	190.430	259.515	130.373	50.824	159.905	7.45	59.389	0.9232
9	120.694	190.363	259.617	130.451	50.829	159.998	7.45	59.418	0.9232
10	169.558	199.771	260.114	150.629	40.731	150.223	7.35	66.869	0.9263

** Fitness Criterion

7.3.1 Simulation results of LQR designed using FLIWOH

The fittest seeds, which are S1=50.348, S2=500.587, S3=400.658, S4=250.002, S5=150.174 and S6=100.002, are selected for analysis. Using Equation 6.6, the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 0.1191e^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.1924e^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6.4779e^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.5798e^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9569e^3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.4266e^3 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -508.2175 & -217.2200 & -26.5988 & 92.8740 & 48.8864 & 6.0772 \\ -390.6186 & -175.1066 & 6.8220 & 71.8430 & 37.8372 & 4.6029 \end{matrix}$$

The parameters obtained are subjected to the same tests as the WCMIWO parameters.

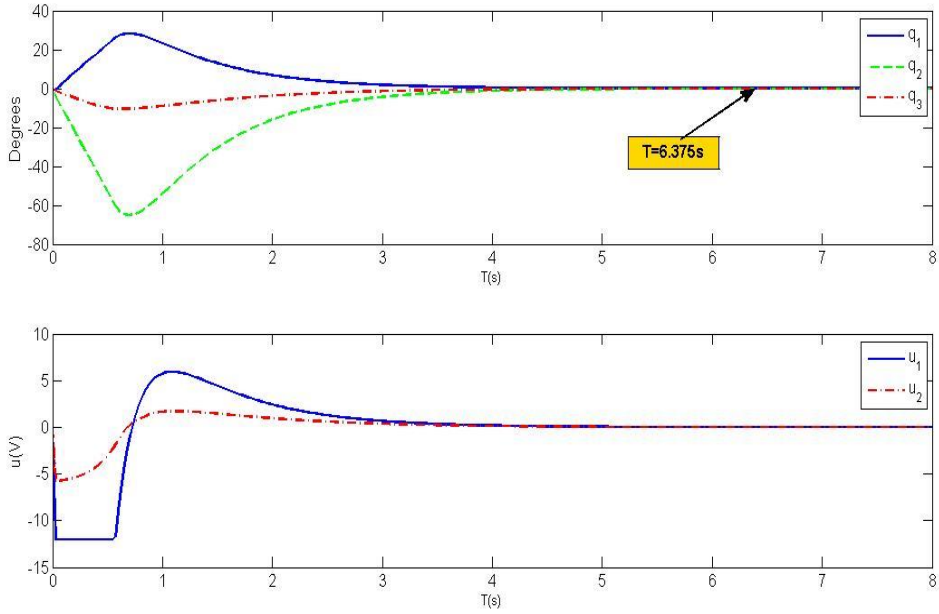


Figure 7.10: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

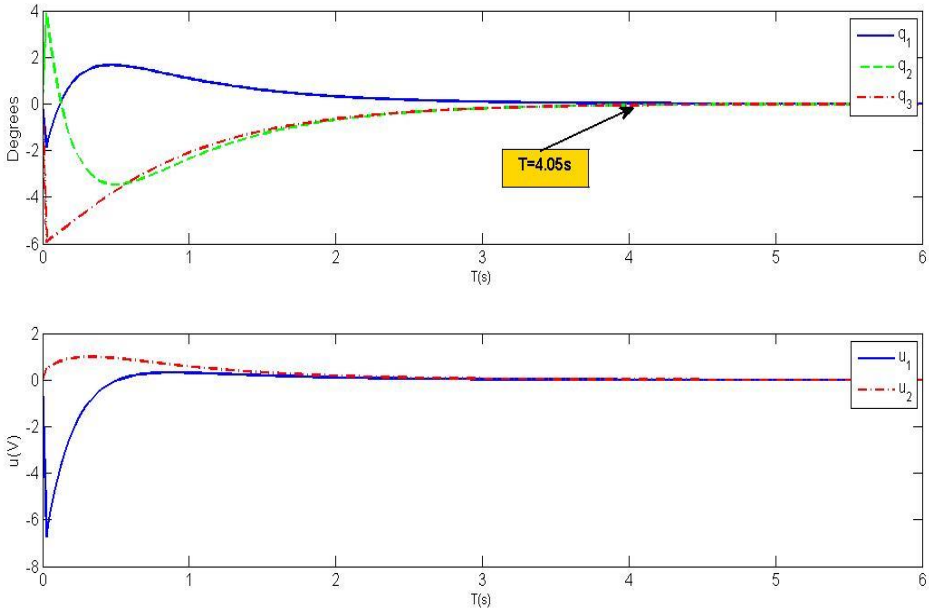


Figure 7.11: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°

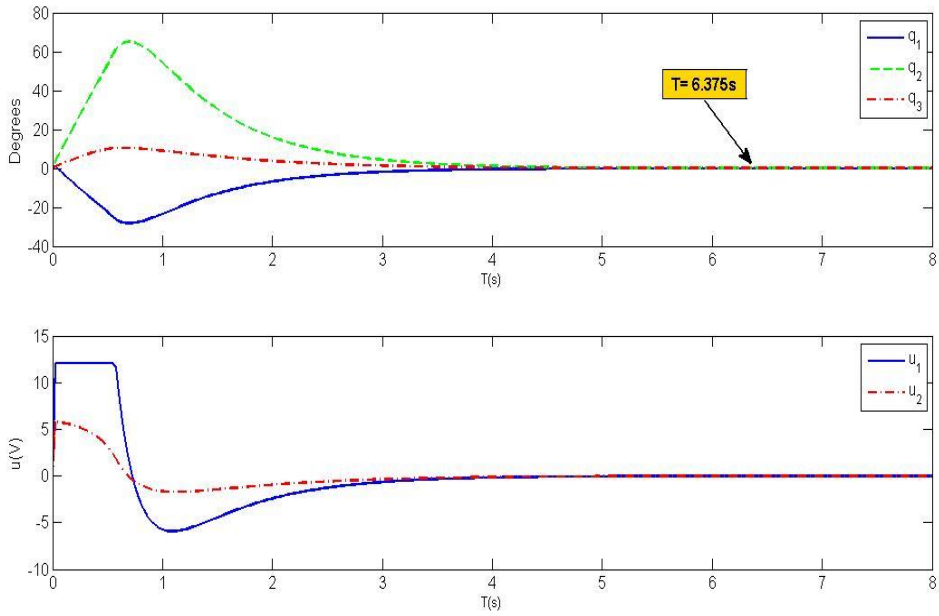


Figure 7.12: Simulation of LQR with initial deflection of θ_1=3°; θ_2=3°; θ_2=3°

Figure 7.10 illustrates the controlled system response and the voltages when the Robogymnast is in the upright position with the initial absolute angular position equal to [-3°, -3°, -3°]. The maximum voltage u_1 is 12V, while u_2 is significantly lower at 5.8159V. It can be seen that the time taken to reach a stable upright position is 6.375 seconds. Figure 7.11 shows the response of the system when the initial absolute angular position is equal to [-3°, 3°, -3°]. The time taken for the system to stabilize is 4.05 seconds. The maximum voltage is 6.7462 volts for u_1 and 0.9909 volts for u_2. Figure 7.12 illustrates the controller's ability to stabilise the Robogymnast when it is in the upright position with the initial absolute angular position equal to [3°, 3°, 3°]. The maximum voltage for motor 1 (u_1) is 12 volts,

and for motor 2 (u_2) is 5.8159 volts. It can be seen that the time taken to reach a stable upright position is 6.375 seconds.

An external disturbance of 0.05 rad or 2.87° was applied to each of the Robogymnast links one at a time and its reaction was observed. The disturbance was applied about two seconds after the controller attempted to stabilize the system from an initial absolute angular position equal to [1.5°, 1.5°, 1.5°]. Figure 7.13 shows the effect a disturbance has on the system when applied to the first link. The system was able to counter the disturbance and stabilize the system successfully. Maximum voltage of u_1 is more than double of voltage u_2, thus showing that most of the work is done by motor 1.

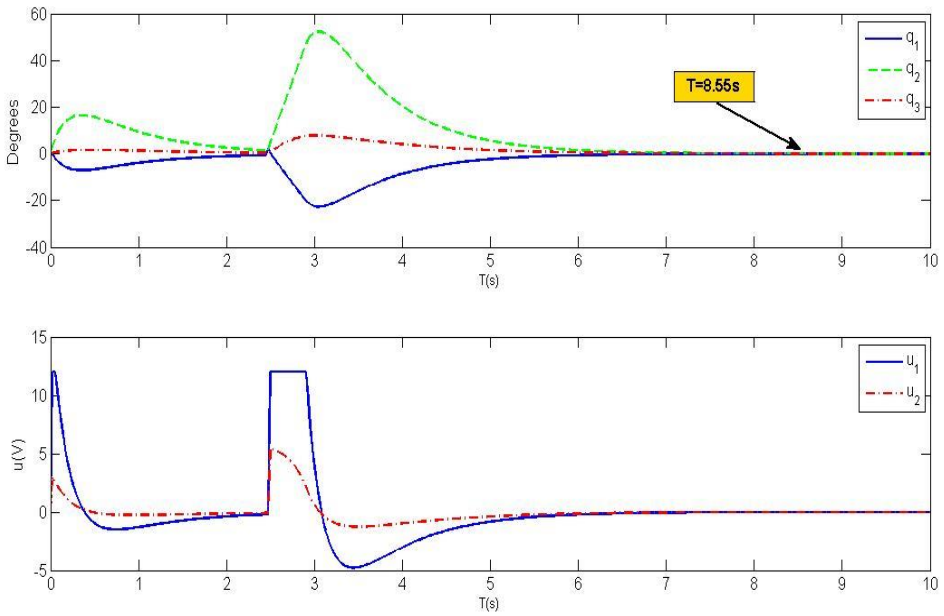


Figure 7.13: Disturbance to Link 1

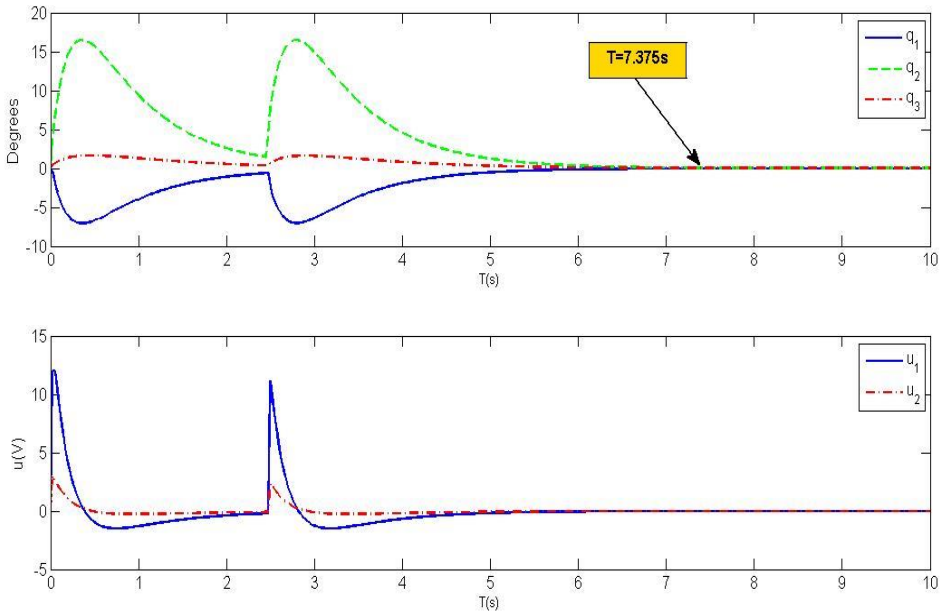


Figure 7.14: Disturbance to Link 2

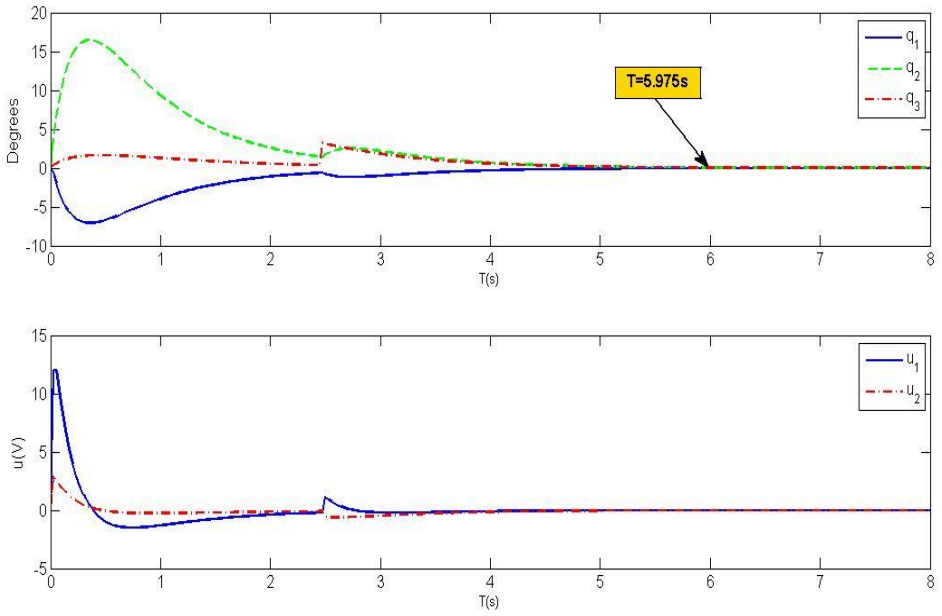


Figure 7.15: Disturbance to Link 3

Figure 7.14 depicts the controller's successful attempt to balance the Robogymnast when a disturbance is applied to the second link. It can be seen that motor 1 requires significantly larger voltage compared to motor 2. The controller is able to stabilize the robot in 7.375 seconds. Figure 7.15 displays the reaction of the system when a disturbance is applied to the third link. The displacement caused by the system is minor, thus requiring very small voltages for both motors.

7.4 Training with disturbance

In this section, the optimization procedures in sections 7.4 and 7.5 were repeated with minor disturbance applied to the system model. The disturbance consists of random values between the range [0.01 rad, 0.05 rad], which were multiplied with previous states and added to the present states. This is to simulate the application of external disturbance to the system. It is expected that the increased difficulty in the optimization process would generate seeds that would perform much better when applied to the system without disturbance.

7.4.1 WCMIWO training with disturbance results

The fittest seeds, which are S1=28.398, S2=26.475, S3=29.353, S4=11.210, S5=6.811 and S6=10.833, are selected for analysis. Using Equation 6.6, the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 806.4354 & 0 & 0 & 0 & 0 & 0 \\ 0 & 700.9120 & 0 & 0 & 0 & 0 \\ 0 & 0 & 861.5681 & 0 & 0 & 0 \\ 0 & 0 & 0 & 125.6577 & 0 & 0 \\ 0 & 0 & 0 & 0 & 46.3923 & 0 \\ 0 & 0 & 0 & 0 & 0 & 117.3512 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -551.7618 & -237.6276 & -27.8633 & 100.8604 & 53.0928 & 6.5943 \\ -88.0097 & -39.6871 & 12.5886 & 16.1881 & 8.5312 & 1.0046 \end{matrix}$$

Figures 7.16 to 7.18 shows the controller's reaction when attempting to balance the Robogymnast in an inverted position from three different configurations as in Figure 6.1.

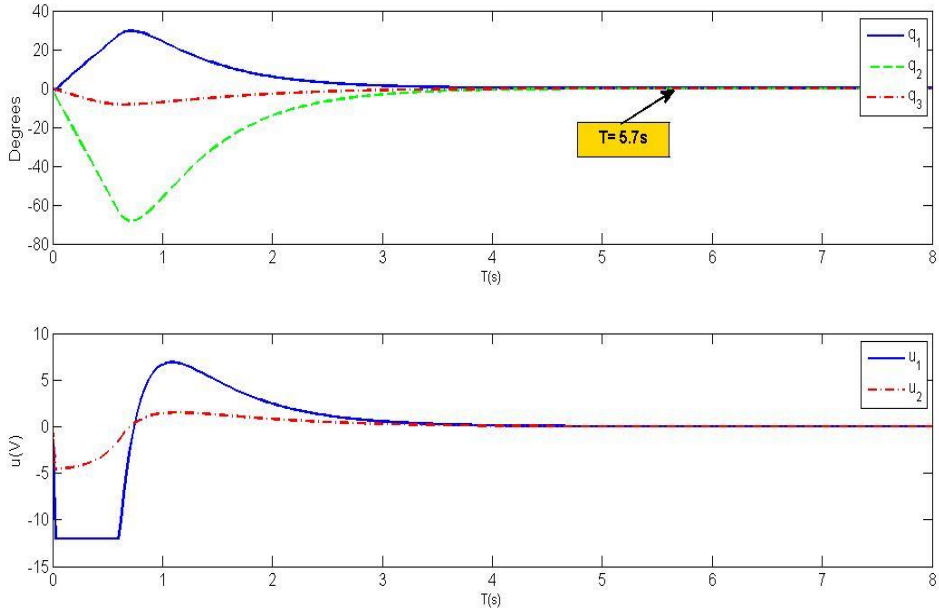


Figure 7.16: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

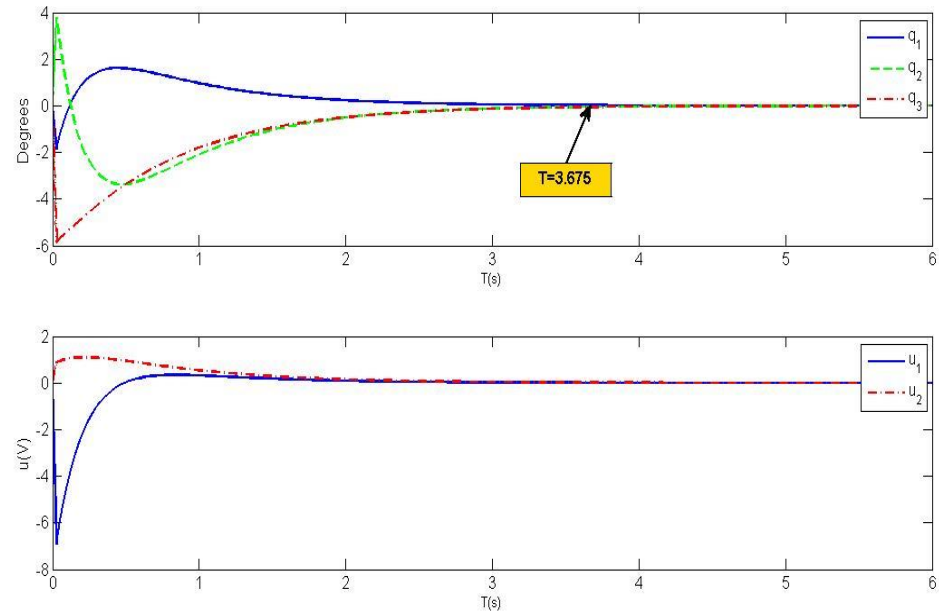


Figure 7.17: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°

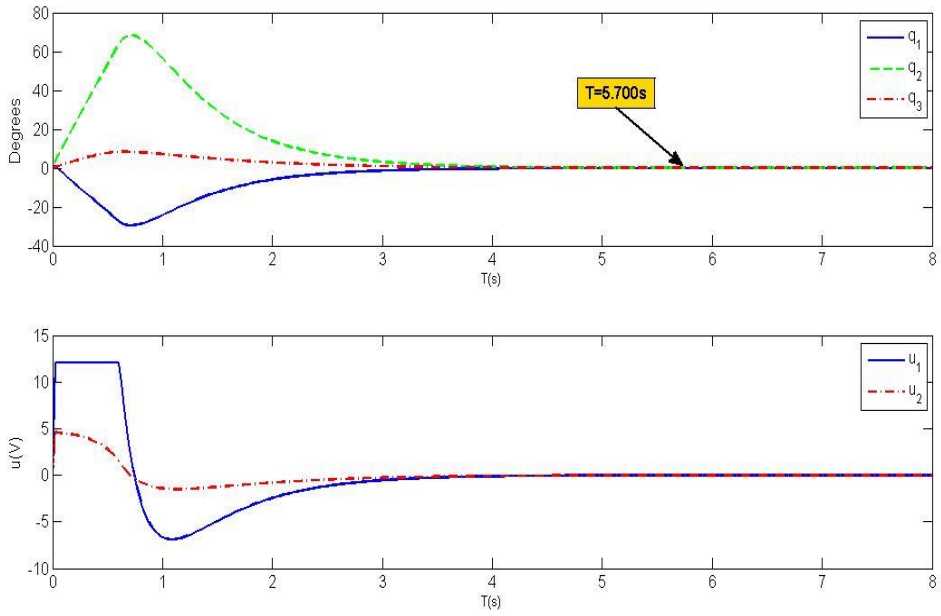


Figure 7.18: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

7.4.2 FLIWOH training with disturbance results

The fittest seeds, which are S1=25.0659, S2=16.9821, S3=24.7988, S4=8.7119, S5=3.8212 and S6=9.9280, are selected for analysis. Using equation (6.6), the Q matrix obtained from the seeds is:

$$Q = \begin{bmatrix} 628.2976 & 0 & 0 & 0 & 0 & 0 \\ 0 & 288.3915 & 0 & 0 & 0 & 0 \\ 0 & 0 & 614.9822 & 0 & 0 & 0 \\ 0 & 0 & 0 & 75.8976 & 0 & 0 \\ 0 & 0 & 0 & 0 & 14.6019 & 0 \\ 0 & 0 & 0 & 0 & 0 & 98.5659 \end{bmatrix}$$

and the corresponding gain matrix is:

$$F = \begin{matrix} -550.7529 & -236.9672 & -27.5668 & 100.8604 & 52.9685 & 6.5779 \\ -49.5513 & -22.2015 & 12.8659 & 9.1323 & 4.8142 & 0.5500 \end{matrix}$$

The controller is tested in three different configurations as in Section 7.4.1 and the results are shown in Figures 7.19 to 7.21.

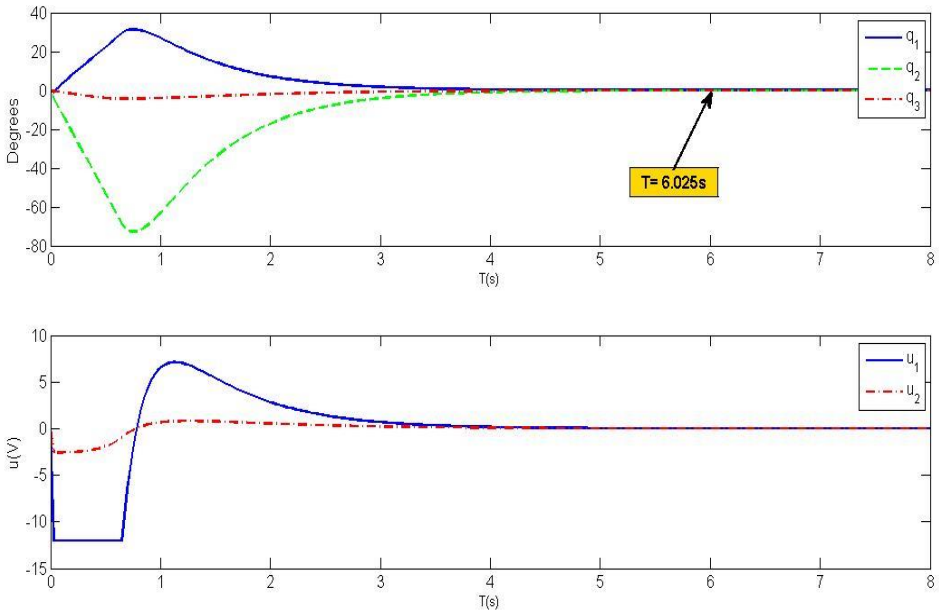


Figure 7.19: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3

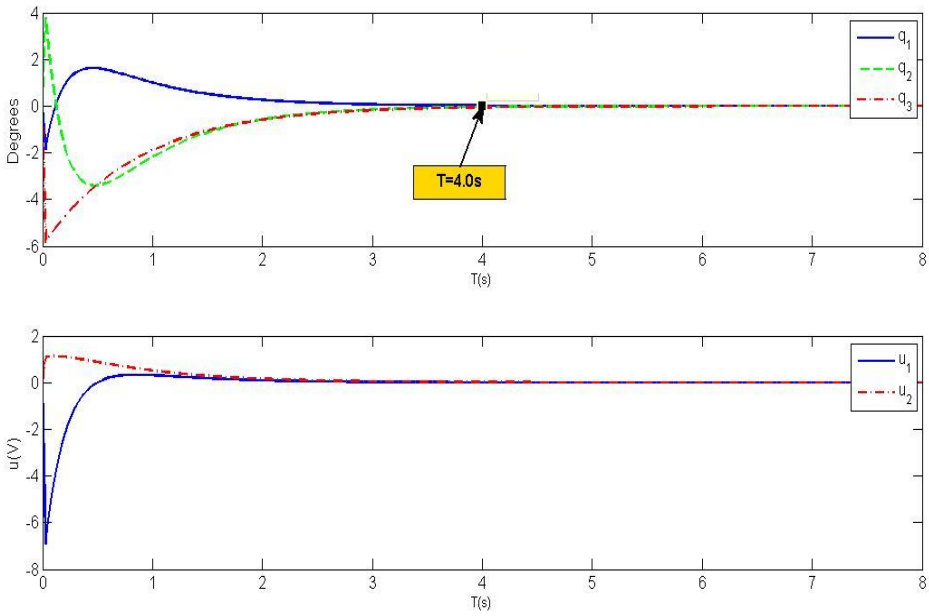


Figure 7.20: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=3°; θ_2=-3°

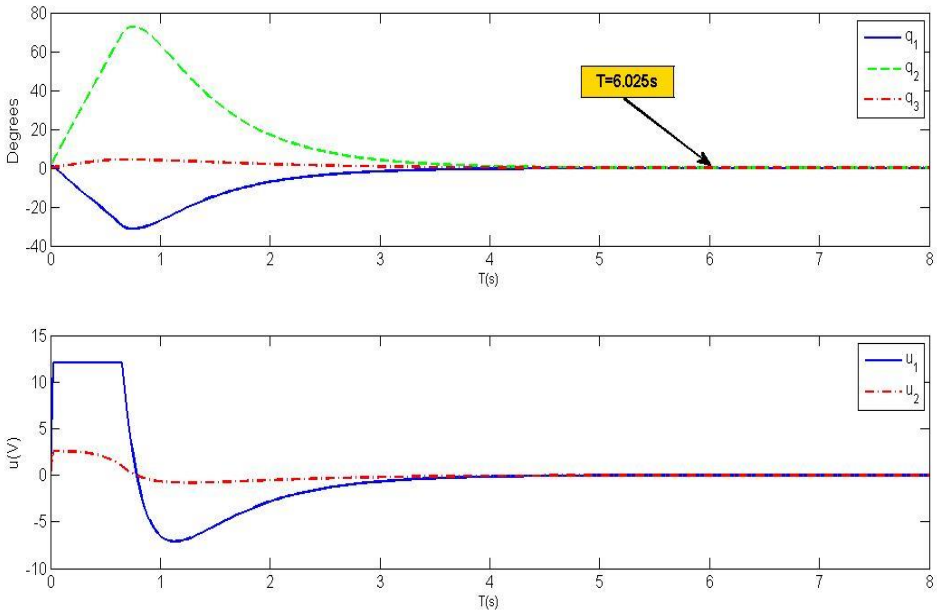


Figure 7.21: Simulation of LQR with initial deflection of θ_1=-3°; θ_2=-3°; θ_2=-3°

7.5 Discussion and conclusion

The simulation results proved that the LQR controller designed using parameters obtained by both methods can successfully bring the Robogymnast to an inverted and stable configuration. WCMIWO and FLIWOH produced LQR controllers that have similar reaction times (within the range of less than 4%) to each other but are slower compared to the LQR controllers trained with disturbance. The WCMIWO LQR controller uses between 1.26% to 8.62% less voltage for motor 1 (u_1) compared to the other methods. However, it requires between 27.66% to 88.9% higher voltage for motor 2 (u_2) when compared to the other methods, with FLIWOH with disturbance requiring the lowest voltage (u_2) for motor 2 in almost all configurations. This result is consistent throughout the three configurations. In order to further analyse the performance of the controllers, more tests had to be done. Tables 7.6 and 7.7 compare the performance of the two controllers in different initial angular configurations

Table 7. 6: Comparison of the performance of controllers in different initial angular *(small angles)* configurations.

Deflection Angle	Method	Jsum	Tmax(s)	$u_1$max(V)	$u_2$max(V)	Description/Purpose
θ_1=1°; θ_2=1°; θ_3=1	WCMIWO	1313100	4.775	8.866	7.457	To examine controller's reaction at low deflection angles.
	FLIWOH	1649	4.600	9.272	1.939	
	WCMIWO with Disturbance	3449	4.075	9.630	1.536	
	FLIWOH with Disturbance	1745	4.250	9.613	0.865	
θ_1=1°; θ_2=-1°; θ_3=1°	WCMIWO	72391	3.175	2.221	0.517	To examine controller's reaction at low deflection angles.
	FLIWOH	94	3.125	2.249	0.330	
	WCMIWO with Disturbance	196	2.850	2.308	0.366	
	FLIWOH with Disturbance	109	3.050	2.303	0.374	
θ_1=3°; θ_2=3°; θ_3=3°	WCMIWO	22296000	6.35	12.000	12.000	Figure 7.3 and 7.12
	FLIWOH	34779	6.375	12.000	5.816	
	WCMIWO with Disturbance	81669	5.700	12.000	4.608	
	FLIWOH with Disturbance	44946	6.025	12.000	2.595	
θ_1=3°; θ_2=-3°; θ_3=3°	WCMIWO	651520	4.075	6.662	1.550	Opposite of Figure 7.2 and 7.11
	FLIWOH	843	4.050	6.746	0.991	
	WCMIWO with Disturbance	1766	3.675	6.924	1.097	
	FLIWOH with Disturbance	979	4.000	6.909	1.121	
θ_1=3.1°; θ_2=3.1°; θ_3=3.1°	WCMIWO	28180000	6.525	12.000	12.000	To examine the controller's reaction
	FLIWOH	49311	6.650	12.000	6.010	
	WCMIWO with Disturbance	123270	6.025	12.000	4.762	
	FLIWOH with Disturbance	77390	6.475	12.000	2.700	

Table 7. 7: Comparison of the performance of controllers in different initial angular *(large angles)* configurations.

Deflection Angle	Method	Jsum	Tmax(s)	$u_1$max(V)	$u_2$max(V)	Description/Purpose
θ_1=0°; θ_2=4°; θ_3=5°	WCMIWO	4526100	5.375	12.000	12.000	To examine the controller's reaction when higher deflection angles are applied to link 2 and link 3.
	FLIWOH	5751	5.200	12.000	3.298	
	WCMIWO with Disturbance	12349	4.600	12.000	2.551	
	FLIWOH with Disturbance	6220	4.800	12.000	1.325	
θ_1=4°; θ_2=0°; θ_3=0°	WCMIWO	7485100	5.575	12.000	12.000	To examine the controller's reaction when higher deflection angles are applied to link 1.
	FLIWOH	9819	5.450	12.000	4.277	
	WCMIWO with Disturbance	20920	4.825	12.000	3.374	
	FLIWOH with Disturbance	10680	5.050	12.000	1.909	
θ_1=5.45°; θ_2=0°; θ_3=0°	WCMIWO	33873000	6.700	12.000	12.000	To test maximum deflection angle the controller can recover from.
	FLIWOH	1411751	6.800	12.000	12.000	
	WCMIWO with Disturbance	Inf	Inf	12.000	12.000	
	FLIWOH with Disturbance	Inf	Inf	12.000	12.000	
θ_1=5.65°; θ_2=0°; θ_3=0°	WCMIWO	71745000	7.375	12.000	12.000	To test maximum deflection angle the controller can recover from.
	FLIWOH	2978200	7.500	12.000	12.000	
	WCMIWO with Disturbance	Inf	Inf	12.000	12.000	
	FLIWOH with Disturbance	Inf	Inf	12.000	12.000	
θ_1=5.7°; θ_2=0°; θ_3=0°	WCMIWO	Inf	Inf	12.000	12.000	To test maximum deflection angle the controller can recover from.
	FLIWOH	Inf	Inf	12.000	12.000	
	WCMIWO with Disturbance	Inf	Inf	12.000	12.000	
	FLIWOH with Disturbance	Inf	Inf	12.000	12.000	

Tables 7.6 and 7.7 show that all the controllers trained with disturbance achieved faster settling times compared to their counterparts that were not trained with disturbance. At small angles, voltage u_1 is lower for WCMIWO, but at large angles this difference ceases to exist. Voltage u_2 is lower for FLIWOH trained with disturbance at almost all configurations. Both controllers that were trained with disturbance were unable to recover the Robogymnast when the initial absolute angular position was equal to [5.45°, 0°, 0°]. Further tests showed that WCMIWO trained with disturbance can recover from a maximum initial angular position of [5.432°, 0°, 0°], while FLIWOH trained with disturbance can only recover from the initial angular position of [5.369°, 0°, 0°]. WCMIWO and FLIWOH can recover from a maximum initial angular position of [5.692°, 0°, 0°]. Table 7.8 shows the ranking of performance of the four controllers, where 1 is the best and 4 is the worst.

Table 7.8: Ranking of Performance

Method	**Settling Time**	**Efficiency of motor 1 (u_1)**	**Efficiency of motor 2 (u_2)**	**Ability to upright from larger initial angles of deflection**
WCMIWO	4	1	4	1
FLIWOH	3	2	3	1
WCMIWO with Disturbance	1	3	2	2
FLIWOH with Disturbance	2	3	1	3

Based on Table 7.8 the WCMIWO LQR controller has the highest ranking for the efficiency of motor 1 and the ability to achieve an upright position from larger initial angles. This makes it the most suitable controller for this application, largely because of the system's dependency on u_1 to maintain the Robogymnast in an upright position is larger than its dependency on u_2. This can be seen from the results, as u_1 is usually larger than u_2. Since both u_1 and u_2 have a maximum limit of 12V, it is in its best interest that the required value of u_1 be as small as possible.

Table 7.9 shows the performance of the four controllers compared to the controllers LQRJ and LQRT in chapter 6. As expected LQRT performs faster compared to the other techniques, while LQRJ performs the slowest. The results also show that LQRJ uses the least amount of voltage for u_1 at small angles but uses the most amount of voltage at medium angles.

Table 7.9: Comparison with LQRJ and LQRT

Deflection Angle	Method	Jsum	Tmax(s)	$u_1$max(V)	$u_2$max(V)	Remarks
θ_1=1°; θ_2=1°; θ_3=1	LQRJ	605640	6.700	7.732	12.000	Small angles
	LQRT	2725800	3.625	10.105	2.097	
	WCMIWO	1313100	4.775	8.866	7.457	
	FLIWOH	1649	4.600	9.272	1.939	
	WCMIWO with Disturbance	3449	4.075	9.630	1.536	
	FLIWOH with Disturbance	1745	4.250	9.613	0.865	
θ_1=3°; θ_2=-3°; θ_3=3°	LQRJ	325330	5.700	7.322	9.541	Medium angles
	LQRT	1301800	3.100	7.296	1.927	
	WCMIWO	651520	4.075	6.662	1.550	
	FLIWOH	843	4.050	6.746	0.991	
	WCMIWO with Disturbance	1766	3.675	6.924	1.097	
	FLIWOH with Disturbance	979	4.000	6.909	1.121	
θ_1=5.45°; θ_2=0°; θ_3=0°	LQRJ	15839000	9.375	12.000	12.000	Large angles
	LQRT	201070000	5.600	12.000	6.195	
	WCMIWO	33873000	6.700	12.000	12.000	
	FLIWOH	1411751	6.800	12.000	12.000	
	WCMIWO with Disturbance	Inf	Inf	12.000	12.000	
	FLIWOH with Disturbance	Inf	Inf	12.000	12.000	

7.6 Summary

The purpose of this chapter was to determine whether the multi-objective IWO could produce an LQR controller that takes into consideration the values of the J cost function and settling time (T_{st}). The first optimization method (WCMIWO) applies IWO for WCM optimization of the J and T_{st} values. Weight values were assigned to each variable and the resulting values were multiplied by each other to produce a single value which is used as the fitness criterion. The second optimization method (FLIWOH) is a hybrid IWO that employs fuzzy logic to attain a membership value which is used as the fitness criterion. Using the Q values obtained, two LQR controllers were designed and tested using simulation. Two other controllers were designed using the previous two methods but trained with minor disturbances. All four controllers successfully balanced the Robogymnast in an inverted configuration even when external disturbance was applied to it. The four controllers were examined and their performance evaluated.

CHAPTER 8

Conclusions, Contributions and Future Work

This chapter summarises the conclusions and contributions of this research. It also provides suggestions for future work.

8.1 Conclusions

In conclusion, all the objectives in Chapter 1 have been met.

- A swing-up method for the Robogymnast through the manipulation of its motor control signals was successfully designed. This was achieved by selecting the values of control parameters $\Delta\alpha_1$, $\Delta\alpha_2$, $\Delta\delta_1$ and $\Delta\delta_2$. Two parameters ($\Delta\alpha_1$ and $\Delta\delta_1$) were assigned to control signal u_1 and another two parameters ($\Delta\alpha_2$ and $\Delta\delta_2$) were assigned to control signal u_2.
- A swarm-based optimisation technique was applied to optimise the parameters of the control signals. The IWO was used to optimize the swing-up motion of

the robot by determining the optimum values of parameters that control the input sinusoidal voltage of the two motors.

- The optimised parameters were applied to the swing-up motion of the Robogymnast. The values obtained from IWO were then applied to both simulation and experiment. Results showed that the swing-up motion of the Robogymnast from the stable downward position to the inverted configuration was successfully accomplished.
- A neural network model of the system was developed and applied. A multi-layered ENN model was used to represent the system. The ENN was selected because its behaviour is similar to that of a state space equation.
- The ENN model of the Robogymnast was analysed and validated. Inputs were applied to both the mathematical model and the neural network model and their outputs were analyzed and compared with the actual system's behaviour.
- Controllers to balance the Robogymnast in an inverted configuration were developed. Two LQR controllers were designed and their behaviour examined.
- The optimised parameters of the controllers were selected using swarm-based optimization techniques. The fitness criteria chosen were the cost function J and settling time T_{st}. The fitness criteria were employed on the IWO and used to obtain optimum diagonal values of the Q matrix.
- Modified swarm-based multi-objective optimisation techniques were developed to optimise the selection of the controller parameters. Two multi-objective optimization methods based on IWO were proposed. The WCMIWO

is a weight criteria method, while the FLIWOH uses fuzzy logic to determine the fitness of the set seeds.

- Four controllers were proposed using the MOO. The proposed controllers were validated through simulation.

8.2 Contributions

The novel contributions made in this study are as follows:

1. Introduced a new method to manipulate the amplitudes and frequencies of the sinusoidal control signals of the motors by assigning four parameters ($\Delta\alpha_1$, $\Delta\alpha_2$, $\Delta\delta_1$ and $\Delta\delta_2$) to control the amplitudes and frequencies of the control signals **(Chapter 4).**
2. Employed a novel optimisation method (Invasive Weed Optimization) to find the optimal values of the control signals' parameters to achieve smooth swing-up motion of the Robogymnast **(Chapter 4).**
3. Applied the optimised parameters of the swing-control on the real system **(Chapter 4).**
4. Created a neural network model of the Robogymnast as an alternative to the mathematical model **(Chapter 5).**
5. Applied the neural network model of the Robogymnast to the upright balancing control **(Chapter 6 and Chapter 7).**

6. Developed an LQR controller by using IWO and incorporating the cost function (J) and settling time (T_{st}) as fitness criteria in its design **(Chapter 6).**
7. Designed two multi-objective optimisation techniques to optimize the LQR controller parameters **(Chapter 7).**
8. Applied disturbance to the system during the optimisation procedure to create a more robust controller **(Chapter 7).**
9. Analyzed the ability of the designed controllers to overcome external disturbance to the system **(Chapter 7).**
10. Compared the performance of the controllers in different configurations of the Robogymnast **(Chapter 6 and Chapter 7).**

8.3 Future work

This section discusses some of the future works that can be implemented:

1. Design a catching controller to 'catch' or hold the system as it swings close to the inverted position to assist in the transition from swing-up control to balancing control.
2. Implement different control methods such as Q-learning in the design of a balancing controller for the Robogymnast.

3. Develop learning algorithms for the WCMIWO to make the weights adaptable based on the application.
4. Analyze the Robogymnast system to determine any improvements that can be made.
5. Perform a study on the dimensions of the Robogymnast in order to find a more optimal design of the mechanism. The length and weight of the links have to be re-evaluated. The material used (aluminium) should be replaced with stronger but lighter material.
6. Analyze selection of actuators that would improve the motion of the Robogymnast. This might include replacing the actuator on joint 2 (shoulder joint) with more powerful ones. This is because based on the findings in the research, the power requirement for the actuator on joint 2 is greater than the actuator on joint 3 (hip joint).
7. Apply the proposed controllers on the Robogymnast system after implementation of points 4, 5 and 6.

APPENDIX

Appendix A

A.1. LM12CL 80W Operational Amplifier

The LM12 is a power op amp capable of driving ±25V at ±10A while operating from ±30V supplies. The monolithic IC can deliver 80W of sine wave power into a 4Ω load with 0.01% distortion. Power bandwidth is 60 kHz. Further, a peak dissipation capability of 800W allows it to handle reactive loads such as transducers, actuators or small motors without derating. Important features include the following:

The IC delivers ±10A output current at any output voltage yet is completely protected against overloads, including shorts to the supplies. The dynamic safe area protection is provided by instantaneous peak temperature limiting within the power transistor array. The turn-on characteristics are controlled by keeping the output open-circuited until the total supply voltage reaches 14V. The output is also opened as the case temperature exceeds 150°C or as the supply voltage approaches the BV of the output transistors. The IC withstands over-voltages to 80V. This monolithic op amp is compensated for unity-gain feedback, with a small-signal bandwidth of 700 kHz. Slew rate is 9V/μseconds, even as a follower. Distortion and capacitive-

load stability rival that of the best designs using complementary output 186 transistors. Further, the IC withstands large differential input voltages and is well behaved should the common-mode range be exceeded. The LM12 establishes that monolithic ICs can deliver considerable output power without resorting to complex switching schemes. Devices can be paralleled or bridged for even greater output capability. Applications include operational power supplies, high-voltage regulators, high-quality audio amplifiers, tape-head petitioners, x-y plotters or other servo-control systems.

The LM12 is supplied in a four-lead, TO-220 package with V− on the case. A gold-eutectic die-attach to a molybdenum interface is used to avoid thermal fatigue problems. The LM12 is specified for either military or commercial temperature range. The LM12 is prone to low-amplitude oscillation bursts coming out of saturation if the high-frequency loop gain is near unity. The voltage follower connection is most susceptible. This glitching can be eliminated at the expense of small-signal bandwidth using input compensation. When a push-pull amplifier goes into power limit while driving an inductive load, the stored energy in the load inductance can drive the output outside the supplies. Although the LM12 has internal clamp diodes that can handle several amperes for a few milliseconds, extreme conditions can cause destruction of the IC. The internal clamp diodes are imperfect in that about half the clamp current flows into the supply to which 187 the output is clamped, while the other half flows across the supplies. Therefore, the

use of external diodes to clamp the output to the power supplies is strongly recommended.

A.2. E-Series Tachometer Generators

The E-Series (Subminiature) 3 volts/1000 RPM DC tachometer generator is the smallest tach generator among those offering similar technical characteristics. Many outstanding features make it particularly suitable for use in all types of servo systems. Although its diameter is only 0.760‖, the E-Series provides up to 3 V/1000 RPM output. Almost any Servo-Tek DC tachometer generator can be manufactured with special configurations, various electrical specifications and shaft modifications such as flats, pinions, holes, etc.

Appendix B

B.1. Robogymnast Controller Program

```
/************************************************************************
****/
/*
*/
/*                        Custom load subroutine for DAQ-2501 Card
*/
/*                              Robotic Gymnast Controller
*/
/*
*/
/*       Manufacturing Engineering Centre, Cardiff School of Engineering
*/
/*                 - Copyright 2003 Cardiff University
*/
/*
*/
/************************************************************************
****/
#include<iostream.h>
#include<stdio.h>
#include<float.h>
#include<math.h>
#include<conio.h>
#include<iomanip.h>
#include<fstream.h>
#include<stdlib.h>
#include <windows.h>
#include <string.h>
#include <time.h>
#include "d2kdask.h"
//#include "resource.h"
#define DA_REF_VOL    10.0
#define CHAN_NUM      All_Channels
#define DA_POLAR      DAQ2K_DA_BiPolar
#define DA_REFER      DAQ2K_DA_Int_REF
#define AI_RANGE      AD_B_10_V
#define PI            3.14159
#define max_data      10000
FILE  *infile1;
FILE  *infile2;
FILE  *infile3;
FILE  *infile4;
FILE  *infile5;

void main(void)
{
    I16 AIchcnt = 8, AOchcnt = 4, card_num=0, card_type, card, err, i=0;
    I16 out_data[4], ch_num[4];
    U16 chan_data[8];
    int j;
    long  delay,delay2,JJ;
     float temp[6][8],temp1[3],EE[3];
```

```
    double  Ref1, Ref2, F[2][8], vv[3],T1[2][6], L1[6][6], K1[2][4];
    double v[3],vvv[6], w[2], xx[6],U1_2[2], Uw[2], Ux[2], E1_2[2];
    double u1=0.0,u2=0.0,u3=0.0,Ref1n, Ref2n,Refn1_2[2],pot_gain, test;
    F64 chan_voltage[8], chan_voltage_error0=0.0,
chan_voltage_error1=0.0,
chan_voltage_error2=0.0,chan_voltage_error3=0.0,chan_voltage_initial=0.0;
    F64 theta0;
    F32 out_voltage[4];

    FILE *fp, *fopen();
    float U[max_data][2], X[max_data][6], I[max_data][2], V[max_data][3];
    int count=0;
system( "cls" ); // Clear Screen
//printf("\n\n\n\n");
printf("    ********************&&&&&&**************************\n");
printf("    *                                                  *\n");
printf("    *      ROBOTIC GYMNAST CONTROL SYSTEM              *\n");
printf("    *                                                  *\n");
printf("    *                                                  *\n");
printf("    *                                                  *\n");
printf("    *      CARDIFF SCHOOL OF ENGINEERING               *\n");
printf("    *                                                  *\n");
printf("    *           CARDIFF UNIVERSITY                     *\n");
printf("    *                                                  *\n");
printf("    *                                                  *\n");
printf("    *            HAFIZUL A. ISMAIL                      *\n");
printf("    *                                                  *\n");
printf("    *       &&&&&&&&&&&&&&&&&&&&&&&&&                  *\n");
printf("    *                                                  *\n");
printf("    *                                                  *\n");
printf("    *    PRESS ANY KEY TO STOP THE PROGRAMME           *\n");
printf("    *                                                  *\n");
printf("    *                                                  *\n");
printf("    ********************&&&&&&**************************\n");

/*********************************************************************
****/
/*    This part registers card and performs AD and DA auto-calibration
*/
/*********************************************************************
****/

    card_type = DAQ_2501;
    if ((card=D2K_Register_Card(card_type,card_num))<0)
    {
        printf("Register_Card error=%d\n", card);
        exit(1);
    }


/*********************************************************************
********/
/* This part reads inputs from all channels
*/
/*********************************************************************
********/

    err = D2K_AI_CH_Config (card, CHAN_NUM, AI_RANGE);
```

```
    if (err!=NoError)
    {
       printf("D2K_AI_CH_Config error=%d", err);
       exit(1);
    }

    err = D2K_AO_CH_Config (card, CHAN_NUM, DA_POLAR, DA_REFER,
(F64)DA_REF_VOL);
    if (err!=NoError)
    {
       printf("D2K_AO_CH_Config error=%d", err);
       exit(1);
    }

    for(i=0; i<AOchcnt; i++)
    {
    ch_num[i] = i;
    }

  err = D2K_AO_Group_Setup (card, DA_Group_A, AOchcnt, &ch_num);
    if (err!=NoError)
    {
       printf("D2K_AO_Group_Setup error=%d", err);
       exit(1);
    }

    for(i=0; i<AIchcnt; i++)
        {


// Get AI Hexadecimal value and transform it to voltage //
        err = D2K_AI_ReadChannel (card, i, &chan_data[i]);
        if (err!=NoError)
        {
            printf("D2K_AI_ReadChannel No.%d", i, "error=%d", err);
            exit(1);
        }

        //transform AI value to voltage
        err = D2K_AI_VoltScale (card, AI_RANGE, chan_data[i],
&chan_voltage[i]);
        if (err!=NoError)
        {
        printf("D2K_AI_VoltScale error No.%d", i, "error=%d", err);
            exit(1);
        }
        }
    //theta0=-chan_voltage[0];
    //for(i=1;i<2;i++){chan_voltage[i]=-chan_voltage[i];};

// Potentiometer gain = 10x(2xpi)/30 = 2x3.142/3 (10 turns with +15 V, -
15 V supply).
// Amplifiers gain = 3.7
// Overall gain = 2x3.142/(3x3.7) = 0.566 rad/V
// All chan_voltage values should be multiplied by this gain to convert
the angles from volts to radians.
```

```
// This section reads the Controller and the Observer Parameters from
their files //


infile1=fopen("F.txt","r"); // Controller F for u=-Fx

for (i=0;i<2;i++)
{
fscanf(infile1,"%f %f %f %f %f %f %f %f
\n",&temp[i][0],&temp[i][1],&temp[i][2],&temp[i][3],
       &temp[i][4],&temp[i][5],&temp[i][6],&temp[i][7]);

};
for (i=0;i<2;i++)
{
    for (j=0;j<8;j++)
    {
F[i][j]= (double) temp[i][j];
    }
}

fclose(infile1);

/*for (i=0;i<2;i++)
{
printf("%f %f %f %f %f %f %f %f
\n",F[i][0],F[i][1],F[i][2],F[i][3],F[i][4],F[i][5],
       F[i][6],F[i][7]);
};*/



// This section reads the first choice of Observer parameters from their
files

infile2=fopen("K1.txt","r"); // Observer k1 for v(k+1)=Ev(k)+Hu(k)+KKy(k)

for (i=0;i<2;i++)
{
    fscanf(infile2,"%f %f %f
%f\n",&temp[i][0],&temp[i][1],&temp[i][2],&temp[i][3]);
};

for (i=0;i<2;i++)
{
    for (j=0;j<4;j++)
    {
K1[i][j]= (double) temp[i][j];
    }
}
fclose(infile2);

/*for (i=0;i<2;i++)
{
    printf("%f %f %f %f\n",K1[i][0],K1[i][1],K1[i][2],K1[i][3]);
};*/
```

```
infile3=fopen("L1.txt","r"); // State Estimate xx=L*[y;v]=L1*y+L2*v

for (i=0;i<6;i++)
{
    fscanf(infile3,"%f %f %f %f %f %f
\n",&temp[i][0],&temp[i][1],&temp[i][2],&temp[i][3],
           &temp[i][4],&temp[i][5]);
};

for (i=0;i<6;i++)
{
    for (j=0;j<6;j++)
    {
L1[i][j]= (double) temp[i][j];
    }
}
fclose(infile3);

/*for (i=0;i<6;i++)
{
    printf("%f %f %f %f %f %f
\n",L1[i][0],L1[i][1],L1[i][2],L1[i][3],L1[i][4],L1[i][5]);

};*/



infile4=fopen("T1.txt","r"); // Observer Transform v=T1x

for (i=0;i<2;i++)
{
    fscanf(infile4,"%f %f %f %f %f %f
\n",&temp[i][0],&temp[i][1],&temp[i][2],&temp[i][3],
           &temp[i][4],&temp[i][5]);
};

for (i=0;i<2;i++)
{
    for (j=0;j<6;j++)
    {
T1[i][j]= (double) temp[i][j];
    }
}
fclose(infile4);

/*for (i=0;i<2;i++)

{
printf("%f %f %f %f %f %f
\n",T1[i][0],T1[i][1],T1[i][2],T1[i][3],T1[i][4],T1[i][5]);

};*/



infile5=fopen("EE.txt","r"); // Observer Eigenvalues
```

```
    fscanf(infile5,"%f %f \n",&temp1[0],&temp1[1]);

for (i=0;i<2;i++)
{

EE[i]=temp1[i];
//printf("%f\n",EE[i]);

}
fclose(infile5);



test=4.0; // For adjusting the sampling time




    count=0;
    JJ=0;
    delay2=5.80*1330000*0.7;
    //initialization of the integral

    for (i=0;i<2;i++)
    {
    {w[i]=0.0;};
    };


for (i=0;i<3;i++)
    {
        v[i]=0.0;
    };

    //chan_voltage_error0=chan_voltage[0];
    //chan_voltage_error1=chan_voltage[1];
    //chan_voltage_error2=chan_voltage[2];
    //chan_voltage_error3=chan_voltage[3];
    //printf("%f %f
%f\n",chan_voltage[0],chan_voltage[1],chan_voltage[2]);

    //(double)(chan_voltage[0])=((double)(chan_voltage[0])-
(double)(chan_voltage_error0));
    //(double)(chan_voltage[1])=((double)(chan_voltage[1])-
(double)(chan_voltage_error1));
    //(double)(chan_voltage[2])=((double)(chan_voltage[2])-
(double)(chan_voltage_error2));
    //(double)(chan_voltage[3])=((double)(chan_voltage[3])-
(double)(chan_voltage_error3));

    /*if( fabs((double)(chan_voltage[0]))*0.566 <= 0.01)
        {(double)(chan_voltage[0])=0.0;}
    if( fabs((double)(chan_voltage[1]))*0.105 <= 0.0215)
        {(double)(chan_voltage[1])=0.0;}
    if( fabs((double)(chan_voltage[2]))*0.105 <= 0.0215)
        {(double)(chan_voltage[2])=0.0;}
```

```
   if( fabs((double)(chan_voltage[3]))*4.475 <= 0.08)
        {(double)(chan_voltage[3])=0.0;}*/
    //printf("%f %f
%f\n",chan_voltage[0],chan_voltage[1],chan_voltage[2]);

    Ref1n=(double)(chan_voltage[1]);
    Ref2n=(double)(chan_voltage[2]);
    Ref1=(double)(chan_voltage[1]);
    Ref2=(double)(chan_voltage[2]);
   Ref1=0.0;
   Ref2=0.0;

//(double)(chan_voltage_initial)=((double)(chan_voltage[0]));



       // Observer State Initialisation //

for(j=0;j<2;j++)
{
    for (i=0;i<4;i++)
    {
        if (i==0)

        {(pot_gain=0.566);}

        if (i==3)

            {(pot_gain=4.475);}

        else

        {(pot_gain=0.105);}


        v[j]+=1.0*T1[j][i]*(double)(chan_voltage[i])*(pot_gain); // v=Tx
    }
};

//Refn1_2[1]=0.0;

//E1_2[1]=(Refn1_2[1]-(double)(chan_voltage[2]))*0.105;


for (i=0;i<2;i++)
        {
            E1_2[i]=0.0;

        };


//printf("%f\n",(double)(chan_voltage_initial));


// This is where the do loop starts
```

```
if ( (fabs((double)(chan_voltage[0]))<3.0)&&
            (fabs((double)(chan_voltage[1]))<9.5)&&
            (fabs((double)(chan_voltage[2]))<9.5))

{
do{

//printf("%f %f %f\n",chan_voltage[0],chan_voltage[1],chan_voltage[2]);


// States Resetting //

        for (i=0;i<6;i++)
        {
            xx[i]=0.0;
            vvv[i]=0.0;
        };



        for (i=0;i<2;i++)
        {
            Ux[i]=0.0;
            Uw[i]=0.0;
            vv[i]=0.0;
        };

//printf("%f %f\n",v[0],v[1]);



    // Reference Smoothing //

    Ref1n=0.85*Ref1n+0.15*Ref1;
    Ref2n=0.85*Ref2n+0.15*Ref2;
    Refn1_2[0]=Ref1n;
    Refn1_2[1]=Ref2n;



    for (j=0;j<6;j++)
    {
        for (i=0;i<2;i++)
        {
    vvv[j]+=L1[j][i+4]*v[i];                        // vvv(k)=L2*vn(k)

        }
    //  printf("%f\n",vvv[j]);

    };


for (j=0;j<6;j++)

{
    for (i=0;i<4;i++)
        {
```

```
        if (i==0)

        {(pot_gain=0.566);}
        else
            if (i==3)

            {(pot_gain=4.475);}   // this value if it is checking the
velocity of first pendulum

                else

                {(pot_gain=0.105);}


                xx[j]+=L1[j][i]*((double)(chan_voltage[i])*(pot_gain));
// xe(k)=L1*y(k)


    }

    xx[j]=xx[j]+vvv[j];
//  printf("%f\n",xx[j]);
        X[count][j]=(float)xx[j];

    };
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@//

    // Motor Control Equations (u=-F*xx) //

    for (i=0;i<2;i++)

    {
        for (j=0;j<6;j++)
        {
            Ux[i]+=F[i][j]*xx[j];
        }

        for (j=0;j<2;j++)
        {
            Uw[i]+=F[i][j+6]*w[j];
        }

    };



for(i=0;i<2;i++)
{
    U1_2[i]=-Ux[i]-Uw[i];
};
//  printf("%f %f\n",u1,u2);
     u1=1.0*(U1_2[0]);
    //u1=test;
     u2=1.0*(U1_2[1]);
    //u2=test;

  // Preparation of Motor Controls (adjusting the Op Amp offsets) //
```

```
    /*  if (u1<0.0)
        {u1=u1+0.15;}
        if (u2<0.0)
        {u2=u2+0.15;}
         if (u2>0.0)
        {u2=u2+0.035;}*/
        out_voltage[0]= (F32)(u1);
        out_voltage[1]= (F32)(u2);
        out_voltage[2]= (F32)(u3);


// Sending Motor Controls //

        U[count][0]=(float)out_voltage[0];
        for (i=0;i<AOchcnt;i++)
        {
            if (out_voltage[i]>10) out_voltage[i]=10;
            if (out_voltage[i]<(-10)) {out_voltage[i]=-10;}
        }


        for(i=0; i<AOchcnt;i++)
        {
        out_data[i]= (U16)( ((out_voltage[i]*0.5) +10) / 20 * 4095 );
        }

        err = D2K_AO_Group_Update (card, DA_Group_A, out_data);


        if (err!=NoError)
        {
            printf("D2K_AO_Group_Update error=%d", err);
            exit(1);
        }

        U[count][1]=(float)out_voltage[1];


        // Delay to adjust the sampling time to 25 mse
        for (delay=0;delay<delay2;delay++)
        {
        };
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@//

        // Observer Equation //

        for (j=0;j<2;j++)
    {
        for (i=0;i<4;i++)
    {
    if (i==0)
    {(pot_gain=0.566);}
    else
    {(pot_gain=0.105);}
     if (i==3)
    {(pot_gain=4.475);}
```

```
        vv[j]+=1.0*K1[j][i]*((double)(chan_voltage[i])*(pot_gain));
    //printf("%f\n",pot_gain);
        }



    v[j]=EE[j]*v[j]+vv[j];              // v(k+1)=Ev(k)+K1y(k), H=0

    V[count][j]=(float)v[j];


    };

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@//

    // Integrator Equation //

    for(i=0;i<2;i++)
    {
        E1_2[i]=(Refn1_2[i]-(double)(chan_voltage[i+1]))*0.105;
        //if(fabs(E1_2[i]) <= 0.005)
        //{E1_2[i]=0.0;}

    };
    //E1_2[1]=E1_2[1]/13;
    for (i=0;i<2;i++)
    {
        w[i]=w[i]+10.0*0.01*E1_2[i]; // w(k+1)=w(k)+0.025*(yr-y)
        //w[1]=w[1]+0.01*E1_2[1];
        I[count][0]=(float)w[i];
    };

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@//


// Analogue Inputs Reading (Joint Angles) //

        for(i=0; i<AIchcnt; i++)
        {


            err = D2K_AI_ReadChannel (card, i, &chan_data[i]);
        if (err!=NoError)
        {
            printf("D2K_AI_ReadChannel No.%d", i, "error=%d", err);
            exit(1);
        }

    //transform AI value to voltage //

        err = D2K_AI_VoltScale (card, AI_RANGE, chan_data[i],
&chan_voltage[i]);
        if (err!=NoError)
        {
        printf("D2K_AI_VoltScale error No.%d", i, "error=%d", err);
            exit(1);
        }
```

```
        };
        //for(i=1;i<2;i++){chan_voltage[i]=-chan_voltage[i];};
    //(double)(chan_voltage[0])=((double)(chan_voltage[0])-
(double)(chan_voltage_error0));
    //(double)(chan_voltage[1])=((double)(chan_voltage[1])-
(double)(chan_voltage_error1));
    //(double)(chan_voltage[2])=((double)(chan_voltage[2])-
(double)(chan_voltage_error2));
    //(double)(chan_voltage[3])=((double)(chan_voltage[3])-
(double)(chan_voltage_error3));

    /*if( fabs((double)(chan_voltage[0]))*0.566 <= 0.01)
        {(double)(chan_voltage[0])=0.0;}
    if( fabs((double)(chan_voltage[1]))*0.105 <= 0.0215)
        {(double)(chan_voltage[1])=0.0;}
    if( fabs((double)(chan_voltage[2]))*0.105 <= 0.0215)
        {(double)(chan_voltage[2])=0.0;}
     //if( fabs((double)(chan_voltage[3]))*4.475 <= 0.08)
        //{(double)(chan_voltage[3])=0.0;}*/

    count=count+1;
    //test=-test;

      // printf("%f %f\n",v[0],v[1]);
}while( (kbhit()==0)&&((fabs((double)(chan_voltage[0])))<3.0)&&
            ((fabs((double)(chan_voltage[1])))<9.5)&&
            ((fabs((double)(chan_voltage[2])))<9.5)&&(count<max_data));

    }  // End of IF Condition




printf(" Analogue Inputs:  Ch1     Ch2      Ch3     Ch4     Ch5\n\n");



        // Release the I/O Card //

     D2K_Release_Card(card);

    if ((fp=fopen("motor_voltages.mat","w"))==0){printf("can't open a
file\n");exit(1);}
    for (i=0;i<max_data;i++)
    {
        fprintf(fp,"%.4f %.4f\n",U[i][0],U[i][1]);
    };

    fclose(fp);
    if ((fp=fopen("states.mat","w"))==0){printf("can't open a
file\n");exit(1);}
    for (i=0;i<max_data;i++)
    {
        fprintf(fp,"%.4f %.4f %.4f %.4f %.4f
%.4f\n",X[i][0],X[i][1],X[i][2],X[i][3],X[i][4],X[i][5]);
    };
```

```
    fclose(fp);

    if
((fp=fopen("observer_states_Tacho1_Est23.mat","w"))==0){printf("can't
open a file\n");exit(1);}
    for (i=0;i<max_data;i++)
    {
        fprintf(fp,"%.4f %.4f %.4f\n",V[i][0],V[i][1], V[i][2]);
    };
    fclose(fp);

if
((fp=fopen("integrator_states_Tacho1_Est23.mat","w"))==0){printf("can't
open a file\n");exit(1);}
    for (i=0;i<max_data;i++)
    {
        fprintf(fp,"%.4f %.4f\n",I[i][0],I[i][1]);
    };
    fclose(fp);

printf("%d this count after inc\n",count);
} ; // End of the main Loop
```

B.2. Robogymnast swing-up program

```
%%Robogymnast Swinging%%

%%Lagrange Eqn Matrix%%

M1= (m1*a1) + (m2+m3)*l1;
M2 = (m2*a2)+(m3*l2);
M3= m3*a3;
J1 = I1+(m1*a1*a1)+(m2+m3)*(l1*l1);
J2= I2+(m2*a2*a2)+(m3*l2*l2);
J3=I3+(m3*a3*a3);

%%Lagrange%%

M=[J1+Ip1 (l1*M2)-Ip1 l1*M3; (l1*M2)-Ip1 J2+Ip1+Ip2 (l2*M3)-Ip2; l1*M3
(l2*M3)-Ip2 J3+Ip2];
N=[C1+C2+Cp1 -C2-Cp1 0; -C2-Cp1 C2+C3+Cp1+Cp2 -C3-Cp2; 0 -C3-Cp2 C3+Cp2];
P=-[M1*g 0 0; 0 M2*g 0; 0 0 M3*g];
H=[G1 0; -G1 G2; 0 -G2];


W= [1 0 0;-1 1 0;0 -1 1];
O3= [0 0 0; 0 0 0;0 0 0];
O3x2= [0 0; 0 0; 0 0];
A21= W*inv(M)*P*inv(W);
A22= -W*inv(M)*N*inv(W);
B21= -W*inv(M)*H;
II3=[1 0 0; 0 1 0; 0 0 1];
```

```
C =[II3 O3];
A=[O3 II3;A21 A22];
B=[O32;B21];
[Ad Bd Cd Dd]=robogymnastc2d(A,B,C,D,Ts);
[yn xn un T]=motorvoltage_ori(Ad,Bd,Cd);
ydeg=yn*(180/pi);
xdeg=xn*(180/pi);
subplot(3,1,1), plot(T,un(1,:))
  ylabel('u1(V)')
subplot(3,1,2), plot(T,un(2,:))
  ylabel('u2(V)')
subplot(3,1,3), plot(T,ydeg(1,:))
  ylabel('Theta1(Deg)')
 xlabel('T(s)')
```

B.3. Motor controller subroutine program

```
function [yn xn un Tt] = motorvoltage_ori(Ad,Bd,Cd)

%motor input voltage generation
angle1=1;
angle2=1;
alpha1=1;
alpha2=1;
delta1=0;
delta2=0;

inc_alpha1= 0.6616;
inc_alpha2= 0.1699;
inc_delta1= 5.512/100;
inc_delta2= 5.512/100;
target=(179/180)*pi;
%delta=0;
n=3.142*0.1;
T=0;
x=[0;0;0;0;0;0];
y=[0; 0; 0];
v=10;
for i= 1:1000000;
    if y(1,:)<target;
      if y(1,:)>-target;

        if angle1>2*pi;
           angle1=0;
           alpha1=alpha1+inc_alpha1;
           delta1=delta1+inc_delta1;
        end

        if angle2>2*pi;
           angle2=0;
           alpha2=alpha2+inc_alpha2;
           delta2=delta2+inc_delta2;
        end

     if angle1<=2*pi;
```

```
        T=T+0.025;
        Tt(i)=T;
       u1(i)=3*(alpha1)*sin(angle1);
       u2(i)=2.5*(alpha2)*sin(angle2);

       if u1(i)>=v;
          u1(i)=v;
       end
       if u1(i)<=-v;
           u1(i)=-v;
       end
       if u2(i)>v;
           u2(i)=v;
       end
       if u2(i)<-v;
        u2(i)=-v;
       end


        x=(Ad*x+Bd*[u1(i);u2(i)]);
        y=Cd*x;
        xt=[x];
        yt=[y];
        xn(:,i)=xt;
        yn(:,i)=yt;

       angle1=angle1+(n/delta1);
       angle2=angle2+(n/delta2);
       un=[u1;u2];

        end
     end
  end
```

B.3. Invasive Weed Optimization of swing-up parameters

```
%IWO Program for Robogymnast
%This program is used to calculate optimized values of 4 variables
%(alpha1,alpha2,delta1,delta2)based on Invasive Weed Optimization.
%Original Author: H.A Ismail (GERMAN MALAYSIAN INSTITUTE,CARDIFF
UNIVERSITY)
%02/06/2014
function [yIwo xn un T Tf TfxF SGF]=IWO3(Ad,Bd,Cd)


itermax=5;
iterz=(1:itermax);
iter= iterz-1;
sdAInit= 0.04; %Stand Dev for alpha1
sdAFinal=0.01;
sdA2Init= 0.04; %Stand Dev for alpha2
sdA2Final=0.01;
sdBInit= 0.04;  %Stand Dev for deltas
```

```
sdBFinal=0.01;

NMI=0.001;
MaxPop=500;
Seeds=zeros;
PP1= zeros;
PP2= zeros;
PP3 = zeros;
PP4 = zeros;
rng default;
rand;  % returns the same value as at startup

%Generating Parent Plants
PP1 = 0+(0.7-0).*rand(10,1); %Setting search range for alpha1
PP2 = 0+(0.2-0).*rand(10,1); %Setting search range for alpha2
PP3 = 5+(6-5).*rand(10,1); %Setting search range for delta1
PP4 = 5+(6-5).*rand(10,1); %%Setting search range for delta2


PP=[PP1 PP2 PP4 PP4];  %Initial Population
Seeds=PP;
[sr sc]=size(Seeds);

%Fitness test for parent plants
for k=1:sr;
[yt xn un T Tiwo]=motorvoltage_IWO3(Ad,Bd,Cd,Seeds,k);
yIwo(:,k)=yt.*(180/pi);
Tf(k)=[Tiwo;];
%Tf(: ,k)=Tiwo;
end

%Tfx=[PP1 PP2 PP3 PP4 Tf' yIwo'];
Tfx=[PP1 PP2 PP3 PP3 Tf' yIwo'];
SGF =sortrows(Tfx,5);     %Arrange based on fastest time

%This part is to divide the population into 4 groups based on fitness
  [sr sc]=size(SGF);
   z2=sr/4;
   z2=round(z2);
   z1=1;

%Beginning generation of Weeds

for i=1:itermax;

%Generating Standard Deviatons
%rng default;
%randn;  % returns the same value as at startup

%Spatial Distribution Formula for IWO
 sdA(i)=(((itermax-iter(i))^NMI)/(itermax^NMI))*(sdAInit-
sdAFinal)+sdAFinal;
 sdA2(i)=(((itermax-iter(i))^NMI)/(itermax^NMI))*(sdAInit-
sdAFinal)+sdAFinal;
 sdB(i)=(((itermax-iter(i))^NMI)/(itermax^NMI))*(sdBInit-
sdBFinal)+sdBFinal;

 PP(1,4)=zeros;
```

```
      % Generating next seeds
          for n=1:5  %Number of seeds generated (decreasing with fitness
level)
              if z2<sr
              for m=z1:z2
                 PA1= SGF(m,1) + sdA(i).*randn(6-n,1); %alpha1
                 PA2= SGF(m,2) + sdA2(i).*randn(6-n,1); %alpha2
                 PD1= SGF(m,3) + sdB(i).*randn(6-n,1); %delta1
                 PD2= SGF(m,4) + sdB(i).*randn(6-n,1); %delta2

                 PP=[PP; PA1 PA2 PD1 PD1];    %NewPopulation
              end
              z1=z2+1; %Next Group
              z2=z2+z2;
              end
           end
           if i==1;
           PP(1,:)=[];
           end
           Seeds=PP;
           [sr sc]=size(Seeds);

          %Fitness test
           for k=1:sr;
           [yt xn un T Tiwo]=motorvoltage_IWO3(Ad,Bd,Cd,Seeds,k);
           yIwo(:,k)=yt.*(180/pi);
            Tf(k)=[Tiwo;];
           end

           %Rearranging based on fitness
            TfxF=[PP Tf' yIwo'];
            SGF =sortrows(TfxF,5);
                     [sr sc]=size(SGF);

           %Competive Exclusion
                     if sr>MaxPop
                         SGF((MaxPop+1):end,:)=[];
                         [sr sc]=size(SGF);
                     end

          z2=sr/4;
          z2=round(z2);
          z1=1;

end


 Alpha1= SGF(1,1)
 Alpha2= SGF(1,2)
 Delta1= SGF(1,3)
 Delta2= SGF(1,4)
 Time = SGF(1,5)


end
```

B.4. Fuzzy Logic program for FLIWOH

```
function [u]=Standard_IWOFLC(Jfl,Tfl,JW,TW)
%Part_I :Member-ship Functions
%Creates a new Mamdani-style FIS structure
a=newfis('FL_LQR');



MedJ=median(JW);
          UAvgJ=MedJ+0.25*MedJ;
          LAvgJ=MedJ-0.25*MedJ;
          PCJ1=0.25*MedJ;
          PCJ2=1-PCJ1;
MedT=median(TW);
          UAvgT=MedT+0.25*MedT;
          LAvgT=MedT-0.25*MedT;
          PCT1=0.25*MedT;
          PCT2=1-PCT1;

a=addvar(a,'input','J',[0 1]);
a=addmf(a,'input',1,'Low','trapmf',[0 0 PCJ1 MedJ]);
a=addmf(a,'input',1,'Avg','trimf',[LAvgJ MedJ UAvgJ]);
a=addmf(a,'input',1,'High','trapmf',[MedJ PCJ2 1 1]);

a=addvar(a,'input','T',[0 1]);
a=addmf(a,'input',2,'Low','trapmf',[0 0 PCT1 MedT]);
a=addmf(a,'input',2,'Avg','trimf',[LAvgT MedT UAvgT]);
a=addmf(a,'input',2,'High','trapmf',[MedT PCT2 1 1]);


a=addvar(a,'output','Quality',[0 5]);
a=addmf(a,'output',1,'G','trapmf', [0 0 1 2.5]);
a=addmf(a,'output',1,'Av','trimf', [2 2.5 3]);
a=addmf(a,'output',1,'NG','trapmf', [2.5 3 5 5]);

ruleList=[
1 1 1 1 1
1 2 2 1 1
1 3 3 1 1
2 1 2 1 1
2 2 2 1 1
2 3 3 1 1
3 1 2 1 1
3 2 3 1 1
3 3 3 1 1
];
a = addrule(a,ruleList);


FLin=[Jfl,Tfl];%defining inputs to fuzzy
u=evalfis(FLin,a);%evaluating output a.fis


fuzzy(a)%--- displays the FIS Editor.%
```

```
note it will display FIS editor for %every time step so for 10 sec it
will produce 1001 FIS editors.
mfedit(a)%---- displays the Membership Function Editor.
ruleedit(a)%--- displays the Rule Editor.
ruleview(a)%--- displays the Rule Viewer.
surfview(a)%---- displays the Surface View
```

REFERENCES

Ahmad, S. 2012. A study of search neighbourhood in the bees algorithm. Cardiff University.

Akbari, R. et al. 2012. A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation* 2, pp. 39–52.

Anderson, C.W. 1989. Learning to control an inverted pendulum using neural networks. *Control Systems Magazine, IEEE* 9(3), pp. 31–37.

Anon 2000. *Miriam Webster's Collegiate Encyclopedia*. Miriam Webster.

Asadi, H. et al. 2016. Robust Optimal Motion Cueing Algorithm Based on the Linear Quadratic Regulator Method and a Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* PP(99), pp. 1–17.

Åström, K.J. and Furuta, K. 2000. Swinging up a pendulum by energy control. *Automatica* 36(2), pp. 287–295.

Aström, K.J. and Murray, R.M. 2010. *Feedback systems: an introduction for scientists and engineers*. Princeton university press.

Awrejcewicz, J. et al. 2012. An experiment with swinging up a double pendulum using feedback control. *Journal of Computer and Systems Sciences International* 51(2), pp. 176–182.

Awtar, S. et al. 2002. Inverted pendulum systems: rotary and arm-driven-a mechatronic system design case study. *Mechatronics* 12(2), pp. 357–370.

Bharti, S. and Singh, S.N. 2015. Analytical study of heart disease prediction comparing with different algorithms. *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, pp. 78–82.

Boubaker, O. 2013. The inverted pendulum benchmark in nonlinear control theory: a survey. *International Journal of Advanced Robotic Systems* 10.

Brown, S.C. and Passino, K.M. 1997. Intelligent Control for an Acrobot. *Journal of Intelligent and Robotic Systems: Theory and Applications* 18(3), pp. 209–248.

Cavazzuti, M. 2013. Deterministic Optimization. In: *Optimization Methods SE - 4*. Springer Berlin Heidelberg, pp. 77–102.

Chai, Y. et al. 2009. Mamdani model based adaptive neural fuzzy inference

system and its application. *International Journal of Computational Intelligence* 5(1), pp. 22–29.

Cheng, H. et al. 2013. General swing-up methodology for the vertical three-link underactuated manipulator. *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*, pp. 379–384.

Chiu, C.-H. 2010. The Design and Implementation of a Wheeled Inverted Pendulum Using an Adaptive Output Recurrent Cerebellar Model Articulation Controller. *Industrial Electronics, IEEE Transactions on* 57(5), pp. 1814–1822.

Chua, L.O. and Yang, L. 1988a. Cellular neural networks: applications. *Circuits and Systems, IEEE Transactions on* 35(10), pp. 1273–1290.

Chua, L.O. and Yang, L. 1988b. Cellular neural networks: theory. *Circuits and Systems, IEEE Transactions on* 35(10), pp. 1257–1272.

Darwish, A.H. 2009. Enhanced Bees Algorithm with fuzzy logic and Kalman filtering. Cardiff University.

Das, S. et al. 2015. Multi-objective LQR with optimum weight selection to design FOPID controllers for delayed fractional order processes. *ISA transactions* 58, pp. 35–49.

Deb, K. 2014. Multi-objective optimization. In: *Search methodologies*. Springer, pp. 403–449.

DeJong, G. and Spong, M.W. 1994. Swinging up the acrobot: an example of intelligent control. In: *Proceedings of the American Control Conference*. Univ of Illinois, United States, pp. 2158–2162.

Dorigo, M. and Birattari, M. 2010. Ant Colony Optimization. In: Sammut, C. and Webb, G. eds. *Encyclopedia of Machine Learning SE - 22*. Springer US, pp. 36–39.

Dracopoulos, D.C. and Nichols, B.D. 2012. Swing up and balance control of the acrobot solved by genetic programming. In: *Res. and Dev. in Intelligent Syst. XXIX: Incorporating Applications and Innovations in Intel. Sys. XX - AI 2012, 32nd SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intel.* 32nd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, AI 2012. School of Electronics and Computer Science, University of Westminster, 115 New Cavendish Street, London W1W 6UW, United Kingdom, pp. 229–242.

Dracopoulos, D.C. and Nichols, B.D. 2015. Genetic programming for the

minimum time swing up and balance control acrobot problem. *Expert Systems*, p. n/a-n/a.

Duan, H. et al. 2008. Air robot path planning based on Intelligent Water Drops optimization. *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 1397–1401.

Eberhart, R.C. and Kennedy, J. 1995. A new optimizer using particle swarm theory. In: *Proceedings of the sixth international symposium on micro machine and human science*. New York, NY, pp. 39–43.

Eldukhri, E.E. and Pham, D.T. 2010. Autonomous swing-up control of a three-link robot gymnast. *Pro Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineeringceedings of the* 224(7), pp. 825–833.

Elman, J. 1990. Finding structure in time. *Cognitive Science* 14(2), pp. 179–211.

Engelbrecht, A.P. 2005. *Fundamentals of Computational Swarm Intelligence*. Wiley.

Eom, M. and Chwa, D. 2015. Robust Swing-Up and Balancing Control Using a Nonlinear Disturbance Observer for the Pendubot System With Dynamic Friction. *IEEE Transactions on Robotics* 31(2), pp. 331–343.

Furuta, K. et al. 1984. ATTITUDE CONTROL OF A TRIPLE INVERTED PENDULUM. *International Journal of Control* 39(6), pp. 1351–1365.

Gao, X.Z. et al. 1996. A modified Elman neural network model with application to dynamical systems identification. *Systems, Man, and Cybernetics, 1996., IEEE International Conference on* 2, pp. 1376–1381 vol.2.

Gawthrop, P.J. and Wang, L. 2006. Intermittent predictive control of an inverted pendulum. *Control Engineering Practice* 14(11), pp. 1347–1356.

Ghalenoei, M.R. et al. 2009. Discrete invasive weed optimization algorithm: Application to cooperative multiple task assignment of UAVs. In: Shanghai, pp. 1665–1670.

Ghoreishi, S.A. et al. 2011. Optimal design of LQR weighting matrices based on intelligent optimization methods. *International journal of intellgent Information Processing* 2(1.7).

Gmiterko, A. and Grossman, M. 2009. An n-Link Inverted Pendulum Modeling. *Acta Mechanica Slovaca* 13(3), pp. 22–29.

Godfrey, C.O. and Babu, B.. 2013. *New Optimization Techniques in Engineering*.

Vol. 141. New York: Springer.

Grossimon, P.G. et al. 1996. Sliding mode control of an inverted pendulum. In: *System Theory, 1996., Proceedings of the Twenty-Eighth Southeastern Symposium on*. IEEE, pp. 248–252.

Günther, M. and Wagner, H. 2015. Dynamics of quiet human stance: computer simulations of a triple inverted pendulum model. *Computer methods in biomechanics and biomedical engineering*, pp. 1–16.

Habtie, A.B. et al. 2015. Cellular Network Based Real-Time Urban Road Traffic State Estimation Framework Using Neural Network Model Estimation. *Computational Intelligence, 2015 IEEE Symposium Series on*, pp. 38–44.

Hamam, A. and Georganas, N.D. 2008. A comparison of Mamdani and Sugeno fuzzy inference systems for evaluating the quality of experience of Hapto-Audio-Visual applications. In: *Haptic Audio visual Environments and Games, 2008. HAVE 2008. IEEE International Workshop on*. IEEE, pp. 87–92.

Heyman, D.P. and Sobel, M.J. 2003. *Stochastic models in operations research: stochastic optimization*. Courier Corporation.

Huang, S.-J. and Huang, C.-L. 2000. Control of an inverted pendulum using grey prediction model. *IEEE Transactions on Industry Applications* 36(2), pp. 452–458.

Jain, A.K. et al. 1996. Artificial neural networks: A tutorial. *Computer* 29(3), pp. 31–44.

Jaiwat, P. and Ohtsuka, T. 2014. Real-Time Swing-up of Double Inverted Pendulum by Nonlinear Model Predictive Control. In: *5th International Symposium on Advanced Control of Industrial Processes*.

Jang, J.S.R. 1993. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics* 23(3), pp. 665–685.

Jones, E.R. (Visual N.I. 2004. *An Introduction to Neural Networks*. Visual Numerics, Inc.

Jose, A. et al. 2015. Performance Study of PID Controller and LQR Technique for Inverted Pendulum. *World Journal of Engineering and Technology* 3(2), p. 76.

Kamil, H. 2015. Intelligent model-based control of complex three-link mechanisms. Cardiff University.

Kamil, H.G. et al. 2012. Optimisation of swing-up control parameters for a robot gymnast using the Bees Algorithm. In: *8th International Symposium on Intelligent*

and Manufacturing Systems. pp. 456–466.

Kamil, H.G. et al. 2014. Balancing control of Robogymnast Based on Discrete-time Linear Quadratic Regulator Technique. In: *2014 Second International Conference on Artificial Intelligence, Modelling and Simulation*. IEEE, pp. 137–142.

Kaur, A. and Kaur, A. 2012. Comparison of mamdani-type and sugeno-type fuzzy inference systems for air conditioning system. *International journal of soft computing and engineering* 2(2), pp. 2231–2307.

Kawada, K. et al. 2004. A design of evolutionary recurrent neural-net based controllers for an inverted pendulum. *Control Conference, 2004. 5th Asian* 3, p. 1419–1422 Vol.3.

Khalaf, P. et al. 2015. MULTI-OBJECTIVE OPTIMIZATION OF IMPEDANCE PARAMETERS IN A PROSTHESIS TEST ROBOT. In: *ASME Dynamic Systems and Control Conference*.

Kharola, A. et al. 2016. A Comparison Study for Control And Stabilisation of Inverted Pendulum on Inclined Surface (IPIS) Using PID And Fuzzy Controllers. *Perspectives in Science*.

Klir, G. and Yuan, B. 1995. *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey.

Kolovsky, M.Z. et al. 2012. *Advanced_Theory_of_Mechanisms_and_Machin*. Springer Science & Business Media.

Kuo, A.D. 2007. The six determinants of gait and the inverted pendulum analogy: A dynamic walking perspective. *Human Movement Science* 26(4), pp. 617–656.

Lai, X. et al. 2011. Motion control of underactuated three-link gymnast robot based on combination of energy and posture. *Control Theory & Applications, IET* 5(13), pp. 1484–1493.

Lee, E. and Perkins, J. 2008. Comparison of Techniques for Stabilization of a Triple Inverted Pendulum.

Lee, S. et al. 2015. Robust swing up and balancing control of the acrobot based on a disturbance observer. *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, pp. 48–53.

Li, H. and Yen, V.C. 1995. *Fuzzy sets and fuzzy decision-making*. CRC press.

LIU, D. and YAMAURA, H. 2011. Giant Swing Motion Control of 3-link Gymnastic Robot with Friction around an Underactuated Joint. *Journal of System Design and Dynamics* 5(5), pp. 925–936.

Liu, J. et al. 2008. Design approach of weighting matrices for LQR based on multi-objective evolution algorithm. *Information and Automation, 2008. ICIA 2008. International Conference on*, pp. 1188–1192.

Lozano, R. et al. 2000. Stabilization of the inverted pendulum around its homoclinic orbit. *Systems & Control Letters* 40(3), pp. 197–204.

Luma, N.M.T. and Yaseen, A.O. 2013. Fast Training Algorithms for Feed Forward Neural Network. *Ibn Al-Haitham Journal for Pure & Applied Science* 26, pp. 275–280.

Madivada Hymavathi and Rao, C.S.P. 2012. AN INVASIVE WEED OPTIMIZATION (IWO) APPROACH FOR MULTI-OBJECTIVE JOB SHOP SCHEDULING PROBLEMS (JSSPs). *INTERNATIONAL JOURNAL OF MECHANICAL ENGINEERING AND TECHNOLOGY (IJMET)* 3(3), p. 10.

Manickavelan, K. et al. 2014. Design, Fabrication and Analysis of Four Bar Walking Machine Based on Chebyshev's Parallel Motion Mechanism. *European International Journal of Science and Technology* 3, pp. 65–73.

Marler, R.T. and Arora, J.S. 2004. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization* 26(6), pp. 369–395.

McGrath, M. et al. 2015. The strengths and weaknesses of inverted pendulum models of human walking. *Gait & Posture* 41(2), pp. 389–394.

Medrano-Cerda, G.A. et al. 1995. Balancing and Attitude Control of Double and Triple Inverted Pendulums. *Transactions of the Institute of Measurement and Control* 17(3), pp. 143–154.

Medsker, L.R. and Jain, L.C. 2001. Recurrent neural networks. *Design and Applications*.

Mehrabian, A.R. and Lucas, C. 2006. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics* 1(4), pp. 355–366.

Mohamed, H. et al. 2009. COMBINATION OF OPTIMISATION ALGORITHMS FOR A MULTI-OBJECTIVE BUILDING DESIGN PROBLEM. In: *Eleventh International IBPSA Conference*. Glasgow, pp. 173–179.

Mohan, C. and Deep, K. 2009. *Optimization Techniques*.

Namba, M. and Zhang, Z. 2006. Cellular Neural Network for Associative Memory and Its Application to Braille Image Recognition. *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pp. 2409–2414.

Negnevitsky, M. 2005. *Artificial intelligence: a guide to intelligent systems*. Pearson Education.

De Oca, M.A.M. et al. 2006. A comparison of particle swarm optimization algorithms based on run-length distributions. In: *Ant Colony Optimization and Swarm Intelligence*. Springer, pp. 1–12.

Özbakir, L. et al. 2010. Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation* 215(11), pp. 3782–3795.

Park, K.-H. et al. 2004. Stabilization of a biped robot based on two mode Q-learning. In: *2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand*.

Park, M. et al. 2011. Swing-up and LQR stabilization of a rotary inverted pendulum. *Artificial Life and Robotics* 16(1), pp. 94–97.

Parpinelli, R.S. et al. 2002. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on* 6(4), pp. 321–332.

Parsopoulos, K.E. and Vrahatis, M.N. 2002. Particle Swarm Optimization Method in Multiobjective Problems. In: *Proceedings of the 2002 ACM Symposium on Applied Computing*. SAC '02. New York, NY, USA: ACM, pp. 603–607.

Pham, D. and Liu, X. 1995. *Neural Networks for Identification, Prediction and Control*. Springer.

Pham, D.T. et al. 2006. The Bees Algorithm, A Novel Tool for Complex Optimisation Problems. In: *Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006)*. Oxford: Elsevier, pp. 454–459.

Pham, D.T. and Ghanbarzadeh, A. 2007. Multi-objective optimisation using the bees algorithm. In: *Proceedings of IPROMS 2007 Conference*.

Pham, D.T. and Karaboga, D. 1999. Training Elman and Jordan networks for system identification using genetic algorithms. *Artificial Intelligence in Engineering* 13(2), pp. 107–117.

Pham, D.T. and Liu, X. 1996. Training of Elman networks and dynamic system modelling. *International Journal of Systems Science* 27(2), pp. 221–226.

Poli, R. et al. 2007. Particle swarm optimization. *Swarm intelligence* 1(1), pp. 33–57.

Rahimi, A. et al. 2013. Controller design for rotary inverted pendulum system using particle swarm optimization algorithm. *Electrical and Computer*

Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on, pp. 1–5.

Raj, S. and Kumar, C.S. Q Learning based Reinforcement Learning Approach to Bipedal Walking Control.

Rangaiah, G.P. and Bonilla-Petriciolet, A. 2013. *Multi-Objective Optimization in Chemical Engineering: Developments and Applications*. John Wiley & Sons.

Rubi, J. et al. 2002. Swing-up control problem for a self-erecting double inverted pendulum. *IEE Proceedings - Control Theory and Applications* 149(2), pp. 169–175.

Schrodt, P.A. and Johnson, P.E. 2004. *Mathematical Models of Political Behavior*. Cambridge University Press.

Sehgal, S. and Tiwari, S. 2012. LQR control for stabilizing triple link inverted pendulum system. *Power, Control and Embedded Systems (ICPCES), 2012 2nd International Conference on*, pp. 1–5.

Shah-Hosseini, H. 2009a. Optimization with the nature-inspired intelligent water drops algorithm. *Evolutionary computation*, pp. 297–320.

Shah-Hosseini, H. 2009b. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation* 1(1), pp. 71–79.

Sharma, A. et al. 2012. Automatic Generation Control of Multi Area Power System using ANN Controller. *International Journal of Computer Science and Telecommunications* 3(3), pp. 55–59.

Smith, S.W. 1999. *The Scientist and Engineer's Guide to Digital Signal Processing*.

Souza, L.C.G. and Bigot, P. 2016. An adaptive method with weight matrix as a function of the state to design the rotatory flexible system control law. *Mechanical Systems and Signal Processing*.

Spong, M.W. 1994. Swing up control of the Acrobot. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 2356–2361 vol.3.

Spong, M.W. et al. 2006. *Robot modeling and control*. Wiley New York.

Spong, M.W. and Block, D.J. 1995. The pendubot: A mechatronic system for control research and education. In: *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*. IEEE, pp. 555–556.

Taherkhorsandi, M. et al. 2015. Optimal Sliding and Decoupled Sliding Mode Tracking Control by Multi-objective Particle Swarm Optimization and Genetic Algorithms. In: *Advances and Applications in Sliding Mode Control systems*. Springer, pp. 43–78.

Takashiro, S. and Yoshihiko, N. 1997. Analysis and Control of Underactuated Mechanisms via the Averaging Method. *Proc. of 2nd Asian Control Conference* 1, pp. 273–276.

Tedrake, R. 2009. Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines. *Course Notes for MIT 6.832*.

Toha, S.F. and Tokhi, M.O. 2008. MLP and Elman recurrent neural network modelling for the TRMS. *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*, pp. 1–6.

Uicker, J.J. et al. 2003. *Theory of Machines and Mechanisms*. Oxford University Press.

Wang, H. et al. 2014. Improved Artificial Bee Colony Algorithm and Its Application in LQR Controller Optimization. *Mathematical Problems in Engineering* 2014.

Wang, Y.Z. 2016. Optimal Swing-Up control of an Inverted Pendulum.

Wysocki, A. and Ławryńczuk, M. 2015. Jordan neural network for modelling and predictive control of dynamic systems. *Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on*, pp. 145–150.

Xin, X. and Kaneda, M. 2001. A robust control approach to the swing up control problem for the Acrobot. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on* 3, pp. 1650–1655 vol.3.

Xin, X. and Kaneda, M. 2007a. Analysis of the energy-based swing-up control of the Acrobot. *International Journal of Robust and Nonlinear Control* 17(16), pp. 1503–1524.

Xin, X. and Kaneda, M. 2007b. Swing-Up Control for a 3-DOF Gymnastic Robot With Passive First Joint: Design and Analysis. *IEEE Transactions on Robotics* 23(6), pp. 1277–1285.

Xin, X. and Yamasaki, T. 2012. Energy-Based Swing-Up Control for a Remotely Driven Acrobot: Theoretical and Experimental Results. *IEEE Transactions on Control Systems Technology* 20(4), pp. 1048–1056.

Xiong, X. and Wan, Z. 2010. The simulation of double inverted pendulum control based on particle swarm optimization LQR algorithm. *Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on*, pp. 253–256.

Yamamoto, A. et al. 2015. Behavioral effect of knee joint motion on body's center of mass during human quiet standing. *Gait & Posture* 41(1), pp. 291–294.

Yoshida, K. 1999. Swing-up control of an inverted pendulum by energy-based methods. In: *Proceedings of the American control conference*. pp. 4045–4047.

Youssef, M.S. and Aly, A.A. 2013. Artificial Neural Network Turbulent Modeling for Predicting the Pressure Drop of Nanofluid. *International Journal of Information Technology and Computer Science (IJITCS)* 5(11), p. 13.

Zadeh, L.A. 1965. Fuzzy sets. *Information and Control* 8(3), pp. 338–353.

Zhang, G.P. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50(0), pp. 159–175.