

UNIVERSITY OF THE WITWATERSRAND

MASTERS DISSERTATION

**Nature-Inspired Meta-Heuristic Algorithms in
PID Controller Tuning for Gimbal Stabilization**

Sive Baartman - 723263

PROF. LING CHENG

*A dissertation submitted in fulfilment of the requirements
for the degree of Master of Science*

to the

School of Electrical and Information Engineering

September 2020



The financial assistance of the **Council for Scientific and Industrial Research (CSIR), Optronic Sensor Systems (OSS) department** towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the **CSIR OSS department**.

Declaration

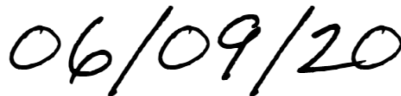
I, **Sive Baartman**, declare that this dissertation titled, “**Nature-Inspired Meta-Heuristic Algorithms in PID Controller Tuning for Gimbal Stabilization**” and the work presented in it are my own. I confirm that:

- ◊ This work was done wholly or mainly while in candidature for a research degree at this University.
- ◊ Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, has been clearly stated.
- ◊ Where I have consulted the published work of others, this is always clearly attributed.
- ◊ Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- ◊ I have acknowledged all the main sources of help.

Signed:



Date:



UNIVERSITY OF THE WITWATERSRAND

Abstract

Engineering and the Built Environment
School of Electrical and Information Engineering

Master of Science

Nature-Inspired Meta-Heuristic Algorithms in PID Controller Tuning for Gimbal Stabilization

by **Sive Baartman** - 723263

Supervisor: PROF. LING CHENG

Inertial stabilization systems are essential in ensuring that the optical system tracks the target of interest and rejects any disturbance. The work presented in this dissertation focuses on optimizing the Proportional-Integral-Derivative (PID) controller in order to ensure that the gimbal used in the chosen inertial stabilization system follows the Line-of-Sight (LOS) rate command input (which represents the target velocity) and rejects disturbance. The main objective of this research was to compare which optimization methods for tuning the PID controller work best for the one-axis gimbal stabilization system. The methods compared are three nature-inspired meta-heuristic algorithms; the Teaching Learning Based Optimization (TLBO) algorithm, the Flower Pollination Algorithm (FPA) and the Genetic Algorithm (GA). This research also involved tuning the parameters of the algorithms themselves in order for the algorithms to optimize the controller. This work also encompasses tuning the common algorithm parameters including the population size and search space bounds, tuning algorithm-specific parameters for each algorithm that requires this, and comparing whether dynamic or static parameters are better suited for the problem instances presented. These parameters were optimized for three different problem instances, which represent different target motions and additional disturbances in the system. It was found that different parameters work best for different problem instances and that this research favoured the TLBO when comparing the algorithm performances overall.

*This dissertation is dedicated to my paternal grandmother **Doreen Baartman** and my maternal grandfather **Milton Mzukhona Mjamekwana**. In achieving their master's degrees during unfavourable circumstances, they displayed exceptional character. Their resilience and belief in education is inspirational.*

Acknowledgements

First and foremost, I thank my supervisor, Professor Ling Cheng, for the guidance, support and encouragement throughout this study. Furthermore, the support, assistance and facilities provided by the University of the Witwatersrand do not go unappreciated, with special regards to Mr Tapiwa Venge, who provided relief during stressful long nights.

I would also like to recognise the invaluable assistance and contribution provided by Mr Nelis Willers during this study. The excellent advice provided by him, and others at the Council for Scientific and Industrial Research (CSIR), proved monumental towards the success of this study.

I acknowledge the financial assistance provided by the CSIR and will forever be grateful for the opportunity provided by the Optronics Sensor System (OSS) department.

Finally, I wish to express my deepest gratitude to my family and friends, and in particular my mom, Fundiswa Mjamekwana, my dad, Sibusiso Baartman and my sister, Lilitha Baartman, whom without even realising, provided me with the strength to continue.

Contents

Declaration	i
Abstract	ii
Dedication	iii
Acknowledgements	iv
Contents	v
List of Figures	ix
List of Tables	xii
Abbreviations	xiv
1 Introduction	1
1.1 Problem Statement	3
1.2 Research Questions	4
1.3 Research Significance	4
1.4 Research Objectives and Scope	5
1.4.1 Objectives	6
1.4.2 Scope	7
1.5 Research Contributions	8
1.6 Dissertation Outline	9
2 Literature Review	11
2.1 The Gimbal Stabilization Problem and the PID Controller	11
2.1.1 Gimbal Stabilization	11
2.1.2 The PID Controller for Gimbal Stabilization	12
2.2 PID Controller Tuning Techniques	14
2.2.1 Classical Controller Tuning Techniques	14
2.2.2 Optimization Controller Tuning Techniques	15
2.3 Meta-Heuristic Algorithms used on Control Systems	15
2.3.1 The Genetic Algorithm	18
2.3.2 The Flower Pollination Algorithm	19
2.3.3 The Teaching-Learning-Based Optimization Algorithm	19
2.4 Parameter Optimization	20
2.4.1 Parameter Tuning and Parameter Control	20

2.4.2	Common Algorithm Parameters	24
2.4.2.1	Population Size	25
2.4.2.2	Stopping Criteria	26
2.4.2.3	Search Space	28
2.4.2.4	Fitness Function	29
2.5	Performance Measures for Evaluating Meta-Heuristic Algorithms in Gimbal Stabilization	32
2.5.1	Performance Measures of Meta-Heuristic Algorithms	33
2.5.2	Performance Measures of Tuning PID Controllers Gimbal Stabilization Control Systems	34
2.6	Theoretical Analysis of Meta-Heuristic Algorithm Performance	35
2.6.1	Convergence Analysis	35
2.6.2	Statistical Analysis	38
2.6.3	Time Complexity of Algorithms	40
2.7	Computational Experimentation	42
2.8	Concluding Remarks	46
3	Background and Preliminaries	47
3.1	Plant Model Details	47
3.1.1	The Rate Gyro Sensor	48
3.1.2	The Gimbal	50
3.1.3	The DC Motor	52
3.1.4	PID Controller	54
3.1.4.1	Proportional Gain	54
3.1.4.2	Integral Gain	55
3.1.4.3	Derivative Gain	55
3.1.5	Added Non-Linearities	57
3.1.5.1	Torque Disturbance due to Friction	57
3.1.5.2	Torque Disturbance due to Base Motion and Mass Unbalance	59
3.2	Algorithm Details	60
3.2.1	The Genetic Algorithm	60
3.2.1.1	Dynamic Parameters	64
3.2.2	The Flower Pollination Algorithm	70
3.2.2.1	Dynamic Switch Probability	71
3.2.2.2	Global Pollination	72
3.2.2.3	Local Pollination	73
3.2.3	The Teaching-Learning-Based Optimization Algorithm	73
3.2.3.1	Teacher Phase	74
3.2.3.2	Learner Phase	74
3.3	Performance Criteria for Evaluating Gimbal Stabilization System	76
3.3.1	Transient Response	76
3.3.2	Fitness Function	78
3.4	Theoretical Analysis Details	79

3.4.1	Statistical Analysis	80
3.4.2	Time Complexity of Algorithms	81
3.5	Method	81
3.5.1	Computational Experimental Design	82
3.5.2	Common Parameters	82
3.5.3	Algorithm-Specific Parameters	83
3.5.4	Problem Instances used in the Experiments	84
3.5.5	Research Environments	85
3.5.6	Work Breakdown Structure	85
4	Parameter Optimization for Nature-Inspired Meta-Heuristic Algorithms	87
4.1	Research Questions	87
4.2	Parameterless Teaching Learning Based Optimization Algorithm Results	88
4.3	Dynamic and Static Flower Pollination Algorithm Results	94
4.4	Dynamic and Static Genetic Algorithm Results	100
4.5	Concluding Remarks	105
4.5.1	Conclusions	105
4.5.2	Suggestions for Future Practitioners	106
5	Robustness Investigation of Nature-Inspired Meta-Heuristic Algorithms	108
5.1	Research Questions	108
5.2	Dynamic Input	109
5.2.1	Parameterless Teaching Learning Based Optimization Results	109
5.2.2	Dynamic and Static Flower Pollination Algorithm Results .	112
5.2.3	Dynamic and Static Genetic Algorithm Results	115
5.3	Additional Non-Linearities	119
5.3.1	Parameterless Teaching Learning Based Optimization Results	119
5.3.2	Dynamic and Static Flower Pollination Algorithm Results .	123
5.3.3	Dynamic and Static Genetic Algorithm Results	126
5.4	Concluding Remarks	131
5.4.1	Dynamic Input Conclusions	131
5.4.2	Additional Non-Linearity Conclusions	131
5.4.3	Suggestions for Future Practitioners	132
6	Performance Evaluation	133
6.1	Research Questions	133
6.2	Comparison of Algorithm Performance using Fitness Value and Behavioural Response	133
6.3	Statistical Analysis	138
6.4	Convergence	139
6.5	Time Complexity	141
6.6	Concluding Remarks	144

6.6.1	Conclusions	144
6.6.2	Suggestions for Future Practitioners	145
7	Conclusions and Future Work	146
7.1	Research Summary	146
7.2	Research Contributions	147
7.3	Research Conclusions	147
7.4	Possible Future Work	148
A	Model Validation	150
B	Time Complexity Analysis	153
	References	157

List of Figures

1.1	Applications of optical systems. The left image shows a guided missile [1], the middle image shows a fighting vehicle with optical systems [2], and the right image shows a Hubble space telescope used to point at distant stars and galaxies [3]	1
1.2	Illustration adapted from [4] showing the two optimization problems found in applying meta-heuristics on a problem with the control flow (right) and information flow (left)	3
2.1	Diagram showing how the experiments were conducted	45
3.1	Block diagram showing the model of the simple gimbal control system adapted from [5]. The Figure shows V_{in} , which is the command voltage taken from the controller, T_m is the output motor torque, ω_{out} is the angular velocity of the gimbal, and ω_m is the measured angular velocity.	48
3.2	Block diagram showing the rate gyro model adapted from [5] . . .	49
3.3	CAD adaptation of the camera gimbal that was used in this study taken from [6]	51
3.4	Block diagram showing the DC motor adapted from [5]	53
3.5	PID controller structure adapted from [7]	54
3.6	PID controller structure with an added low-pass filter	56
3.7	A representation of the relationship between friction force and velocity adapted from [8]. (a) Coulomb, viscous and static friction. (b) Stribeck friction model	58
3.8	Flowchart showing dynamic GA adapted from [9]	64
3.9	Diagram showing the change of the switch probability value p as the iteration number progresses	72
3.10	Basic figures of merit of a control system for the time-domain adapted from [10]	77
3.11	Work breakdown structure describing tasks required for this study .	86
4.1	Bar graph showing the solution and fitness values for each algorithm design for the TLBO	90
4.2	The step response of the algorithm designs for the TLBO	92
4.3	Bar graph showing the solution and fitness values for each algorithm design for the static and dynamic FPA	96
4.4	The step response of the algorithm designs for the static and dynamic FPA	98

4.5	Bar graph showing the solution and fitness values for each algorithm design for the static and dynamic GA	102
4.6	The step response of the algorithm designs for the static and dynamic GA	103
5.1	Bar graph showing the solution and fitness values for each algorithm design for TLBO on the control system on the second problem instance	110
5.2	The ramp response of the algorithm designs for the TLBO	111
5.3	Bar graph showing the solution and fitness values for each algorithm design for static and dynamic FPA on the second problem instance	114
5.4	Bar graph showing the fitness values for each algorithm design for FPA on the second problem instance	114
5.5	The ramp response of the algorithm designs for the FPA	115
5.6	Bar graph showing the solution and fitness values for each algorithm design for GA on the second problem instance	117
5.7	Bar graph showing the change of fitness value with changing algorithm design for the GA algorithm on the second problem instance	117
5.8	The ramp response of the algorithm designs for the GA	118
5.9	Bar graph showing the solution and fitness values for each algorithm design for TLBO on the last problem instance	120
5.10	Bar graph showing the time domain responses with changing algorithm design for the TLBO on the last problem instance	120
5.11	The step response of the algorithm designs for the TLBO on the last problem instance	121
5.12	The step response showing the TLBO algorithm without the voltage constraint of the motor	122
5.13	Bar graph showing the solution and fitness values for each algorithm design for FPA and the last problem instance	124
5.14	Bar graph showing the change of fitness value with changing algorithm design for the FPA algorithm on the last problem instance	124
5.15	Bar graph showing the change of time-domain responses with changing algorithm design for the static and dynamic FPA on the last problem instance	125
5.16	The step response of the algorithm designs for the FPA with the last problem instance	125
5.17	The step response showing the Dynamic FPA algorithm without the voltage constraint of the motor	126
5.18	Bar graph showing the solution and fitness values for each algorithm design for FPA on the last problem instance	128
5.19	The change of fitness value with changing algorithm design for the static and dynamic GA on the last problem instance	128
5.20	The change of the time domain responses with changing algorithm design for the static and dynamic GA algorithm on the last problem instance	129

5.21	The step response of the algorithm designs for the GA on the last problem instance	129
5.22	The step response showing the dynamic GA algorithm without the voltage constraint of the motor on the control system	130
6.1	The step response comparing the algorithm performance for the plant with step input and no additional friction	134
6.2	The step response comparing the algorithm performance for the control system with step input	135
6.3	The comparison of the ramp response of the algorithm performance	136
6.4	The comparison of the step response of the algorithm performance on the control system with additional non-linearity	137
6.5	Run-length distributions showing the convergence performance for different Simulation conditions	140
6.6	Bar graph comparing the computational time required for all three runs	142
A.1	Diagram showing the results for the step response system shown in Figure 11 of [5]	150
A.2	Diagram showing the simulated results for the step response of the system from the model	151
B.1	Time complexity of FPA	153
B.2	Time complexity of TLBO	154
B.3	Time complexity of Static GA	154
B.4	Time complexity of Dynamic GA	155

List of Tables

2.1	Meta-heuristic algorithms for tuning PID controllers in different applications	16
3.1	Gyro information	49
3.2	Gimbal performance specifications	52
3.3	Motor specifications	53
3.4	Friction values taken from [11]	58
3.5	Parameters of the GA compared	63
3.6	2-level factorial design bounds	83
3.7	Factorial design	83
4.1	Descriptive statistical results illustrating the changes in the fitness value for the different algorithm designs	88
4.2	Time-domain results for the TLBO for the first problem instance . .	89
4.3	Descriptive statistical results for the FPA for the first problem instance	94
4.4	Time-domain results for the FPA for the first problem instance . . .	95
4.5	Descriptive statistical results for the GA for the first problem instance	101
4.6	Time-domain results for the GA for the first problem instance . . .	101
5.1	Descriptive statistical results for the TLBO for the second problem instance	109
5.2	PIDN solution results for the TLBO for the second problem instance	110
5.3	Descriptive statistical results of the fitness value for the FPA for the second problem instance	113
5.4	Time domain results for the FPA for the second problem instance .	113
5.5	Descriptive statistical results of the fitness value for the GA for second problem instance	116
5.6	Time-domain results for the GA for the second problem instance . .	116
5.7	Descriptive statistical results for the TLBO for the last problem instance	119
5.8	Time-domain results for the TLBO for the last problem instance .	119
5.9	Descriptive statistical results for the FPA for the last problem instance	123
5.10	Time-domain results for the FPA for the last problem instance . .	123
5.11	Descriptive statistical results for the GA on the last problem instance	127
5.12	Time-domain results for the GA for the last problem instance . . .	127

6.1	Comparison of time-domain results for a step input	134
6.2	Comparison of fitness value and algorithm solutions	136
6.3	Comparison of time-domain results for a step input and added non- linearity	137
6.4	Data for the Friedman Test	138
6.5	Friedman Test results	138
6.6	Posthoc comparison of absolute difference among mean ranks	138
6.7	Posthoc comparison of p-values	139
6.8	Posthoc comparison of p-values $< \alpha$	139
6.9	Time complexity result for each algorithm analysed in appendix B .	143
A.1	Showing the maximum, minimum and average fitness value at each iteration value.	152

Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization Algorithm
AGC	Automatic Generation Control
AI	Artificial Intelligence
ANOVA	ANalysis Of VAriance
AVR	Automatic Voltage Regulator
BA	Bat Algorithm
BFO	Bacteria Foraging Optimization Algorithm
CI	Computational Intelligence
CIA	Computational Intelligence Approach
CS	Cuckoo Search Algorithm
DE	Differential Evolution Algorithm
DOE	Design Of Experiments
EA	Evolutionary Algorithms
FA	Firefly Algorithm
FOPDT	First Order Plus Dead Time
FOPID	Fractional Order Proportional Integral Derivative
FPA	Flower Pollination Algorithm
GA	Genetic Algorithm
HBA	Hybrid Bat Algorithm
HVAC	Heat Ventilating and Air Conditioning
HPD	Healthy Population Diversity
LFC	Load Frequency Control
LOS	Line Of Sight
LQG	Linear Quadratic Regulator

LQR	L inear Q uadratic G aussian
LTl	L inear T ime I nvariant
MIMO	M ultiple I nput M ultiple O utput
MRAC	M odel R eference A daptive C ontrol
NFL	N o F ree L unch
NM	N elder M ead
NP	N on-deterministic P olynomial
PID	P roportional I ntegral D erivative
PSO	P article S warm O ptimization Algorithm
QFT	Q uantitative F eedback T heory
SA	S imulated A nnealing Algorithm
SI	S warm I ntelligence
SISO	S ingle I nput S ingle O utput
SPD	S tandard P opulation D iversity
TLBO	T eaching L earning B ased O ptimization Algorithm
UAV	U nmanned A erial V ehicle
WBS	W ork B reakdown S tructure

CHAPTER 1

Introduction

The human strive to perfection is expressed in
optimisation theory

Author Unknown

Moving vehicles, hand-held or ground-mounted systems make use of optical systems such as infra-red cameras, radars, and lasers [12]. Examples of these applications include navigation and tracking systems, guided missiles, and astronomical telescopes, as seen in Figure 1.1 [13].



FIGURE 1.1: Applications of optical systems. The left image shows a guided missile [1], the middle image shows a fighting vehicle with optical systems [2], and the right image shows a Hubble space telescope used to point at distant stars and galaxies [3]

These optical systems mainly function by stabilizing¹ the sensor's line-of-sight (LOS) towards a target of interest. Thus, the ability of the system to reject disturbances and accurately track a target is imperative. There are instances, such as when the optical system is mounted on a moving base, that creates a stabilization challenge [14].

A gimbal stabilization system (sometimes referred to as an inertial stabilization platform (ISP) and thus these terms will be used interchangeably) is a solution

¹Stabilization in this context refers to following a commanded rate input [12], and not other definitions of stabilization in other fields.

for this problem. How the ISP performs generally is categorized into two different groups. The first group deals with command tracking, i.e. how well the motion follows the commanded input. The second group deals with how well the disturbance is rejected [11]. Accurately following the commanded input means rejecting the disturbance. The ISP system makes use of a controller, amongst other elements, to stabilize the sensor towards the target and reject disturbance.

How the controller performs highly depends on how it is tuned. A well-tuned controller can have excellent disturbance rejection, good tracking response without being easily influenced by noise [15]. Classical tuning methods have proven to possess limitations because compared to intelligent methods, the classical tuning methods do not produce desired results [15] [16]. Because of the limitations that classical tuning methods possess, intelligent tuning methods, such as meta-heuristic algorithms, are a promising alternative to tuning the controller.

Nature-inspired meta-heuristic algorithms can aid in solving problems that include non-linear constraints, multiple objectives, and dynamic non-linear properties. However, there are over 100 different types of algorithms and new (or deviations of) meta-heuristic algorithms are continually being developed without understanding the underlying mechanisms of those that already exist [17]. Not much work has gone into finding a method for selecting a suitable one(s) for specific applications and understanding why certain algorithms are suitable for certain applications. Thus, choosing the algorithm for the required application is highly dependent on the experience and mathematical knowledge of the algorithm practitioner [18]. These algorithms have the powerful potential to solve complex problems, although, not having a good understanding of how the underlying mechanisms work can make using nature-inspired meta-heuristic algorithms a less practical and accessible solution. Because the complexity of problems which researchers attempt to solve is ever-increasing, improving these algorithms, and having a good understanding of these algorithms is essential.

Because these algorithms are powerful, many researchers make use of them and have to either conduct various experiments for configuring and analysing them.

Some researchers have to guess which algorithm would work best to suit their application. Using the algorithms this way limits their performance, and the experimental work can be time-consuming. Figure 1.2 shows the two essential factors to consider when applying meta-heuristic algorithms. These two factors are parameter tuning of the algorithm and solving the problem presented by the application.

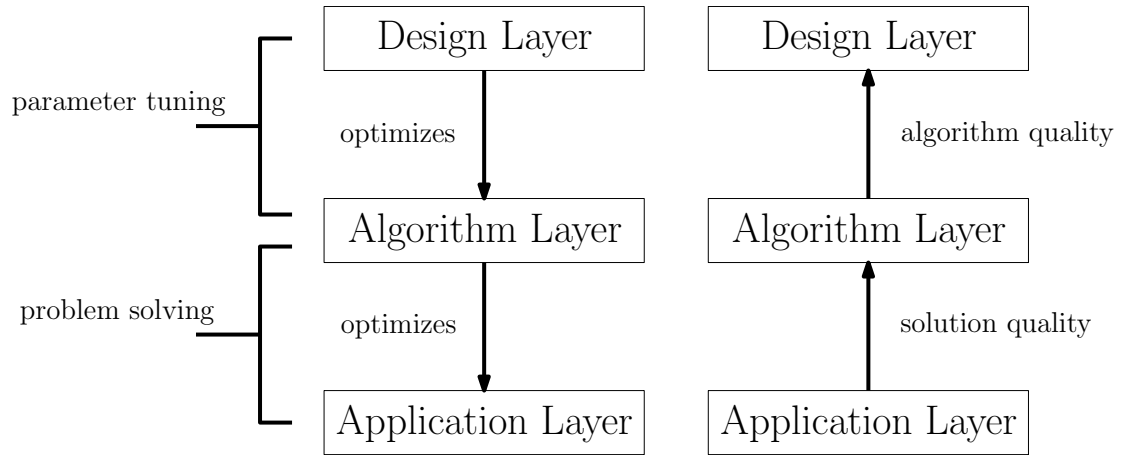


FIGURE 1.2: Illustration adapted from [4] showing the two optimization problems found in applying meta-heuristics on a problem with the control flow (right) and information flow (left)

The left diagram in Figure 1.2 illustrates how parameter tuning of the algorithm optimizes its design, and how the algorithm is then used for optimization in the problem solving of the application. The right diagram shows how the application determines the quality of the solution, which then determines the algorithm quality, which indicates the performance of the algorithm design. How the parameters of the algorithm itself or the application of interest affect the performance, for example, is important and should be understood to be able to choose the appropriate algorithm [19].

1.1 Problem Statement

The one-axis gimbal system used in this study as the application of interest was developed computationally and tested. The controller in the gimbal system is

imperative for stabilization, and for tracking the command input. However, tuning the controller using meta-heuristic techniques to improve the performance of the proposed one-axis gimbal stabilization system has not been investigated. Furthermore, the lack of investigation of how the parameters of the algorithm and application affect the performance makes choosing which algorithm to use for the characteristics displayed in the application difficult. Investigating the effects of the algorithm and application parameters is essential in order for algorithms to be seen as a more practical solution and to optimize their performance. It is also crucial because classical tuning techniques seem to possess limitations that meta-heuristic algorithms could be able to solve.

1.2 Research Questions

From the problem statement above, the research questions are as follows:

- *How does applying meta-heuristic algorithms to tune a Proportional-Integral-Derivative (PID) controller, namely the Flower Pollination Algorithm (FPA), the Genetic Algorithm (GA), and the Teaching-Learning-Based Optimization Algorithm (TLBO) affect the one-axis inertial stabilization system? How do these algorithms compare in performance?*
- *Do the parameters of the algorithms and the characteristics of the one-axis inertial stabilization system affect the performance? If so, how?*

1.3 Research Significance

Meta-heuristic algorithms are continually being developed and are growing in numbers. Rather than creating yet another algorithm, the significance of this research lies in investigating how the parameters of the algorithm affect the performance of the control system. Furthermore, the significance also lies in showing what characteristics of the control system are essential in the performance with the hopes that this will aid the next researcher in choosing an algorithm appropriately for

a similar type of application with similar characteristics. In attempting to investigate how these parameters affect the performance of the algorithms and control system, this research gives an understanding of the underlying mechanisms of the algorithms.

Secondly, the application of the meta-heuristic algorithms for tuning the PID controller on a real-world application such as the gimbal stabilization system, rather than benchmark problems is significant to prove that these algorithms can aid in real-world scenarios. Real-world problems come with real-world non-linearities, and it is important to observe whether these algorithms would be able to perform in these scenarios.

PID controllers are continually being used (approximately 95% of the controllers in the process industry are PID controllers) due to their ease of implementation. Thus improving on their performance makes this research practical and beneficial for the many that make use of this controller. Various applications use gimbal stabilization systems. Therefore, this research can contribute to improving such applications.

1.4 Research Objectives and Scope

The focus of this research was two-fold. The focus included investigating the performance of each algorithm and comparing the performance of these algorithms with each other. Hooker [20] explained the difference between competitive and scientific testing. Scientific testing of meta-heuristic algorithms focuses on using scientific methods in experimentation to gain insights into the working mechanisms of the algorithm. In contrast, competitive testing is more concerned with obtaining an algorithmic set-up that performs better than a specific benchmark [4] [20]. This research combined these testing methods.

1.4.1 Objectives

The first two objectives were related to scientific testing because they address the effects of changing the parameters and the problem instances to gain insights into each algorithm. The last objective was related to competitive testing because it compares the algorithm performances to each other.

- The first objective identified the possible effects the algorithm parameters had on the performance, and whether this effect was constant for all algorithms. Determining these effects included observing the common-algorithm parameter changes, and algorithm-specific parameter changes. Algorithm-specific parameters, in this research, were either static or dynamic. Investigating the parameter effects aids in understanding the underlying working mechanisms of the algorithms, and thus can contribute to knowing which algorithms to choose for different applications.
- The second objective was to determine the robustness of the algorithms. Robustness can be measured using two factors; robustness to changes in problem instances, and robustness to change in parameter values [4]. Robustness to changes in parameter values was addressed in the first objective. However, this objective was to observe how the changes in the application affect the performance of the algorithms. Thus, the three algorithms were applied on three variations of the gimbal control system: (1) a gimbal stabilization control system with step input where the step input represents a constant target velocity, (2) a gimbal stabilization control system where the ramp input represents a target velocity with constant acceleration, and (3) a control system with added non-linearity and a step input where this application tests whether the LOS rate can follow the target velocity with additional non-linearities in the system.
- The final objective was to compare the performance of the algorithms with their chosen ‘best’ parameters, which performs best in the different variants of the gimbal control system.

1.4.2 Scope

A gimbal control system usually consists of two loops; the rate of the LOS which aids in stabilization and rate disturbance rejection, and the LOS angle control system loop which focuses more on tracking the target of interest. The current study focused on the rate control system loop.

Many studies compare the performance of the meta-heuristic algorithms with the classical PID control tuning methods such as the Ziegler-Nichols (ZN) tuning method. The current study will not be doing this for two main reasons. The first is that the ZN method is used on systems where the step response curve is S-shaped so that the delay time and time constant can be determined to approximate a first order plus dead time (FOPDT) system equation. This then allows the PID controller gains to be determined from a set of rules. The system in this research did not exhibit an S-shaped step response curve so these rules could not be used. The second way to determine the PID controller gains using ZN method is to observe how the system responds when increasing the proportional gain until sustained oscillations, while setting the integral and derivative gains to infinity and zero, respectively. However, this could not be implemented because of the constraints of the motor voltage in the system that did not allow the proportional gain to be increased until the required sustained oscillations.

Ogata [21] states that the ZN tuning method gives an educated guess on the PID controller gains, rather than giving the final settings. This emphasises the importance of investigating other techniques because the classical tuning techniques would still require manual tuning after implementation, which is time-consuming. This also shows why it would be hard to compare the optimization methods to the classical tuning methods, as the researcher would use results obtained from the algorithms for the manual tuning of the PID controller gains after using the ZN method, which would not make for a fair comparison.

Another factor to consider is real-time controller tuning. Real-time controller

tuning can mean different things. It can mean finding the optimal values of controller gains continuously while the target motion changes and environmental disturbances vary. It can also mean including hardware in the loop such that the results obtained are directly from system hardware. In the current study, using meta-heuristic algorithms to tune the controller is not referred to as real-time application as the algorithms produce the optimized parameter gains after a specified iteration number. Also, the study will not include hardware. There are valuable benefits for implementing real-time tuning applications; however, it was not part of the scope of this study.

There are many different types of controllers used to control a gimbal stabilization system. However, this study will only use the classical Proportional, Integral and Derivative controllers. Furthermore, various applications use different PID controller structures. However, this study will only partake in using the conventional parallel form of the PID controller with an added N filter coefficient on the Derivative controller.

The gimbal platform was assumed to be rigid and not flexible. It was also assumed that the rotation of the earth has a negligible impact on the gimbal motion. This simplified the gimbal equations of motion derivation. The rate gyro has drift and drag [22] [23], which is important over longer time runs, however since these experiments were conducted over a short time (1.5 seconds), this will be ignored.

1.5 Research Contributions

The contributions of this research are as follows:

- A comprehensive application of nature-inspired meta-heuristic algorithms on an existing real-life problem. The three algorithms chosen are from different categories, and thus this application contributes to a deeper understanding of the underlying mechanisms of the different categories of the algorithms.

- Parameter optimization of the three algorithms chosen to contribute to knowing how parameters affect algorithm performance in the chosen existing real-life problem. Chapter 4 shows this contribution.
- Observing how the algorithms perform with changes in the problem instance as a method of evaluating how changes in the problem affect algorithms and parameters. Chapter 5 shows this contribution.
- The computational time complexity analysis for these algorithms is made, and this contributes to further understanding of how these algorithms perform and the computational costs required. Chapter 6 shows this contribution.
- The main purpose of this research is to aid future meta-heuristic practitioners on how to choose algorithms for this kind of problem. Thus, the final contribution is the suggestions for future practitioners obtained from each experiment shown on Chapters 4, 5 and 6.

1.6 Dissertation Outline

The dissertation is arranged as follows: Chapter 2 gives the literature survey of the research. This includes looking at the current state of the literature on PID control and the gimbal stabilization problem, tuning techniques for the PID controller which then leads to meta-heuristic algorithms and how they are applied to the problem. Since this type of research requires computational experimentation, the literature survey also highlights some important considerations for that type of experimentation. The parameter optimization for the algorithms along with the performance measures are crucial factors, and thus, the literature survey elaborated on them. Chapter 2 ends with a survey of the theoretical analysis of meta-heuristic algorithms, followed by concluding remarks.

Chapter 3 gives some background and preliminary technical theory required for this research, including observing the plant model details, algorithm details, and the performance criteria used.

This chapter is then followed by Chapters 4, 5, and 6, which are the contributing chapters for this dissertation. Chapter 4 reports on the results obtained from the parameter optimization experiments. This chapter is followed by Chapter 5, which reports on the results obtained from the robustness experiments. Chapter 6 then discusses and compares the performance of the algorithms using theoretical analysis.

The dissertation concludes in Chapter 7. This chapter includes a research summary, briefly stating the research contributions before detailing the research conclusions. The chapter ends by presenting ideas for future work.

CHAPTER 2

Literature Review

This chapter serves as an overview of the current standing of literature to justify important decisions regarding this research.

It begins by observing the Proportional-Integral-Derivative (PID) controller on the gimbal stabilization problem, including looking at how this controller is tuned and gives an understanding of why this controller was chosen for this study by reviewing similar work. The tuning techniques of the PID controller is surveyed, including classical and optimization techniques. This chapter justifies choosing the three algorithms used in this study.

The factors of parameter optimization are then discussed, including the reasons for choosing certain values for parameters, and fitness function structure. How performance is evaluated, including some theoretical aspects, is also reviewed in this chapter. This chapter ends by discussing the essential factors of computational experimentation.

2.1 The Gimbal Stabilization Problem and the PID Controller

2.1.1 Gimbal Stabilization

Inertial Stabilization Platforms (ISP) generally started being utilised approximately 100 years ago [12]. Various applications such as airborne vehicles (including

missiles, Unmanned Ariel Vehicles (UAVs) and satellites), and non-airborne vehicles (such as ground vehicles, submarines and ships) contain the ISPs. ISPs are featured because they consist of optical sensors used for tracking military targets, providing high-resolution imagery, and stabilize communication antennas [12].

There are alternative methods, other than using a gimbal, which can form part of an ISP and can be used to stabilize an optical sensor. An example of such methods is the incorporation of mechanisms such as moving platforms and springs into unstable systems to combat the effect of disturbances [24]. Airborne vehicles such as helicopters or missiles commonly make use of gimbal stabilization systems [24].

The gimbal stabilization system can undertake different electro-mechanical configurations. In some configurations, the optical sensor is directly mounted on the gimbal. In contrast, other configurations consist of mirrors or other optical elements mounted onto the gimbal, with the optical sensor mounted onto the vehicle. What is common in these configurations are some of the elements to make the system. These elements include motors or actuators, gimbal, gyroscopes or rate sensors, optical sensor or payload, and a controller [12]. The gimbal stabilization system in this current study used the PID controller.

2.1.2 The PID Controller for Gimbal Stabilization

The ISP problem has used various types of controllers. Some have used the standard classical controllers (i.e. PID control) [5] [14] [25], whereas others have used robust control methods [26] [27] [28]. Baskin et al. [26] stated that using classical controllers is time-consuming because finding a controller that satisfies both the stability and performance is an iterative process. That is why the authors purely made use of robust controller design, namely the Linear Quadratic Gaussian (LQG)/ Linear Quadratic Regulator (LQR) and H-infinity (H_∞). Bujela [29] compares using PID control with robust control algorithms (such as H_∞ , Quantitative Feedback Theory (QFT), and adaptive control) to observe which control

method is suitable for the gimbal stabilization system. He correctly points out that the need for using robust methods is because these methods can cope with changing environments in plants, which the PID controller may not be able to.

However, the importance of using the PID controller is due to it being the controller used most in industry (approximately 95% [30]) because of its ease of implementation and understanding. The PID controller is then a practical choice to optimize. The performance of the PID controller highly depends on how it is tuned. Many applications make use of the classical PID controllers and combine other controller algorithms to choose the parameters of these classical controllers. Zhou et al. combined the Model Reference Adaptive Control (MRAC) method with the PID method for non-linear disturbance rejection in an aerial ISP [31].

The PID controller used the MRAC method so that the ISP system can reject the non-linear and time-varying mass unbalance torque disturbance. This technique was chosen because it does not require online identification of the mathematical model of mass unbalance torque. Adaptive controllers are useful when the parameters of the plant are unknown or change with time [32]. Khayatian and Aghaee [33] compared the performance of a standard PD controller tuning method with a PD controller adjusted by an adaptive controller to observe which controller performs best in optimizing a two-axis gimbal stabilization system with unknown dynamic parameters.

Other applications made use of alternative methods for choosing the parameters of the PID controller. Caponetto and Xibilia compared the performance of the standard PID controller with a Fractional Order PID (FOPID) controller where the Genetic Algorithm (GA) designed the parameters for the stabilization of a one-axis gimbal [34]. Rajesh et al. chose to implement the standard PID controller but tuned it by using the Particle Swarm Optimization (PSO) and GA for the stabilization of a three-axis gimbal system [25]. Abdo et al. applied fuzzy logic for the choosing of the parameters of the PID controller for the stabilization of a two-axis gimbal [14]. Zhou et al. optimized the PID control parameters for

the two-axis ISP platform by using a method based on the co-simulation of a mechatronic system [35].

Rather than implementing the standard PID controller configuration, other applications implement the cascade form of the classical controllers and use standard tuning techniques. The study conducted by Abdo et al. made use of a cascade controller for the control of a one-axis gimbal system [11].

The studies above mentioned show that PID controllers are efficient for this application. However, the efficiency and performance of these controllers highly depend on the implementation and tuning of the controller gains. Thus, rather than choosing a different controller, this research focused on how to improve the PID controller by implementing meta-heuristic algorithms as a possible method of improving the PID controller performance.

2.2 PID Controller Tuning Techniques

As stated previously, the performance of the feedback control system highly depends on the tuning of the controller [36]. If the controller is not tuned appropriately, the closed-loop system becomes unstable [36]. There are different types of tuning methods which are classical/conventional and optimization techniques.

2.2.1 Classical Controller Tuning Techniques

In classical techniques, the controller parameters highly depend on the assumptions made of the plant and its desired output [36] [37]. These techniques are popular since they are easy to implement and use. However, due to the assumptions required for the technique to work, the preferred outcome can be challenging to obtain in real-life scenarios where those assumptions are not relevant [36]. There are many classical techniques. Some include Skogestad's method, the Internal Model Control method, the Lambda tuning and the Amigo method [38]. However, the most common are the Ziegler-Nichols, Cohen-Coon method, Manual Tuning

method and using PID Tuning Software methods [39]. The Ziegler-Nichols tuning method being the most popular of the four.

These techniques are also time-consuming [40], and do not handle non-linearities very well [36]. Optimization techniques provide better performance when faced with numerous performance specifications [16]. These techniques contribute to the system's fast responses while decreasing overshoot and presenting satisfactory steady-state errors [15].

2.2.2 Optimization Controller Tuning Techniques

Optimization techniques are classified into two categories; classical optimization techniques and advanced optimization techniques [36]. Classical optimization techniques include Calculus Methods, Linear Programming, Dynamic Programming and Stochastic Programming.

The advanced optimization methods are those that use Artificial Intelligent (AI) techniques. The importance of these techniques lies in their abilities to overcome the weaknesses of the classical optimization techniques [36]. One such weakness is that classical optimization techniques are only applicable to functions that are continuous and differentiable. In contrast, advanced optimization techniques are applicable even on discontinuous and non-differentiable functions [36]. These advanced optimization techniques include the nature-inspired meta-heuristic algorithms and other AI techniques such as neural networks, and fuzzy logic.

2.3 Meta-Heuristic Algorithms used on Control Systems

The motivation behind using meta-heuristic algorithms for tuning the PID controller lies in the categorisation of the PID control problem to be NP-hard, according to [41]. Heuristic and meta-heuristic algorithms are efficient solutions to problems characterised as NP-hard or NP-complete [42]. NP is an abbreviation for Non-deterministic Polynomial. A problem classified as just Polynomial (P)

means that a deterministic algorithm can solve it in polynomial time. In contrast, a problem classified as NP means that a non-deterministic algorithm can solve it in polynomial time. The more complex the problem is, the less ability the deterministic algorithm has to solve it. Polynomial time is better than non-polynomial time (e.g. exponential time) as this means that the problem can be solved faster. The complexity of problems theory deals with decision problems. Optimization problems which have a binary yes or no answer reduce to decision problems [43].

Various control systems use meta-heuristic algorithms to tune PID controllers. Table 2.1 illustrates this, including which algorithms performed best.

TABLE 2.1: Meta-heuristic algorithms for tuning PID controllers in different applications

Algorithm	Application	Best	Reference
Genetic Algorithm (GA), Particle Swarm Optimization (PSO) Algorithm	Camera gimbal stabilization system	PSO	[25]
Flower Pollination Algorithm (FPA)	Automatic Generation Control (AGC)	N/A	[40]
PSO, Artificial Bee Colony (ABC) algorithm	AGC	ABC	[44]
Differential Evolution (DE) variants	planar robot	N/A	[45]
FPA, GA, PSO	Load Frequency Control (LFC)	FPA	[46]
Bat Algorithm (BA), Hybrid BA, PSO, DE, GA, Cuckoo Search (CS)	Robot arm	PSO	[47]
PSO, GA, Simulated Annealing (SA), PSO-Nelder-Mead (NM)	Speed control of DC Motor	SA-NM	[48]
GA-NM, SA-NM	CNC machines, industrial robot	N/A	[49]
GA	Heating, Ventilation, and Air Conditioning	N/A	[50]
Teaching-Learning Based Optimization (TLBO), Firefly Algorithm (FA)	Automatic Voltage Regulator	TLBO	[51]
Bacteria Foraging Optimization (BFO), PSO			

The purpose of this research was to focus on the gimbal stabilization problem. Other than the study in [25], not many researchers have used meta-heuristic algorithms to tune the PID controller for the gimbal stabilization problem, which feeds into the motivation for this current work.

The Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Ant Colony Optimization (ACO) algorithms are known to be the most popular and well established meta-heuristic algorithms [52]. The PSO and ACO are both swarm intelligence algorithms, whereas the GA is an evolutionary algorithm. Though these three algorithms are most popular, there are many different algorithms and so choosing which is most appropriate for the application is challenging and an ongoing topic in interdisciplinary research [53]. The algorithms used in this study

belonging to different meta-heuristic algorithm groups also feeds into the motivation for this research as the underlying justification for it is to make it easier to know which group of algorithms work best for the gimbal stabilization system.

Observing which algorithm performs best on other applications is one factor used as a method of selecting the algorithms to use in this study, as seen by studying Table 2.1. Algorithms which perform well in other applications are promising and have a higher chance of performing well in this application as well. Nevertheless, this is not guaranteed as algorithms can perform differently for different applications. Another factor to consider is observing how recently developed the algorithms because the assumption is that recently developed algorithms improve the performance of previous algorithms. Since this is a comparative study, it was essential to choose algorithms with different characteristics to be able to observe whether the characteristics of the algorithms affect the performance of the system.

The three algorithms chosen are the Flower Pollination Algorithm (FPA), TLBO and GA. The FPA was chosen since it has performed well in [40] and [46] and it is also recently developed in 2012. The TLBO was selected since it performed well in [51] and is also recently developed in 2016. Another advantage to choosing the TLBO is because the FPA and GA have algorithm-specific parameters, which the TLBO does not, and thus it was possible to observe whether this made a difference in performance. The reason for choosing the GA is because it is one of the popular and established algorithms, and a ‘standard’ one to compare the performance of others. Since the FPA and TLBO are swarm intelligence algorithms, and GA is the evolutionary algorithms. Thus, it would be useful for a comparison of performance between the two algorithm classes. Since the FPA and TLBO are relatively newly developed in comparison to the PSO and ACO, observing the capabilities of these algorithms would prove to be beneficial.

2.3.1 The Genetic Algorithm

The Genetic Algorithm (GA) is one of the oldest meta-heuristic algorithms, along with DE and PSO. John Holland executed the first GA from the 1960s to 1980s [54] [55]. From this, the DE emerged next in 1995, along with the PSO [56] [57].

The book ‘Adaptation in Natural and Artificial Systems’ by John Holland was the first book documenting the GA [58]. This GA was developed to solve problems that brute force algorithms could not solve. Brute force algorithms refer to algorithms that check every possible solution when solving a problem.

The GA initially developed using binary representation [59]. Binary representation means the variables that the GA optimizes are as an encoded binary string. However, this study uses continuous values as variables because the PIDN controller values are continuous themselves. Thus, there was no need to convert to binary representation. Also, if the value of the variable is large, this would require many bits to represent it. Thus the variables are expressed as floating-point numbers. The advantage of using continuous variables as it requires less storage than binary variables.

Since the Darwinian evolution forms the basis of the GA, knowing the three core principles of this evolution becomes imperative in understanding the GA.

The three Darwinian principles include [55]:

1. Heredity: Creatures pass down genetic information to their descendants, provided they live long enough to reproduce.
2. Variation: There must be variation present in a population. If there is no variation, parents will be identical to children, and no new combination of traits will be present. Therefore nothing can evolve.
3. Selection: There must be some form of mechanism that chooses which parents will be able to reproduce. Selection is ‘survival of the fittest’ with fittest not being described as which creature is the strongest or most prominent,

but which creature has adapted well to the environment, thus having a better likelihood of surviving.

Based on the three core principles mentioned above, the GA is then divided into three parts; creating the population, selection and reproduction. These three parts are what forms the basis of the algorithm for optimization.

2.3.2 The Flower Pollination Algorithm

The Flower Pollination Algorithm (FPA), developed by Yang [60], is a swarm-based optimization technique. It has gained popularity in many optimization fields due to its impressive capabilities [61] because the FPA, though has fewer parameters, has shown to work well in different optimization scenarios [61]. The pollination of flowers forms the basis and inspiration of the FPA. There are two main branches for updating the solutions; local search step that mimics local pollination of flowers, and global search step that mimics global pollination of flowers [62]. Global pollination occurs when pollinators, such as bees and bats, travel long distances to pollinate flowers, whereas local pollination occurs when the flower self-pollinates.

2.3.3 The Teaching-Learning-Based Optimization Algorithm

The Teaching-Learning-Based Optimization (TLBO) optimization technique stemmed from all evolutionary and swarm-based algorithms requiring input parameters for operation. Other algorithms required proper tuning of these input parameters in order for optimal performance. Improper tuning of these parameters may result in local convergence rather than global convergence. Therefore, the need for a non-parameter based optimization technique such as the TLBO came about.

The TLBO is an algorithm based on and inspired by the teaching and learning process [63]. It consists of two phases, the teacher and the learner phase. In the teacher phase, the learners gain knowledge from teachers. In the learner phase,

the learners gain knowledge from each other as they interact with each other [63]. The learners are the population members, and the learner who performs the best is the teacher [63]. The design variables of the optimization problem are the population members subjects offered to each learner. The learner's results or 'marks' of these subjects are the 'fitness' value of the optimization problem. Thus these design variables are parameters which are involved in the objective function of the optimization problem. The best solution to the optimization problem is the best value of the objective function [63].

2.4 Parameter Optimization

The mathematical folklore of the No Free Lunch (NFL) theorem(s) for optimization states that the performance of all algorithms is, on average, the same. In other words, if an algorithm does well in one application, it will 'pay' for this by doing poorly in another application, and thus averaging out the performance of all algorithms [64] [65]. This folklore is a result of the work done by Wolpert and Macready in [64] and highlights the importance of exploiting problem-specific algorithms because its performance depends on the application.

The effects of the parameters on the performance of the algorithm, though, remains an ongoing problem. Perhaps the algorithms which perform poorly in some applications have the potential to perform well if the right parameters are chosen for that algorithm. This section discusses the algorithm-specific parameters, and common algorithm parameters, how these parameters are usually determined, and the possible effects these have.

2.4.1 Parameter Tuning and Parameter Control

Choosing the algorithm parameters may be an optimization problem itself, which is why using these algorithms to tune an optimization problem falls under hyper-optimization. Karr and Wilson [66] state that there are no guidelines for choosing these parameters for the different applications.

Parameter tuning refers to choosing the parameters of the algorithm before running the algorithm for optimization, meaning that the parameters of the algorithm are static throughout its run. Parameter control refers to choosing dynamic parameters of the algorithm such that they change as the algorithm runs. According to Eiben et al. [67], there are three categories for parameter control which are: deterministic, adaptive, and self-adaptive. Deterministic parameter control is using a deterministic rule for altering the parameters. This deterministic rule is determined before the algorithm is run, such as, a formula. Therefore, this formula does not use the feedback of the algorithm performance to determine the parameters. Adaptive parameter control is where the feedback of the algorithm performance feeds into choosing the parameters of the algorithm. Self-adaptive parameter control is where the parameters are part of the parameters that will be optimized by the algorithm.

In parameter tuning, when choosing static parameters for the algorithm, the question about what values to use for the parameters becomes important. The static values are either chosen by testing different combinations and observing the performance, or by choosing the same parameters from similar work. The disadvantage of testing different parameters and observing the performance is that trying all different combinations can be time-consuming and sometimes impossible [68].

The optimal parameter settings for the algorithms tend to differ with different problem instances. Also, if the justification for using meta-heuristic algorithms for tuning the PID controller is that the process of tuning this controller can be automated and more accurate, spending time tuning the algorithm itself makes this justification less significant. The problem with choosing the parameters of an algorithm by using studies which have done ‘similar’ work and have proved to be successful [68], is that it is unclear where the similarities should lie in order that the slight differences in the model do not affect the algorithm performance. Karafotias et al. [68] stated that the perceived similarities between problems do not imply that the algorithm parameters should be similar too. Another problem with choosing static algorithm parameters is that different values of the parameters would work better at different stages of the algorithm. For example, in the EA, the

mutation rate should be higher in the beginning stages to allow for diversification and exploration of solutions. As the algorithm progresses, the mutation should decrease to allow for the intensification and fine-tuning of the solutions [68].

Most studies, however, do choose parameters of an algorithm from observing parameter values chosen from the studies that have done similar work and deem this as enough justification. Some just choose static parameter values without proper justification [48] [47] [69] [70]. The main reason for this is because parameter control requires additional computational effort and may not result in significant performance improvement.

Another tuning method would be to use a theoretical approach. Using a theoretical approach for the parameter setting has been tried, specifically for the evolutionary algorithms, with some theoretical investigations on the optimal mutation and crossover probabilities [71] [72] [73]. However, these investigations are typically based on simplified functions for optimization problems, thus would not be of help for real-world problems. The theoretical approach for many meta-heuristic algorithms still needs to be developed, so this may not be an option for every algorithm.

There are studies which have implemented parameter control on their algorithms. For example, in [49], a changing mutation rate was implemented in the GA, for three experimental tests, namely two Computer Numerical Control (CNC) machines, and an industrial robot. The mutation applied a deterministic parameter control technique—this technique used of a formula which adapts the mutation value depending on the current generation cycle. Lin and Lui [74] compared the adaptive GA, classic GA and Ziegler-Nichols method for tuning the PID controller for a plant described by a 3rd order transfer function. In this study, classical GA is where the parameters are static throughout the algorithm run. The adaptive GA is where the crossover and mutation rate were adapted according to a specified formula. The study found that the adaptive GA performed the best.

Ginley et al. [9] chose to use a self-adaptive parameter control technique for tuning the parameters of the algorithm because it does not rely on the time or iteration

as deterministic techniques do but on the performance output of the algorithm. Notably, the technique focuses on maintaining the diversity of the population of the GA. The GA is well known for local or premature convergence [75], and this technique aids in helping the algorithm in this by ensuring that diversity in the population is maintained. Since this method promises to mitigate premature convergence and to ensure diversity, this GA was used in this study as the adaptive GA.

There are many other parametric studies for the GA, and the literature on this topic is vast since this algorithm is the oldest [76]. According to Yang, the essence of this literature states that the crossover rate should be higher (around 0.7 to 0.95), and the mutation rate should be low (0.1 to 0.01) to ensure convergence [76]. The book by Haupt et al. [59] implements the continuous GA by using a crossover rate of 100% (which means all the population members undergo crossover) and uses the mutation rate of 0.2 or 20%. The static GA uses this implementation in this research.

The FPA has a switch probability parameter which controls the diversity, and intensity balance which is usually chosen as 0.8 [60] however Ozsoydan and Baykasoglu [77] highlight the importance of this parameter and how this needs further analysis even though there are not many studies which conduct this. This became a motivation for analysing the effects of various switch probability characteristics in that study. Their analysis included evaluating the effects of changing the switch probability exponentially, linearly, and through sawtooth changing patterns throughout the iterations. The study in [78] chose a deterministic technique to change this parameter as the algorithm runs, which is dependent on the generation cycle.

This current study makes use of two implementations of the FPA: the static FPA with a switch probability of 80%, and an adaptive FPA with a deterministic switch probability adapted from [78] to compare how this affects performance.

Yang et al. [79] presented a self-tuning optimization algorithm framework in order to mitigate the burden of tuning the algorithm itself first before solving the

problem at hand. However, this framework is applied to optimization problems where the optimal solution is known, which is not the case for this study nor most real-world problems.

Some complexities exist when using adaptive GA in comparison with the adaptive FPA. The GA has more than one parameter to tune with a two-level tuning required. In other words, there are two layers related to most of the parameters. The first layer involves choosing the type of parameter and the second choosing the value associated with that type. For example, in choosing the mutation for the GA, there are different mutation types and different mutation rates. It is the same with the crossover and selection parameter. The FPA however, only consists of two parameters, namely the switch probability and a scaling factor. Moreover, these factors do not have different types of parameters but only different values.

The perceived advantage of the TLBO is that it does not have any algorithm-specific parameters. While the FPA and GA have parameters which choose whether the population encounters diversification or intensification, the TLBO does not. The whole population moves from the teacher phase, where diversity and exploration are fostered, to the learner phase where intensification and exploitation occurs, all in one iteration. That is why the TLBO does not have algorithm-specific parameters. However, what all the algorithms do have to choose and tune are the common algorithm parameters.

2.4.2 Common Algorithm Parameters

These are parameters which are not algorithm-specific. Comparative studies mostly use the same common algorithm parameter values as these parameters can be seen as the ‘resources’ which the algorithms need to perform.

The common algorithm parameters considered in this study include the population size, stopping criteria, search space bounds and fitness function.

2.4.2.1 Population Size

All the algorithms used in this study are population-based. These algorithms use population members, which are different solutions to the problem to observe which member performs the best. Thus, the number of solutions, which is the population size, must be chosen. The population size affects the performance of the algorithm because it is the number of solutions that the algorithm will be testing to observe which performed best. The smaller the population size, the less the solutions, and this risks the optimal solution not being part of the population. However, a larger population size increases the computational time required to obtain the optimal solution.

Fister et al. [47] applied meta-heuristic algorithms to the PID controller to control a robot arm. In that study, the parameter values of the algorithms were small because the study tested the reactive nature of the algorithms. Thus, the population size value was 10, and the number of iterations value was 10. The authors of [60] found that when developing the FPA and testing it against other algorithms (namely the GA and PSO), a population size of 25 was sufficient for the test functions. In [48], the study tested two population-based optimization algorithms, namely the GA and the PSO. The population size chosen for each algorithm differed, with GA population size being 20 and PSO population size being 40.

For the evolutionary algorithms, there are some theoretical investigations on the population size [80] [81] [82] [83]. However, these investigations are generally based on simplified functions for optimization problems. Some suggest varying the population size as the generation number progresses [84]. Talbi [43, Chapter 3], states that for evolutionary algorithms, it is customary to choose a population size between 20 and 100. A population of size 25 has been proposed for the FPA to work the best [60]; however, this study applied these algorithms to benchmark functions. The TLBO population size used in [63, Chapter 2] was 5. However, the study applied the TLBO on benchmark problems.

The study by Bujok et al. [85] made use of two benchmark sets to test the performance of eleven nature-inspired and bio-inspired algorithms. These two benchmarks include the CEC 2011 collection of 22 real-world optimization problems and the suite of 30 artificial optimization problems defined for the competition of the algorithms within CEC 2014. Initially, all algorithms used the same population size, with a population size of 90 for the first benchmark sets, and 50 for the second benchmark sets. The authors then decided to compare these population sizes with the recommended population size for each algorithm to observe whether the recommended population size from other studies allows the algorithm to perform better. Two of the eleven algorithms decreased population size linearly as the iteration progresses.

Since this research was a comparative study, the same population size values were used for all three algorithms to observe which algorithms converge faster. The population size chosen was between 20 and 60 based on the literature above. This decision was made because the FPA recommended population size is 25, the TLBO was developed with a population size of 5 and EA algorithms recommendations are between 20 and 100.

2.4.2.2 Stopping Criteria

There are various methods used in heuristics to halt the execution. These methods include using the CPU time, computational count, or solution quality [86]. In the first method mentioned, the algorithm halts after a given time. The second method halts the execution after a given number of computational counts such as the iteration number or number of fitness function evaluations. The last type mentioned observes how close the solution is to optimal. It usually halts execution after achieving the optimal solution or if the solution is within a defined tolerance.

Using computational count is most common partly because of the disadvantages the other methods possess. Using CPU time has been criticized on the grounds of reproducibility [87]. CPU time is affected by other factors such as the machine the algorithms use and the implementation of the code. Thus, the results may not be

able to be reproduced by other practitioners. Using the solution quality is limited to cases where the optimal solution is known. Another method used as a stopping criterion is by observing the population diversity and halting execution when the diversity is below a given threshold [19]. However, this would require one to define that threshold. Because of this, this current study will use computational count.

This study will make use of the number of fitness function evaluations rather than the iteration number for the computational count stopping criteria. This is because different algorithms have different fitness function evaluations in one iteration. Each function evaluation represents an assessment of the solution and an opportunity to improve this solution. For a fair comparison, it is better to observe the number of fitness function evaluations rather than the iteration number, though these are closely related.

The performance of the algorithm is also dependent on the computational effort required to obtain the optimal solution. The fewer the number of iterations or fitness function evaluations needed to converge to an optimal solution, the better the performance of the algorithm because of the lower computational effort required for the algorithm to perform well [88]. An ideal algorithm would only take one iteration to converge to an optimal solution [88]. However, nothing is ever ideal. Ridge [86] emphasizes the challenge in choosing the right computational count since this has a significant impact in finding the right solution in a reasonable time. Socha [89] investigated the influence of different running times on the parameter choices of the max-min ant system and found that there is an effect that the different running times have on the parameters of the system and that the results are dependent on the stopping criteria used.

Some research aims to investigate the impact of the stopping criteria by changing the stopping criteria after each experiment [86] [89]. This current study has made use of fitness function evaluations as a stopping criteria. Hence, keeping a constant number of iterations but changing the population size to investigate the different levels of fitness function evaluations was most appropriate, as the population size and iteration number are what determine the fitness function evaluations.

2.4.2.3 Search Space

Another common algorithm parameter is the search space. Because these algorithms are search techniques, there must be a limitation to exploring a reasonable region of variable space. This region must be diverse enough in the initial population to explore a reasonably sized variable space before the algorithm focuses on a specific region [59]. Some studies actually improve the algorithm performance by optimizing how the algorithm interacts with the search space [90]. This shows the importance of the algorithm's search space and thus, this is the first step to finding the optimal solution.

The distance from one solution to the next is also an important factor when analyzing the search space. This distance can be a measure of the diversification and intensification of a particular solution. Some algorithms, such as the scatter search take into account diversification when generating the initial population or search space. This algorithm does this by ensuring that the minimum distance between the solutions is maximised [43].

According to Talibu [43, Chapter 3], there are four categories of strategies dealing with the initialization of the population and thus the search space. These categories are random generation; sequential diversification; parallel diversification; and heuristic initialization. Random generation is usually used in continuous optimization and was used in this research. Random generation is where the initial population is generated randomly within specified bounds. However, the term 'randomly' means pseudo-randomly because software cannot produce truly random numbers. The problem with using pseudo-random numbers lies in that this implementation can introduce a bias in the algorithm's behaviour since it is not truly random [86]. Ridge suggests that because of this, the computational experiments must be replicated using more than one generator [86]. However, this will not be considered because all the experiments were implemented in one machine.

The search space bounds, for the random generation of population, are dependent on the application and must be chosen such that they encompass the optimal

solution. In [47], the search space bounds for the PID controller gains differed for the two-axis robotic arm, with the upper bounds being 0 to 400 for one axis, and 0 to 20 for the other axis. In [48], where a DC motor used the PID controller for speed control, the search space bounds for the PID controller gain were taken to be 0.0001 to 1.5 for K_p , and 0.0001 to 1.0 for both the K_i and K_d values. These bounds were taken from [91], where an AVR system used a PID controller. The importance of observing which bound suits the algorithm performance lies in that these bounds are what allows for the optimal region of solutions to exist. Thus, this study tests different search space bounds to observe whether this affects performance. A fitness function is required to measure which solution is, in fact, optimal.

2.4.2.4 Fitness Function

The fitness function is imperative for the optimization of the algorithm and its satisfactory performance on the specified application. Some differentiate between an objective function and a fitness function though it is essentially the same thing. The objective function contains different variables that are the performance measures of the system. The fitness function then becomes a question of whether or not this objective function will be minimised or maximised. This current study uses these terms interchangeably.

Sometimes the fitness function can be based on one performance criterion, such as checking whether the response time of the system is short. However, there are instances where there are multiple performance criteria which need to be satisfied to optimize the specified system. The purpose of a merit function is to combine multiple objectives into a single objective by using a weighted sum [92]. A merit function can be written in the following manner [92]:

$$F = \sum_{i=1}^m w_i f_i,$$

with m being the number of objectives, w being the non-negative weights, and f being the objectives. Choosing which performance criteria to use and what the weighting values should be, is at the discretion of the engineer and depends on the application.

Some studies, such as those in [47] and [48] simply choose the fitness function without any justification. Furthermore, many solely observe the time-domain response of the control system as the variables to the fitness functions. The fitness function chosen to be maximised in [47] is:

$$F = \frac{1}{2}(w_1(1 - |P - M_P|) + w_2(1 - t_S) + w_3(1 - e_{ss})),$$

where w_1 , w_2 and w_3 represent the weights of the function, P represents the desired overshoot value, M_P represents the actual overshoot, t_S represents the actual settling time and e_{ss} represents the actual steady-state error. This is different from the fitness function in [48] shown to be:

$$F = (1 - e^{-w})(M_P + e_{ss}) + e^{-w}(t_S - t_R),$$

where w is the weight value, e_{ss} is the steady-state error, M_P is the overshoot value, t_S is the settling time, and t_R is the rise time of the response of the system. This fitness function, adapted from [93] where the PSO algorithm, was used to tune the PID controller for an Automatic Voltage Regulator (AVR) system. What is interesting in [48] is that the weighting factor w was tested for three different values to see which weighting gives a better response.

Unlike the research mentioned previously, the study in [40] chose not to use the time response of the control system but rather compared the error response of the system.

The study in [40] compared four different performance indexes, namely, the Integral Square Error (ISE), the Integral Time Square Error (ITSE), the Integral Absolute Error (IAE) and lastly, the Integral Time Absolute Error (ITAE). From here, the study found that the ISE and ITSE work best and thus, the research

uses the ISE as the fitness function. These four cost functions were also compared in [69] where the meta-heuristic algorithms tuned the for an Automatic Generation Control (AGC) application. This study also found that the ISE performed best. The study in [49] combined tuning the PID controller using the GA with the Gain-Phase Margin method. This novel new combination was compared with the individual methods (i.e. using GA and Gain-Phase Margin individually) to see which performs best in the application of a CNC machines and an industrial robot. The fitness function used in this study is:

$$F = w_1 V_{\text{ISE}} + w_2 M_P + w_3 V_{\text{MSE}},$$

where V_{ISE} and V_{MSE} are the values of the ISE and the Mean Square Error (MSE) respectively, and w_i are the weight values.

The research in [51] shows the combination of the time-domain and error response variables for the fitness or objective function, where:

$$F = w_1 M_p + w_2 t_s + w_3 V_{\text{ITAE}} + w_4 V_{\text{ITSE}},$$

with w_i being the weights chosen, t_s being the settling time, and V_{ITAE} , and V_{ITSE} being the ITAE and ITSE values respectively. The combination is also seen in the equation:

$$F = w_1 M_p + w_2 t_s + w_3 V_{\text{MSE}}^2,$$

where w_i are the weights and V_{MSE} is the mean square error value, where this equation was used as the objective function in [50] for the tuning of a PID controller using GA, for a Heat Ventilating and Air Conditioning (HVAC) system.

An objective function does not usually combine the performance indices with the time domain step responses because the performance indices affect the time domain response. Thus adding all these variables into one objective function may be redundant, and the weights would hold no significant value. The study performed in [94] indicates that for tuning the FOPID controller, the ITAE reduces oscillations but with the expense of producing high initial system error and a longer

rise time. The ISE and ITSE both improve the rise time of the system, however with both error indices resulting in high system oscillations. The shortest rise time with the largest overshoot was ISE, followed by ITSE.

The designed merit function included weighted averages on both the ISE and the ITAE to compensate for the advantages and disadvantages of the performance indices. The ISE will help decrease the large initial system error and aid in the fast response of the system. The ITAE will reduce oscillations caused by the ISE while mitigating the steady-state error of the system. The merit function chosen is shown in Equation (2.1):

$$F = \tau(V_{\text{ISE}}) + \beta(V_{\text{ITAE}}) \quad (2.1)$$

where τ is the weight value of the ISE chosen to be 0.6; V_{ISE} is the value of the ISE; β is the weight value of the ITAE chosen to be 0.4 and V_{ITAE} is the value of the ITAE. The weights were chosen based on the application at hand. The gimbal stabilization system needs to respond quickly to commands. However, it is also vital that it does not produce large overshoots as it may be supporting optical equipment, hence the weight of the ITAE error-index is not less than 0.4. Image processing can be used to mitigate small errors caused by oscillations, for example. However, tracking the target on time is essential. Equation (2.1) was, therefore used as the fitness function in this current research.

2.5 Performance Measures for Evaluating Meta-Heuristic Algorithms in Gimbal Stabilization

McGeoch correctly states that choosing the performance measures depends on the questions that motivate the research [95]. Since this study was applying meta-heuristic algorithms on a control system, it investigated two categories of performance measures. The one category dealt with the performance of the algorithms themselves. The other category dealt with the performance of the control system. This section aims to tackle these two categories briefly.

2.5.1 Performance Measures of Meta-Heuristic Algorithms

The primary performance measure used for meta-heuristic algorithms is the observation of the fitness value for each solution. The aim of the solutions would either be to decrease or increase this fitness value, and thus the solution with the lowest or highest value is one which has the highest quality. If the optimal solution is known, then presenting solutions as a percentage over this best solution can be done [86].

The probability of the algorithm obtaining the optimal solution for a given instance can also be of interest. However, there are limitations associated with this performance measure. One of which is that this performance measure is useful when the optimal solution is known [87]. This measure also ignores the solutions which are close to the optimal since the focus is on the optimal. Thus, the measure emphasises on finding the optimal rather than finding a good enough solution in a reasonable amount of time, which is the purpose of heuristic algorithms [86].

Since these are stochastic algorithms, each run may not give the same solution. Hence the need for running the algorithms multiple times to observe the average response. The question is whether the overall final solution of the algorithm is the best or the average solution. Birattari and Dorigo [96] criticise that choosing the best solution from a number of runs as the solution of the algorithm is being over-optimistic for the algorithm performance since it is less likely that the algorithm will reproduce this solution and more likely that it will reproduce a solution closer to the average solution. However, Eiben and Jelasity [97] note that in a real-life scenario, the best solution from the number of runs will be chosen rather than the average solution. This research recorded both the best and average solution (i.e. those that give the lowest and average fitness function value). However, for comparison purposes, the average solution will be used as this is purely simulation work and not real-life. With simulation work, the purpose is to observe which solution the algorithm is most likely to be reproducible.

2.5.2 Performance Measures of Tuning PID Controllers Gimbal Stabilization Control Systems

One of the most important ways to measure the performance of the gimbal control system is to observe how accurately it follows the commanded input. There are many different signals to use as the commanded input, but a common one is a step input signal, which can either represent the LOS rate or position. Abdo et al. use this as one method of observing the performance of the control system for the two-axis gimbal system when tuning the PID controller using fuzzy control [14]. Rajesh and Ananda do the same in the control of the camera position in an Unmanned Aerial Vehicle (UAV) using a two-axis gimbal system [98]. This is also evident in Rajesh's work, where the camera gimbal stabilization system used the PID controller [25]. Other commonly used signals for control systems, in general, are ramp functions, sinusoidal functions, impulse functions, and white noise [21].

Using a step response allows for the measure of the control signal to quantify how fast it responds to the command signal. The time response characterises the transient response includes observing the rise time, settling time, overshoot percentage, amongst other measures, except for the steady-state error. For control systems, however, one can also observe the frequency response of the system.

The frequency response of a system is described as the steady-state response with a sinusoidal input signal [10]. The frequency response performance requirements are based on a few classical control theories, two of which are the Nyquist criterion and Bode. The main purpose of these requirements is to quantify the robustness of the system [99]. Singh et al. made use of the bode plot to observe how the optimized PID controller performs with the gimbal system [100]. However, for this research, the main focus was observing the transient response of the system, particularly on the step and ramp input functions. Future work can consider the frequency response and other input signals.

Note that since the control system for this research was on controlling the LOS rate, the step input function represents a constant LOS rate or velocity, which means the

target is moving at a constant rate or velocity. The ramp input function, therefore, represents an increasing LOS rate or velocity, but with a constant acceleration. These are the two scenarios that the two signals used in this research represent.

2.6 Theoretical Analysis of Meta-Heuristic Algorithm Performance

Meta-heuristic algorithms are applied in various systems and problems, however, the theoretical and mathematical analysis of these algorithms still has many open questions [17]. Theoretical analysis of meta-heuristic algorithms is challenging because of the highly non-linear, complex and stochastic interactions between the factors of these algorithms [53]. Krüger [53] states that even when a solution is known to be optimal, it is difficult to prove that theoretically. Even when a sub-optimal solution is known, estimating how far this solution is from the optimal solution is challenging. Giving an analytical prediction of whether a solution is achievable with the given computational resources then becomes impossible [101]. Questions such as what the optimal balance between exploration and exploitation is, or how algorithm parameters affect performance efficiency cannot be theoretically answered [17]. This chapter will focus on three ways which the performance of meta-heuristic algorithms can be theoretically analysed. These three ways are convergence analysis, statistical analysis, and complexity.

2.6.1 Convergence Analysis

In stochastic algorithms, convergence is the ability of the algorithm to transform from one solution to a new random solution which provides the optimal performance [102]. If the considered problem has a solution, the convergence analysis answers the question: Will the algorithm be able to find the desired optimal solution, given sufficient time and resources? In other words, is the current solution able to converge to the optimal solution? if so, how fast does this happen? [53].

Since meta-heuristic algorithms are inherently random, there is no guarantee that the algorithm will find the optimal solution. This is why convergence analysis makes use of probability and the probability theory. There are two types of stochastic convergence defined in convergence probability: convergence in probability and convergence in probability 1 [53].

Definition 1 Convergence in probability [53]: A sequence $\delta_n, n = 1, 2, \dots$ of random variables, with δ being the random variable, converges with a probability towards δ if for all $\varepsilon > 0$

$$\lim_{n \rightarrow \infty} P(|\delta_n - \delta| \geq \varepsilon) = 0 \quad (2.2)$$

Definition 2 Almost sure convergence (or convergence in probability 1) [53]:

A sequence $\delta_n, n = 1, 2, \dots$ of random variables converges almost surely with or almost everywhere towards δ ,

$$\lim_{n \rightarrow \infty} P(\delta_n = \delta) = 1 \quad (2.3)$$

Equation (2.2) is derived from the weak law of large numbers, also known as Bernoulli's theorem of large numbers in probability theory. Equation (2.3) is derived from the strong law of large numbers [103]. Chen et al. [103] analyzes the general convergence properties of the GA, ACO and PSO algorithms by firstly building a computational intelligence approach (CIA) model of the algorithms based from observing their common features in methodology.

After building the CIA model, the study in [103] then made use of convergence in probability and convergence in probability 1 with the aid of the Markov chain and Martingale theory. The problem with using convergence in probability is that it is only applicable to problems where the optimal solution δ is known and thus, ε can be calculated. However, in real-life problems, this is often not the case. Much like this study, in real-life applications, there is no known optimal solution, thus calculating how far the distance from the current solution to the known optimal solution is not feasible.

The study in [53] discusses theoretical methods to prove convergence for meta-heuristic algorithms. One of these methods is the best-so-far convergence where this type of convergence questions whether the best-so-far solution x_t^{bsf} converges ("w. pr. 1" or "in probability") to an optimal solution x^* , as $t \rightarrow \infty$. The study in [104] mentions that best-so-far convergence is easy to prove but not useful practically. The study in [102] states that this type of convergence is too "generous" because even an inefficient algorithm such as the random search, can be proved to converge to a global optimum.

Another method for convergence introduced by Krüger is model convergence [53]. Model convergence is analyzing the properties of the algorithm that guide the search process towards the region, which contains the optimal solution. Unlike best-so-far convergence, which assesses the probability of one solution (x_t^{bsf}) converging to the optimal solution, model convergence assesses the probability that the algorithm eventually generates only the optimal solution(s). This convergence is harder to prove because a balance between exploration and exploitation of the search is required. That is achieved by fine-tuning the parameters of the algorithm [102]. Davidović and Krüger propose conditions that swarm intelligence methods must incorporate to assure model convergence [102]: (i) all the feasible solutions must be able to be obtained, which represents the best-so-far convergence, and (ii) when the optimal solution is known, its generation must be favoured, which represents the model convergence. There are other methods which are used to prove convergence though.

The study in [62] proved global convergence for the FPA by using the Markov chain theory. A further step was conducted on the GA in [105] by producing bounds for the static and adaptive GA convergence, after modelling the GA's as stationary and non-stationary Markov chains, respectively. In analyzing the canonical (binary) GA in [106], it was proved, by making use of the finite Markov chain analysis, that the only way the canonical GA will converge is by maintaining the best solution in the population. The motivation for the study conducted by Mahmoodabadi and Ostadzadeh in [107] was to improve the TLBO algorithm by increasing the convergence speed for better results in a shorter time. This

was done by introducing a social learning factor into the teaching phase, inspired by the PSO algorithm. This new TLBO was tested on mathematical benchmark functions, and a real-life design problem and was found to be a promising improvement. The improved convergence rate was measured by comparing the run-length distributions of the fitness values for the original TLBO, and the TLBO with a social factor. The study in [108] describes the spatial convergence of the population of the TLBO. This was possible after conducting a geometric interpretation of the TLBO.

All these studies mentioned making use of mathematical theory to prove convergence. However, as stated in [102], the practical usability of these mathematical proofs is still an open question. Davidović and Krüger admit that an infinite number of iterations cannot be performed; thus, limits cannot be assessed, so a method of evaluating the convergence speed is needed. The need for evaluating the convergence speed is also seen in the work by He et al. [62] where the FPA convergence was proved using the Markov chain theory. Then this algorithm was tested practically for the convergence speed. The convergence speed was tested using the run-length distribution graphs where fitness function value against the iteration number is plotted.

The run-length distributions are plotted to observe which algorithm gives the fastest convergence speed, given the same resources in order to compare the convergence speed of the three chosen algorithms of this research. These plots give a practical result of how the algorithms perform in real-life scenarios.

2.6.2 Statistical Analysis

The field of statistics forms a foundation in the use and analysis of meta-heuristic algorithms. The advantage of statistics is that it creates a systematic framework in which algorithms can be experimented with and analysed [101]. The importance of using statistical methods in this study was so that the results and conclusions are

objective rather than judgemental [86]. Analysis of Variance (ANOVA) is a statistical term used to define statistical experiments conducted in order to determine whether results are significantly different from each other. In inferential statistics, there are two main problem analyses: single-problem and multiple-problem analysis. Single problem analysis deals with results obtained over several runs of the algorithms over a given problem, whereas multiple problem analysis deals with a result per algorithm and problem pair [109].

In inferential statistics, two different hypotheses are used: the null hypothesis (H_0) which states that there is no difference between two items being tested, i.e. $H_0: \mu_1 = \mu_2 = \dots \mu_k$, where μ is the mean or median of the results, and k is the populations observed. The alternative hypothesis (H_1) states that there is a significant difference between the populations being tested, i.e. H_1 : not all means/medians $\mu_1, \mu_2, \dots \mu_k$ are equal [110]. The deciding factor which determines which hypothesis is accepted or rejected is the level of significance α . The p-value is a method of stipulating α by providing information about whether a statistical test is significant or not, and how significant the result is [109].

In the statistical procedures developed, there are two classes developed; parametric and non-parametric [109]. Parametric tests were popular in the analysis of performance in Computational Intelligence (CI). However, these tests rely on assumptions which are most likely violated. These assumptions include independence, normality, and homoscedasticity. This study made use of non-parametric tests to quantify whether there are differences between the algorithm performance as no distribution is assumed. There are two categories which exist under non-parametric tests, pairwise and multiple comparisons. Pairwise statistical methods compare only two different algorithms, whereas multiple comparisons compare more than this [109]. This study made use of multiple comparisons since it is comparing multiple algorithms with each other.

The tutorial for using non-parametric statistical tests as a method for comparing Swarm Intelligence (SI) algorithms and Evolutionary Algorithms (EA) states that using Friedman tests is useful for multiple comparisons [109]. If the null hypothesis

is rejected (i.e. there is a difference in the algorithm performances), then the experiment must proceed with a posthoc test to find the pairwise comparisons that produce these differences.

There are many different tests and posthoc tests to use. The study in [47], where a PID controller was tuned using meta-heuristic algorithms for controlling a robot arm, makes use of the Friedman test, followed by the Wilcoxon two-paired test applied as a posthoc test, because the Nemenyi procedure does not usually reveal any difference in most experiments [111]. The study in [112] also made use of the Friedman test along with the Bonferroni-Dunn test as a posthoc test to quantify how well the new meta-heuristic algorithm is in comparison to the well-known algorithms, for the tuning of a PI controller in a wind turbine system. Mohammed et al. [113] also made use of the Friedman and Wilcoxon signed-rank tests for determining which algorithm performs best in the tuning of the PID controller for benchmark functions.

The Friedman tests, developed by the United States economist Milton Friedman [114] [115], was used in this current study to quantify the difference in the performance of the algorithms.

2.6.3 Time Complexity of Algorithms

Comparing the running time of the algorithms is imperative to choosing which algorithm suits the application best. Some algorithms may do well with regards to the quality of results, but take time to acquire such a result. Taking time to acquire a result may not be practical when a fast response in the algorithms is essential. The importance of knowing the time complexity matters not only on evaluating how suitable the algorithm is for the application but also in the inherent evaluation of the algorithm itself.

There are tools available which can determine the running time of an algorithm. However, this running time is affected by other factors besides the algorithm itself, which can make for an unfair comparison. These factors include the processor and

hardware used for the algorithm or the programming language and coding style used for implementation [116]. This means that there needs to be an alternative to evaluating and comparing the run time, which disregards these factors.

There are two crucial resources the algorithm requires to solve a problem: time and space [43]. The time complexity of an algorithm includes determining what the run-time and behaviour of an algorithm are when the input increases or decreases in size [116]. The time complexity links the size of the input to the fundamental steps of the algorithm to do this [117]. This is different from space complexity, where the space complexity relates the input size to the fundamental storage locations [117].

The time complexity analysis utilizes three fundamental notations, including the Big-O (O), Theta (Θ), Omega (Ω), however one which is most popular is the Big-O notation because it describes the upper bound of the complexity. In contrast, the Ω and Θ notation describe the lower and tight bound, respectively [43] [116]. The Big-O notation is also easier to determine and is of practical use because it describes the ‘worst-case’ behaviour of the algorithm [43]. Paul Bachmann, Edmund Landau and others invent this notation. It is also referred to as the Bachmann–Landau notation, the asymptotic notation or popularly known as the Big-O notation [118].

The asymptotic behaviour of each algorithm had to be analysed to determine the time complexity of the three algorithms. This meant examining the fundamental steps of each algorithm and determining the complexity.

Previous work has determined the time-complexity of well-known algorithms such as the ACO, and the GA for the all-pairs shortest path problem, and a ‘sophisticated parent selection mechanism’ to be $O(n^3 \log n)$ [119], where n is the input. The discrete FPA algorithm time complexity was also determined in [120] for the TSP problem and also found to be $O(n^3 \log n)$. However, these algorithms, along with the FPA, have not been determined for this type of problem. This was required for the comparison nature of these algorithms.

2.7 Computational Experimentation

It is important to pay attention to the design of computational experiments when researching heuristic and meta-heuristic algorithms. The main purpose for heuristic and meta-heuristic algorithms is to solve difficult optimization problems with satisfactory quality results in a reasonable amount of time [86] and the computational experiments must be conducted such that this is possible. Johnson [87] lists some questions which must be asked when conducting a heuristics research question that deal with factors such as the adequate number of tests and runs, the type of questions the experiment must address, and who will it benefit given the current state of literature.

From these questions, the computational experiment must be conducted appropriately. Barr et al. [121] characterise two different types of experiments with algorithms; (1) comparing the performance of different algorithms for the same class of problems, and (2) observing the performance of one algorithm in isolation. However, Ridge [86] observed that there are various types of experiments that are identified in the literature:

- **Dependency study** [95]. This type of experiment how certain parameters of algorithms affect a specific performance measure. For example, how the switch probability p affects convergence rate in FPA.
- **Robustness study** [95]. This observes the distributions of results and exams how much deviation there is from the average.
- **Probing study**[95]. This is where the internal features of the algorithm are investigated to understand its strengths, weaknesses and inner workings.
- **Horse race study** [87]. This is when different algorithms are put in competition with each other to determine the superiority of one algorithm over another.

- **Application study** [87]. An application study is where a particular algorithm or code is implemented on a particular problem, and thus, the performance of the algorithm is dependent on the application. This experimental type is usually implemented in conjunction with the other experimental types in that all the others are performed using a particular code with a particular application.

This dissertation makes use of a combination of these types. The primary focus is on dependency study because it investigates the parameters of the algorithms to observe their effect on the performance. Robustness study is then conducted to observe how far the results deviate from average, and further to observe if the optimal parameters are robust to handle changes in problem instances. This study is also partly a horse race study because it consists of implementing different algorithms to observe which one results in a better performance.

For the experimental types identified for this study, there are relevant research questions which emerge. For the dependency study:

- What are the effects of changing the parameter value and type [122]?
- How does the problem size and set affect the performance [122]?
- Is there an interaction between the factors and how does it affect the performance [122]?

These are questions which the computational design attempted at answering. For the robustness study:

- Does the new algorithm perform consistently even with the changes in new class instances? [87]
- How far is the best solution from those which are more easily found? [121]

For the horse race:

- Is there a best overall method for solving the problem? [122]
- What is the quality of the best solution found? [122]
- How does the algorithm running time compare with those of its competitors?
[87]

Because of the three studies (dependency, robustness, horse race) that are relevant in this work, the experiment was divided into three parts; as seen in Figure 2.1.

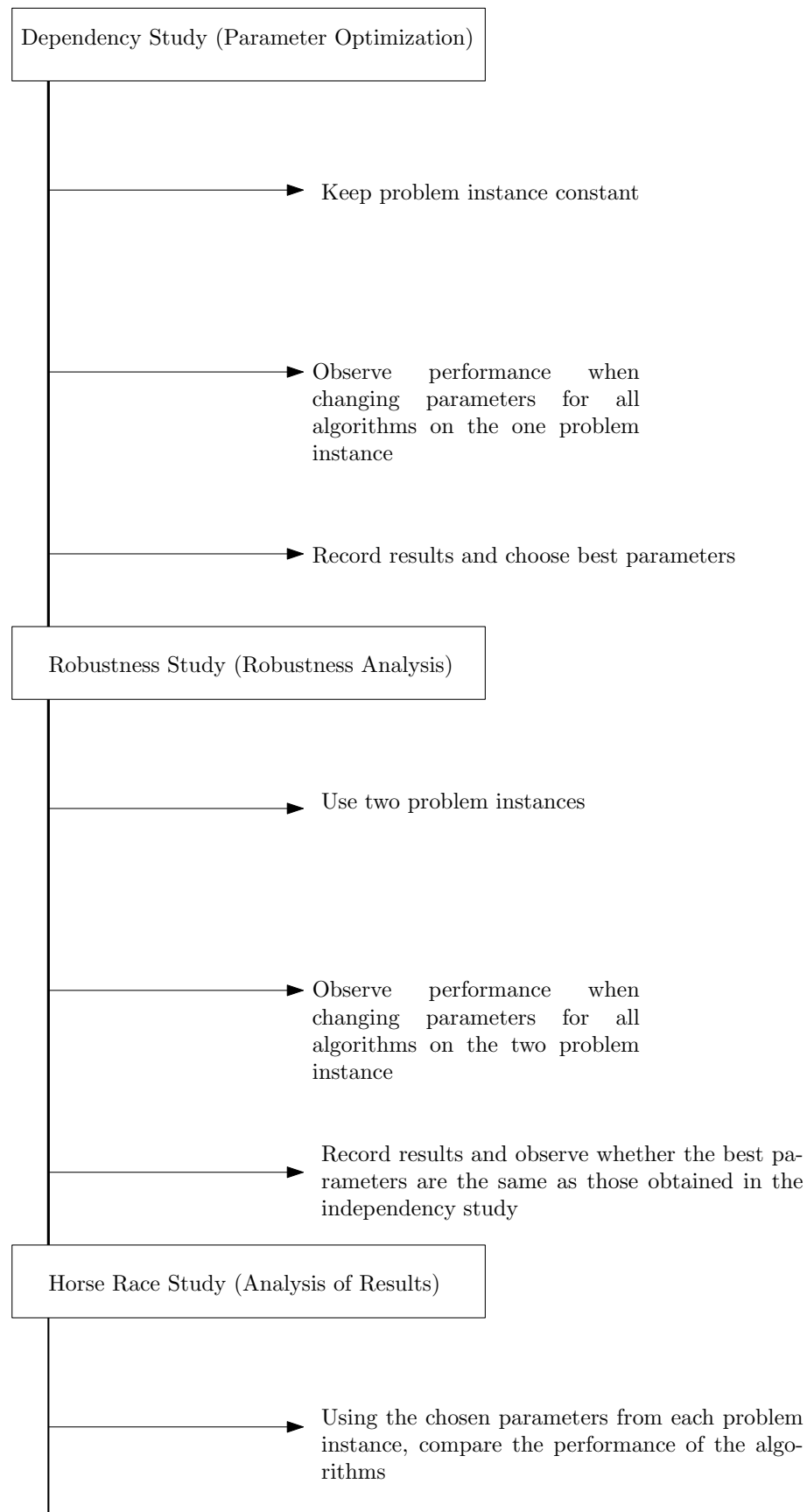


FIGURE 2.1: Diagram showing how the experiments were conducted

2.8 Concluding Remarks

The literature review has various factors which all contributed to this study. This review started with observing the history and elements of the gimbal stabilization, including the PID controller. It then continued by describing the classic and optimization techniques used for tuning the PID controller, which allowed for the introduction of the meta-heuristic algorithms. How to measure the performance for both the algorithms and control system was essential to observe. That includes discussing the theoretical considerations of this work. The review ended by reviewing how to conduct the computational experiments for this type of work.

The next chapter will describe the background and preliminary theory required in order for conducting the experiments.

CHAPTER 3

Background and Preliminaries

This chapter serves as a detailed description of the different models and algorithms used and the underlying theory applicable to the models and algorithms. The chapter begins by describing the gimbal system model and elements and the different algorithms involved. It then continues to describe how the performance was measured. The discussion of the theoretical analysis used for this study follows, and the chapter ends by illustrating the methodology used to obtain the results.

3.1 Plant Model Details

The model of the rate control gimbal system used in this study is shown in Figure 3.1 and adapted from [5]. The primary purpose of the study conducted in [5] was to compare the quality of different rate gyro sensors to determine how this affects the performance of the system. The study mainly focused on testing four different rate gyro sensors on an ideal gimbal system which included a DC motor, a PID controller and a gimbal.

The study conducted by Jia et al. [5] also tested the same rate gyro sensors on the system with added environmental effects which were friction, structural resonance and dynamic system input. This research, however, observed the environmental effects due to friction and added base motion instead of structural resonance, as this effect seems to be commonly studied. The dynamic input observed in [5] included tracking a rate command, an angle command and a sinusoidal command.

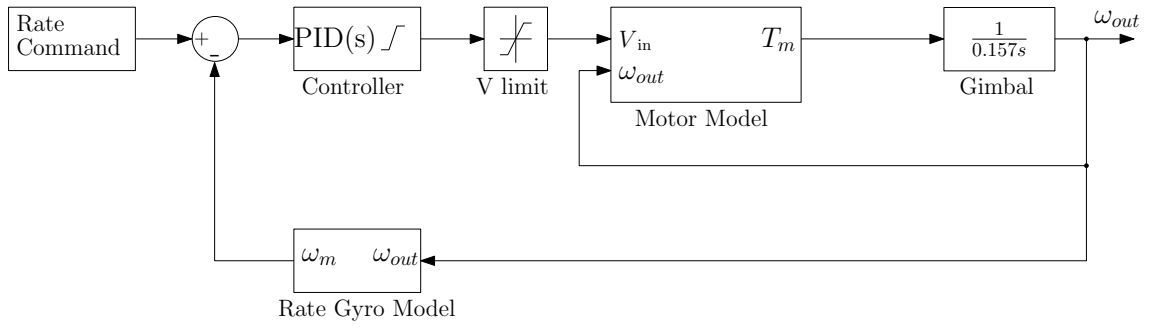


FIGURE 3.1: Block diagram showing the model of the simple gimbal control system adapted from [5]. The Figure shows V_{in} , which is the command voltage taken from the controller, T_m is the output motor torque, ω_{out} is the angular velocity of the gimbal, and ω_m is the measured angular velocity.

This research, though, made use of a step input rate command, and a ramp input rate command.

The model of this study was chosen because it represents a simple generic system which aids in explaining the algorithm performance on this type of plant, rather than on a rare and specific plant. Of course, the results of this study will only be valid for this plant. However, the hope is that choosing a plant with typical elements allows for others with similar control systems to be able to gain from this study. This model was also chosen because the results of the rate command control loop were able to be repeated; thus, achieving model validation. This section aims to discuss the model details.

3.1.1 The Rate Gyro Sensor

The term gyroscope (or gyro), coined by French physicist Leon Foucault, is a joining of two Greek words: *gyros* meaning ‘rotation’ and *skopeen* meaning ‘to see’ [123]. The advantage of the gyro, compared to other sensors, lies in its ability to measure the absolute motion of an object without needing external infrastructure or reference signal [123]. There are two groups which categorize gyros; rate or angle gyros [124]. Rate gyros are different from angle gyros because they measure the angular rates in the plane of interest [14], rather than the angular position. In

a control system, the rate gyro is typically mounted on the gimbal to measure the inertial rotation about the axes that require stabilization and control [11].

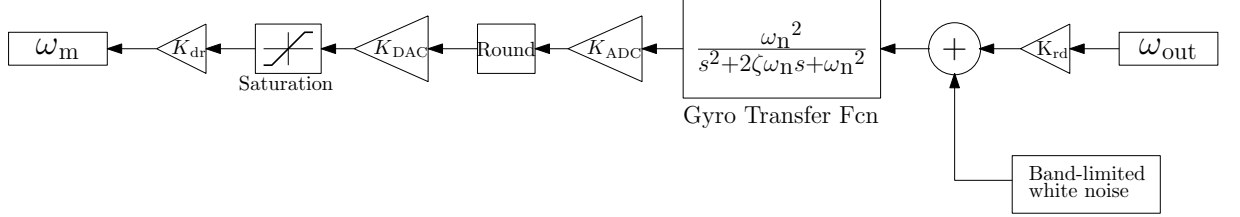


FIGURE 3.2: Block diagram showing the rate gyro model adapted from [5]

Figure 3.2 shows the rate gyro model used for this study. The rate gyro is an essential element in the control system, as its performance can highly affect the performance of the system. Though there are different types of rate gyros, the modelling of the performance of those is quite similar [5]. One of the main contributing factors to the performance of the system is the rate gyro frequency response or bandwidth. This frequency response is modelled as a second-order system with a transfer function shown as:

$$G_{\text{Gyro}}(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad (3.1)$$

where the natural frequency, $\omega_n = 2\pi f_{\text{BW}}$ with f_{BW} being the frequency response or bandwidth and $\zeta = 0.707$ is the damping ratio.

TABLE 3.1: Gyro information

Gyro Specifications	
Bandwidth	250 Hz
ARW	0.0009 deg/ $\sqrt{\text{hr}}$
Range	1000 deg/s
Sampling rate	500 Hz
Analog to digital conversion specifications	
ADC	Signed 32 bit
Maximum rate	± 1000 deg/s
sensitivity	525×10^{-9} deg/s

Since the study's main contribution in [5] was to evaluate how different quality rate gyros affect the performance of the system, the gyro model added non-linearities.

They thus were added in this study as well. One of these non-linearities is the internal noise of the gyro itself. The gyroscope review conducted in [125] states that the main concern in the rate gyro sensors is, in fact, the errors in the measurement of the angular velocity. One cause of these errors is noise. Noise can be expressed differently with different types of gyro's, either through the resolution R or by the Angle Random Walk (ARW) [125]. The study in [5] does not state the type of gyro used, though they state it is a commercially used gyro. However, in [125], it states that usually, the Ring Laser Gyroscopes (RGL) are those that express noise using ARW, and the study in [5] used these. When a rate gyro is measuring the movement of a stationary system, the output can produce an error measurement in the noise. A white noise block in Simulink simulates this noise. This noise block requires a Power Spectral Density (PSD) value in $(\text{deg/s})^2/\text{Hz}$. Equation 3.2, which makes use of the ARW value, calculates the PSD value.

$$PSD[\frac{(\text{deg/s})^2}{\text{Hz}}] = (\frac{1}{3600})(ARW[\frac{\text{deg}}{\sqrt{\text{hr}}})^2 \quad (3.2)$$

Because the gyro measurements are typically used in digital platforms, like microprocessors, an analogue to digital conversion (ADC) needs to be performed for the model to reflect reality, the modelling of the gyro includes this process. Table 3.1 includes the gyro specifications and ADC information.

3.1.2 The Gimbal

The gimbal system, adapted from [6], was re-modelled for this study and is shown in Figure 3.3.

Axsys Technologies developed this gimbal as part of the Cineflex range and is primarily used to house cameras for helicopters [126]. The moment of inertia is determined using Equation (3.3) with assuming a spherical shape for the mass, and observing Table 3.2 for the given mass and dimensions, and assuming a solid sphere since the gimbal system is carrying an optical unit, with the centre of mass assumed at the centre of the sphere.

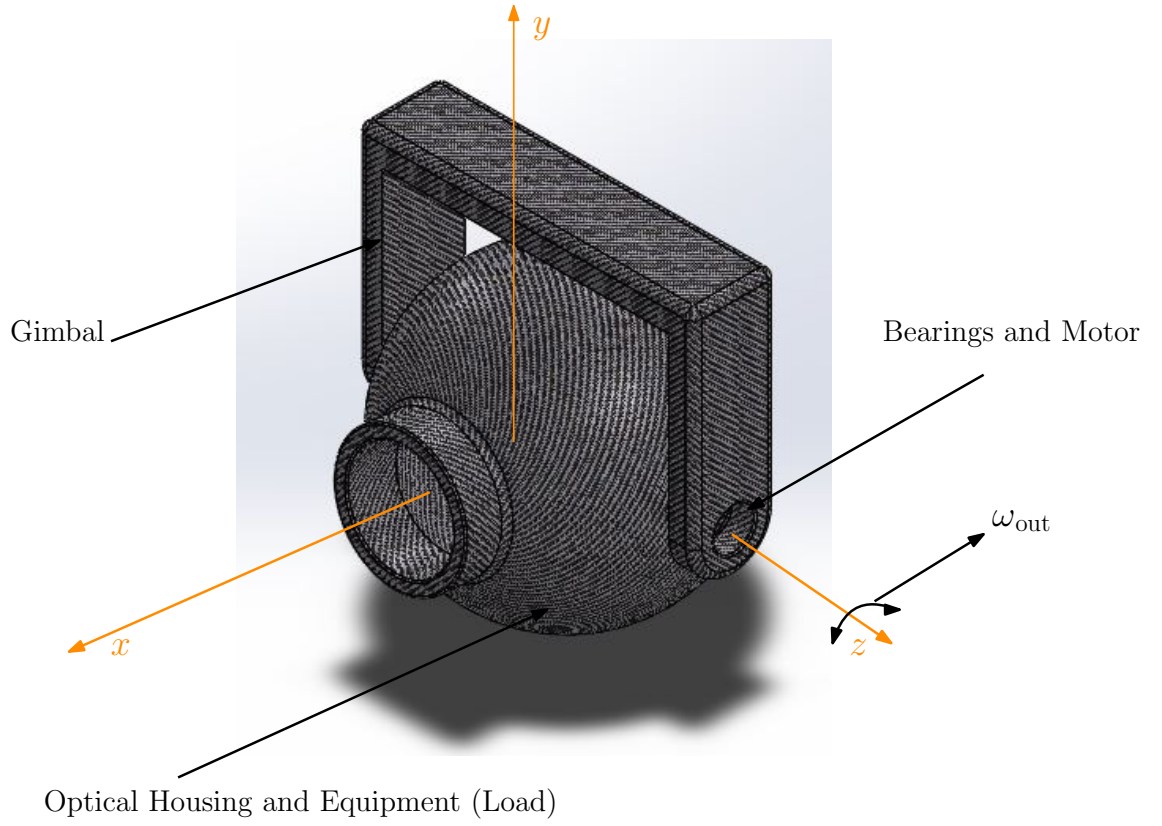


FIGURE 3.3: CAD adaptation of the camera gimbal that was used in this study taken from [6]

$$J_g = \frac{2}{5}mr^2 = 0.155 \text{ kg m}^2 \quad (3.3)$$

With the assumption of an ideal rotational system with no friction, and using Newton's 2nd law, the motor torque T_m is

$$T_m = J_g^* \alpha_g, \quad (3.4)$$

where J_g^* is the sum of the gimbal and motor inertia J_m because the motor torque is exerted on this as well, and α_g is the angular acceleration of the gimbal. Therefore, $J_g^* = J_g + J_m = 0.157 \text{ kg m}^2$. Using the Laplace transform, the transfer function from the motor torque, $T(s)$, to the angular rate, $\omega(s)$, is shown in Equation (3.5).

$$\frac{T(s)}{\omega(s)} = \frac{1}{Js} = \frac{1}{0.157s} \quad (3.5)$$

TABLE 3.2: Gimbal performance specifications

Spec Type	Spec Value
Size (diameter)	35.6 cm
Weight	12.3 kg
Slew rate	55 deg/s
Max acceleration slew rate	100 deg/s ²

The required torque for this gimbal is determined by observing the maximum acceleration slew rate limitation presented in Table 3.2 as 100 deg/s². This maximum acceleration slew rate corresponds to a gimbal slew rate of 55 deg/s. Thus, for a rise time measured from 10-90% to the final commanded input, this gives the range of $55 \times 80\% = 0.44$ deg/s. In order to comply with the maximum acceleration slew rate, this means the gimbal must have a maximum rise time of 0.44 seconds.

To accelerate the gimbal to 100 deg/s², this requires a motor torque of $T_m = J_g \alpha = 0.157 \times (100 \times \frac{\pi}{180}) = 0.274$ N m, J_g is the gimbal inertia, and α is the acceleration. Thus, the power required for the motor is $P = T_m \omega = 0.274 \times 44 \times \frac{\pi}{180} = 0.21042$ W, where ω is the velocity of the gimbal. This calculation was used to determine which motor to use.

3.1.3 The DC Motor

Many actuators are used for this type of application. The advantage of a DC motor comes in its simplicity, ease of implementation and convenience [29]. Differential equations usually represent the dynamics of the motor. These equations have parameters and variables shown in Table 3.3.

These equations are listed below:

$$V_{in}(t) = Ri(t) + L \frac{di}{dt} + e(t), \quad (3.6)$$

$$e(t) = K_e \omega(t), \quad (3.7)$$

TABLE 3.3: Motor specifications

Description	Symbol	Value
Voltage limit	V_{\max}	27 V
Resistance	R	4.5 Ω
Inductance	L	0.003 H
Motor inertia	J_m	0.0017 kg m ²
Motor mechanical constant	K_t	0.85 Nm/A
Motor electrical constant	K_e	0.85 V/(rad/s)

$$T_m(t) = K_t i(t), \quad (3.8)$$

Table 3.3 shows the direct-drive DC Motor specifications derived from [5].

From the equations (3.6) (3.7) (3.8), Figure 3.4 illustrates how the DC motor is applied in the control system, with the description of the symbols shown in Table 3.3.

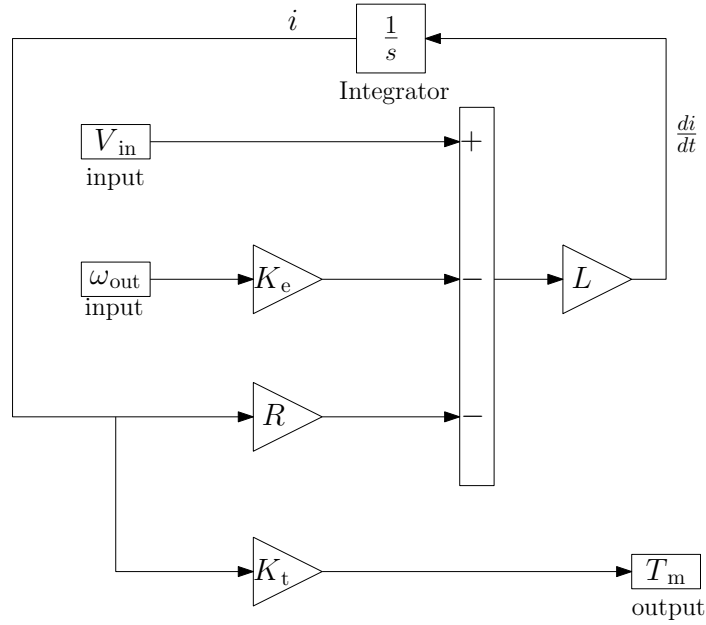


FIGURE 3.4: Block diagram showing the DC motor adapted from [5]

The direct voltage to torque transfer function for the motor input voltage, $V(s)$, to motor torque $T(s)$, is shown in Equation (3.9)

$$\frac{V(s)}{T(s)} = \frac{K_t}{L_s + R} \quad (3.9)$$

3.1.4 PID Controller

The parallel PID controller in the form shown in Figure 3.5 was used.

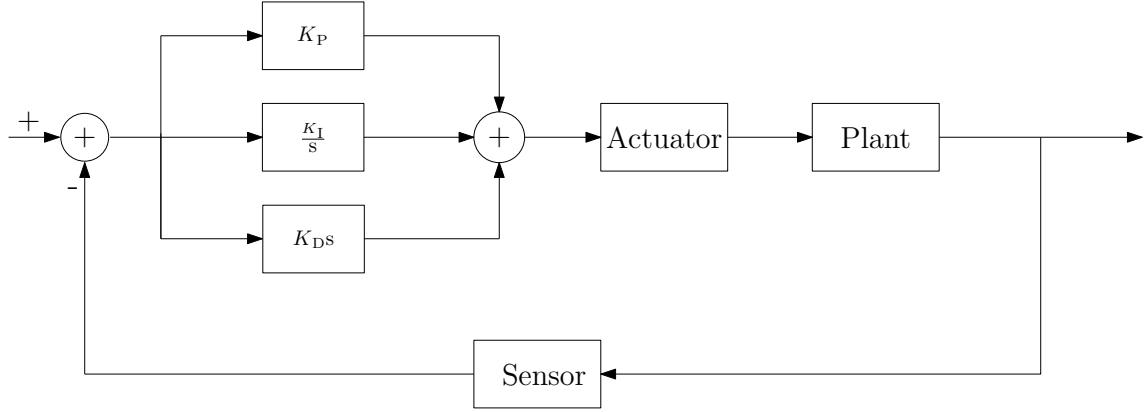


FIGURE 3.5: PID controller structure adapted from [7]

From Figure 3.5, the PID controller has the following gains; K_P representing the proportional constant of the controller, K_I representing the integral constant of the controller and lastly the K_D representing the derivative constant of the controller [37].

Some systems do not necessarily have to make use of all the components in the PID controller because each component in the controller contributes differently to reduce the system error. Some systems do not require all components to reduce the error in the system. The following sections describe how each term of the PID controller affects the controller output.

3.1.4.1 Proportional Gain

The proportional gain component in the controller acts on the present value of the error. That is why the proportional control mode is important when the rate of change of error is high as it improves the transient response [127]. Increasing K_P will only reduce the steady-state error but not diminish it. The K_P gain would have to be increased to infinity to diminish the steady-state error, which is impossible [39]. Increasing the proportional gain decreases the rise time, making the system response faster.

Furthermore, increasing the K_P gain decreases the error; however, the system becomes more oscillatory [37]. Note that increasing K_P only after a certain limit causes the response to overshoot [39]. These observations are mainly based on the type of plant; hence it is important to observe responses with the specific plant of interest [39].

How the system responds also depends on the order of the system. In second-order systems, for example, the P controller generally gives the output described above [39]. Increasing K_P gain decreases the gain and phase margin in the frequency domain, which is disadvantageous [39]. This increase also results in a larger sensitivity to noise [37].

There are a few ways of diminishing the steady-state error discussed above. One method is to add the integral controller, making a PI controller.

3.1.4.2 Integral Gain

The integral gain acts on the average of past errors. Because the integral gain calculates the average of past errors, it can detect the steady-state error and diminish it. Therefore, this system improves the steady-state response [127]. However, using this controller may result in a negative impact on the speed and overall stability of the system. Thus this controller should only be used when the speed of response of the system is not a critical issue [39]. Additionally, because the integral control cannot predict future error, it cannot reduce and diminish oscillations [39]. After a limit, increasing K_I gain will cause oscillations. Hence, the need for the derivative term of the PID controller.

3.1.4.3 Derivative Gain

The derivative gain predicts future errors based on linear extrapolation of the signal. Increasing the derivative gain K_D results in the oscillation, overshoot, and settling time of the system decreasing [127]. Some systems make use of only the

PD controller, without including the integral term, if the specific plant does not require a small steady-state error.

The problem which may arise in the derivative controller is that it amplifies high-frequency signals, and those signals could be noise. That is evident because as the frequency becomes large, this tends the derivative controller to infinity since the algorithm is $K_D \times s$, with $s = j\omega$. Also, the higher the frequency of a sinusoidal signal, the larger the tangential line of that signal (i.e. the larger the derivative). Introducing a low-pass filter changes the definition of the derivative section of the algorithm from $K_D \times s$ to definition (3.10),

$$K_D \times \frac{N}{1 + N\frac{1}{s}} \quad (3.10)$$

where N is the filter coefficient which determines the cut off frequency for the low-pass filter. This change in the derivative section then changes the block diagram of the controller from Figure 3.5 to Figure 3.6.

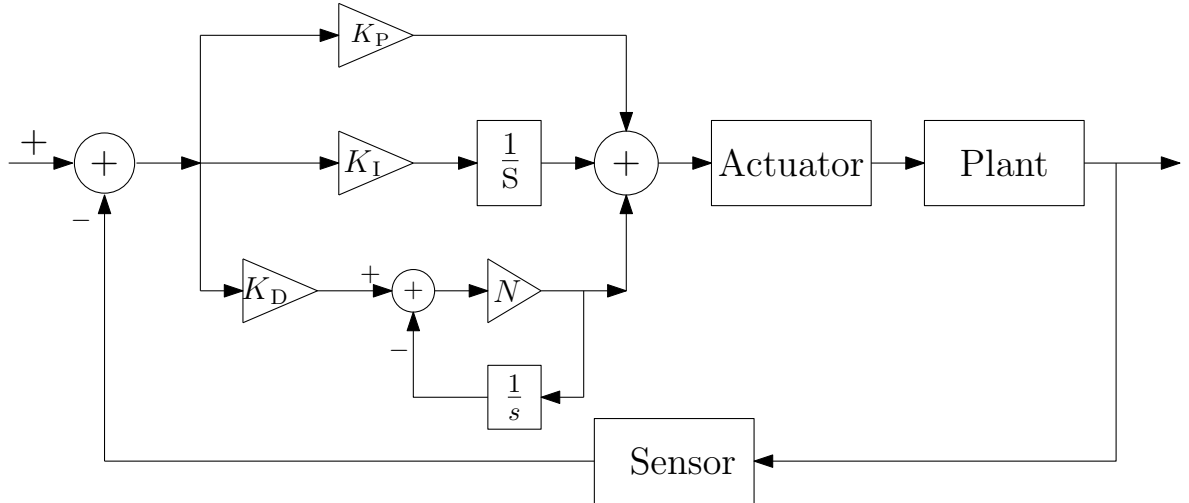


FIGURE 3.6: PID controller structure with an added low-pass filter

Choosing the filter coefficient is not as simple as one might think. In most cases, the signal that is of interest usually has a low frequency but high amplitude, and the noise signal usually has low amplitude but high frequency. However, this is not always the case. Choosing a high cut-off frequency may result in increasing the amount of noise in the system, and choosing a low cut-off frequency may result

in cutting off some of the signal that is of interest. The advantage of using meta-heuristics as the tuning method is that it allows for an additional variable that must be tuned, and that is the filter coefficient. Thus, the algorithms tuned the three PID controller gains, and the filter coefficient gain N .

3.1.5 Added Non-Linearities

3.1.5.1 Torque Disturbance due to Friction

Additional friction was modelled and included in the gimbal system to observe how the different algorithms behave when considering added non-linearities in the plant. From Newton's 2nd law, this then changes the Equation (3.4) to

$$T_m + T_d = J_g^* \alpha_g \quad (3.11)$$

where T_d is the net disturbance torque, caused by friction in this case. This frictional force is considered as acting on the rotational elements of the gimbal [5]. Modelling friction can be done in many ways because the topic is quite a broad and complex phenomena [5] [8]. For engineering purposes, friction contains three components; the static, coulomb and the viscous friction [8]. Static friction refers to the friction that the motor torque needs to overcome in order for the system to move. Coulomb friction occurs during motion and is dependent on the direction of the velocity. Viscous friction also occurs during motion and is proportional to the velocity.

There are two methods of modelling friction: static friction modelling and dynamic friction modelling. Static friction models include an attempt to model the Stribeck effect shown in Figure 3.7. The Stribeck effect is where the frictional force or torque decreases as velocity increases for a particular velocity regime [8]. Dynamic friction modelling, however, takes into account the effect of not only velocity on the frictional force but also material displacement. The model used this study is

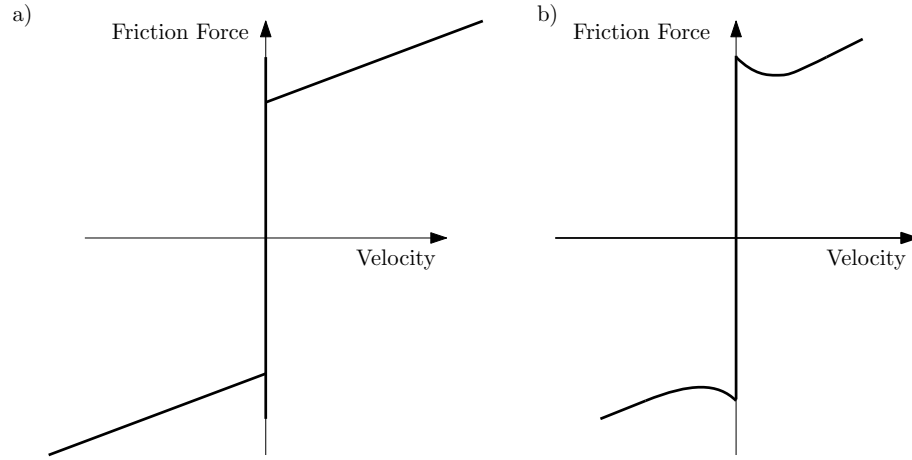


FIGURE 3.7: A representation of the relationship between friction force and velocity adapted from [8]. (a) Coulomb, viscous and static friction. (b) Stribeck friction model

part of the static friction modelling and is named the Tustin model [8], shown in the Equation (3.12),

$$T_f = (T_c + (T_s - T_c)e^{\frac{-|\omega|}{\omega_s}})sign(\omega) + T_v\omega, \quad (3.12)$$

where T_c is the Coulomb friction torque in Nm, T_s is the static friction torque in Nm, ω is the angular velocity in rad/s, ω_s is the stick friction transition rate in rad/s, and T_v is the viscous friction torque coefficient in Nm/rad/s. The values for these quantities are shown in Table 3.4.

TABLE 3.4: Friction values taken from [11]

Friction Type	Friction Value
Static Friction Coefficient T_s	0.2 Nm
Viscous Friction Coefficient T_v	0.025 Nm/rad/s
Coulomb Friction T_c	0.7 T_s
Transition Rate ω_s	0.01 rad/s

3.1.5.2 Torque Disturbance due to Base Motion and Mass Unbalance

Base motion occurs when the movement of the vehicle that the gimbal is mounted causes disturbance in the gimbal field of view. This base motion can cause unwanted disturbance torque because of static mass unbalance.

The static mass imbalance occurs from the centre of gravity of the gimbal not being in the pivot point of the gimbal [128]. This is due to various components like the structure design error, material defects, machining or assembly defects [128].

The gimbal should be able to reject the base motion such that the final LOS signal is not affected. How well the base motion is rejected depends on the application. In some instances, it is sufficient to reject motion such that the target remains within the field of view (FOV). In other instances, like in a laser range finder, the LOS must remain stabilized such that the target remains at the centre of the FOV of the stabilized sensor [29].

The disturbance torque is calculated using the equation

$$T_d = a \times m \times r_{\text{offset}}, \quad (3.13)$$

where a represents the base vibration acceleration, m represents the gimbal mass, and r_{offset} represents the centre of mass offset distance from the rotational axis. These non-linearities were modelled using band-limited white noise block in Simulink [5].

The vibration acceleration used for approximation is taken from [129], where the study measured the vibration level for helicopters and found to have frequency spectrum peaks from 0.84 to 1.68 m s^{-2} . For an average value for the white noise level, the acceleration was taken to be 0.5 m s^{-2} . The offset radius was taken to be 5mm. Thus, using the gimbal mass stated in Table 3.2 of 12.3 kg, and Equation (3.13), the torque disturbance was calculated to be

$$T_d = (0.5)(12.3)(0.005) = 0.03075 \text{ N m}. \quad (3.14)$$

The torque T_d must be squared to determine the PSD torque level to apply to the Simulink noise generation. This squaring equates to

$$PSD = (0.03075)^2 = 0.000941 (\text{N m})^2/\text{Hz}. \quad (3.15)$$

3.2 Algorithm Details

This section details the algorithms chosen, including how they were implemented in this study and important considerations for implementing these algorithms in this study.

3.2.1 The Genetic Algorithm

As stated previously, the GA is based on the three core principles from the Darwinian evolution, which are heredity, variation and selection.

Each member of the population created encompasses ‘DNA’. DNA, in this context, is described as having a set of properties that govern the look and behaviour of population members. Once a randomly generated population of DNA is created, the next phase of the GA is the selection phase. There are two parts to the selection phase, namely evaluating fitness, and creating a mating pool [55].

A fitness function is used to numerically evaluate the fitness of each member of the population. The fittest population members survive to the next generation. This mimics natural selection. These survivors are then called the mating pool because this is where the parents of the new offspring are chosen for reproduction. The selection rate X_{rate} is what determines how many chromosomes or population members will survive to the next generation. Having a small selection rate limits availability in genes for the new population, whereas, having a high selection rate

allows for non-fit members and their bad traits to survive. Another method of determining the mating pool is to use thresholding. Thresholding is where all chromosomes which have a fitness score above some threshold survive [59]. In this study, however, the top 50% of the population (i.e. 50% of the population with the best fitness scores) will be part of the mating pool. Thus, the selection rate is chosen as 50%.

From the mating pool, parents need to be chosen for reproduction. There are different ways of doing this. Tournament selection and roulette wheel are the standard selection methods for most GAs. Tournament selection involves randomly choosing a small subset of population members (tournament size), and the population member with the best fitness score from this, becomes the parent. This process repeats itself until there are enough parents to reproduce the full population size.

Roulette wheel is done by assigning probabilities to chromosomes in the mating pool that are either proportional to their fitness score (cost weighting) or to their rank in the population (rank weighting). The chromosomes that have better fitness scores or better ranks in the population then have a higher chance of being selected to be parents [59]. Once the parents are selected, the reproduction phase begins.

The most common way of reproduction is by using two parents to produce two offsprings [59]. Crossover is one method used. Crossover entails choosing one or more chromosomes in the DNA to be the crossover points. The variables between these crossover points (represented as the arrows) are exchanged to produce the offspring as shown:

$$parent_1 = [p_{m1}, p_{m2}, p_{m3}, p_{m4}, p_{m5}, \dots, p_{mN_{var}}] \quad (3.16)$$

$$parent_2 = [p_{d1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, \dots, p_{dN_{var}}] \quad (3.17)$$

$$offspring_1 = [p_{m1}, p_{m2}, \uparrow p_{d3}, p_{d4}, \uparrow p_{m5}, \dots, p_{mN_{var}}] \quad (3.18)$$

$$offspring_2 = [p_{d1}, p_{d2}, \uparrow p_{m3}, p_{m4}, \uparrow p_{d5}, \dots, p_{dN_{var}}] \quad (3.19)$$

Crossover is a popular method that entails combining both the parent's genetic code to create a child. There are different types of crossover methods, dependent

on the number of crossover points. Single-point crossover is where there is only one crossover point, and the chromosomes after that point are exchanged [59]. There is also two-point crossover where there are two crossover points and uniform crossover where there are crossover points at each chromosome. The reader is referred to [130] for more information. Crossover merely introduces new combinations in the material and does not introduce new genetic material which shows why mutation is needed.

Mutation is usually performed after crossover and before the offspring has been added to the next generation. There are two factors to mutation: type of mutation, and rate of mutation [59]. Some types of mutation include uniform mutation, non-uniform mutation, order k mutation, and simple variation of day mutation. Gaussian mutation is categorised under uniform mutation [59]. For more information on this, the reader is referred to [84] [130]. Mutation is not a necessary step but is only done to introduce additional variety to the evolution process [55]. Mutation rate is described using a percentage. For example, a 1% mutation rate means that for each DNA, there is a 1% probability that it will experience mutation. Mutation is usually not conducted on the best population member, which allows for elitism. Elitism allows the population member to continue to survive without ‘ruining’ what already works. Once the mutation is complete, the offspring is added to the next generation. The process mentioned above continues until the algorithm reaches the maximum number of iterations.

The different parameters of the algorithm aid in balancing exploration and exploitation, or diversification and intensification. Exploration/diversification is what allows an algorithm to explore new regions in the search space, and exploitation/intensification is what fine-tunes the solutions from the best-chosen regions in the search space. The GA is known for slow or premature convergence [59] [130], which means that it does not perform enough exploration but quickly goes to exploitation.

Having a high mutation rate increases diversity in the population and allows the algorithm to explore more regions in the solution space. Performing crossover

does not introduce new genetic material but rather fine-tunes existing solutions by checking different combinations, and thus performs exploitation.

In this study, the crossover type implemented is the single-point crossover. The blending method was used along with single-point crossover to ensure that the algorithm does not purely rely on the mutation to introduce new genetic material [59]. This is where the two offspring members, p_{new1} and p_{new2} , are calculated using Equation 3.20.

$$p_{\text{new1}} = p_m\alpha - \beta(p_m\alpha - p_d\alpha) \quad (3.20)$$

$$p_{\text{new2}} = p_d\alpha + \beta(p_m\alpha - p_d\alpha) \quad (3.21)$$

where α is the randomly chosen variable to be the crossover point, β = random number between 0 and 1, $p_m\alpha$ is the mother chromosome in the crossover point, and $p_d\alpha$ is the crossover point in the father chromosome.

Since this study compared how the parameters of the GA are chosen and how this affects performance, Table 3.5 shows the difference in the implementation of the static and adaptive/dynamic GA. Figure 3.8 shows a flowchart of the dynamic GA implemented. Algorithm 1 and 2 show the implementation of the static and dynamic GA, respectively. Algorithm 3 shows how the dynamic parameters were obtained. Section 3.2.1.1 details how the dynamic parameters of the adaptive GA were calculated.

TABLE 3.5: Parameters of the GA compared

Parameters	Static GA taken from [59]
Crossover Probability	No crossover probability. All chromosomes undergo crossover
Selection Type	Rank weighting
Mutation Rate	Static mutation rate of 0.2
Parameters	Adaptive/Dynamic GA taken from [9]
Crossover Probability	The crossover probability is adaptable
Selection Type	Tournament selection with adaptable tournament size
Mutation Rate	Adaptable mutation rate

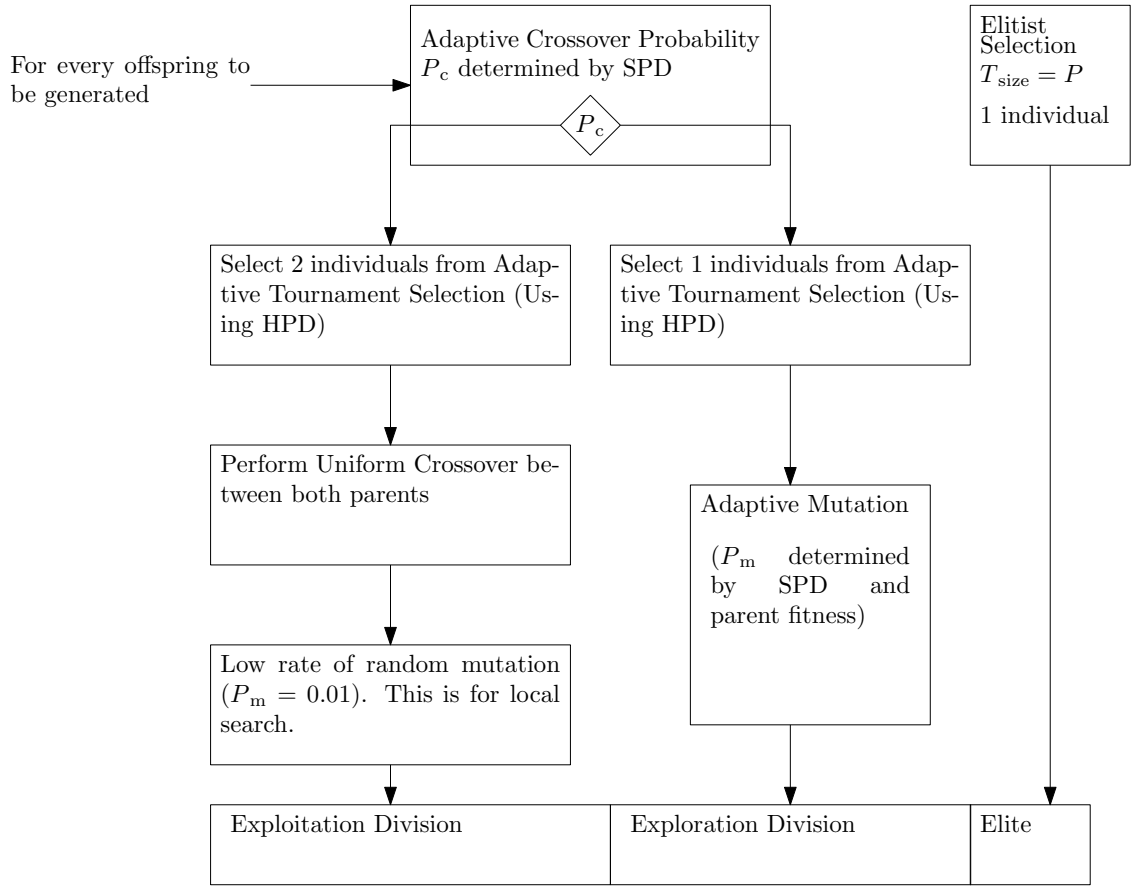


FIGURE 3.8: Flowchart showing dynamic GA adapted from [9]

3.2.1.1 Dynamic Parameters

The GA has various algorithm-specific parameters which need to be tuned accordingly. This includes the mutation rate, crossover rate, and the tournament size. Because the GA is known to converge locally, the study in [9] proposed that these parameters are to be adapted based on maintaining a level of diversity in the algorithm population. This is done by evaluating the Standard Population Diversity (SPD) and the Healthy Population Diversity (HPD) and calculating the parameters based on these.

The SPD is defined as a quantitative value describing the level of variation in the population [131] [132] [133] [134]. This means it shows how far the values of the population are from each other. In a population consisting of R population members, calculating the SPD first involves computing the variable-wise average

Algorithm 1 Static Genetic Algorithm pseudo code

Require:

R ▷ Population size
 M ▷ Iteration number
 $varhi, varlo$ ▷ Search space bounds for population
 N_{var} ▷ Number of variables
 μ ▷ Mutation rate
 X_{rate} ▷ Selection rate
 $f(X)$ ▷ Objective function

- 1: $Pop \leftarrow (varhi - varlo) \times \text{random number}(R, N_{var}) + varlo$ ▷ Population initialization $R \times N_{var}$ matrix
- 2: **for** $j = 1 : R$ **do** ▷ $\mathbf{O}(R)$
- 3: Evaluate population using $f(X)$ ▷ $\mathbf{O}(R)$
- 4: **end for**
- 5: Sort population according to best fitness ▷ $\mathbf{O}(R \log(R))$
- 6: **function** GA($R, M, N_{var}, \mu, X_{rate}$)
- 7: **for** $t = 1 : M$ **do** ▷ $\mathbf{O}(M)$
- 8: $N_{keep} \leftarrow R \times X_{rate}$ ▷ Population members that will be kept
- 9: $nmut \leftarrow (R - 1) \times N_{var} \times \mu$ ▷ Number of mutations
- 10: $N_{mate} \leftarrow \frac{R - N_{keep}}{2}$ ▷ Number of matings
- 11: **procedure** SELECTION
- 12: $pick_1 \leftarrow \text{random number}(1, N_{mate})$ ▷ For ma
- 13: $pick_2 \leftarrow \text{random number}(1, N_{mate})$ ▷ For pa
- 14: $P_j \leftarrow \frac{N_{keep} - j + 1}{\sum_{j=1}^{N_{keep}} j}$ ▷ $N_{keep} \times 1$ probability for selection matrix
- 15: $C_p \leftarrow \sum_{i=1}^j P_j$ ▷ $1 \times N_{keep}$ cumulative probabilities matrix
- 16: **while** $ind \leq N_{mate}$ **do** ▷ $\mathbf{O}(N_{mate})$
- 17: **for** $k = 2 : keep + 1$ **do** ▷ $\mathbf{O}(N_{keep})$
- 18: **if** $pick_1(ind) \leq C_p(k)$ and $pick_1(ind) > C_p(k - 1)$ **then**
- 19: $ma(ind) \leftarrow k - 1$
- 20: **end if**
- 21: **if** $pick_2(ind) \leq C_p(k)$ and $pick_2(ind) > C_p(k - 1)$ **then**
- 22: $pa(ind) \leftarrow k - 1$
- 23: **end if**
- 24: **end for**
- 25: **end while**
- 26: **end procedure**
- 27: **procedure** CROSSOVER
- 28: $\beta \leftarrow \text{random number}(1, N_{mate})$ ▷ mixing parameter
- 29: **for** $s = 1 : N_{mate}$ **do** ▷ $\mathbf{O}(N_{mate})$
- 30: $\rho_{new1} = \rho_{m\alpha} - \beta[\rho_{m\alpha} - \rho_{d\alpha}]$
- 31: $\rho_{new2} = \rho_{d\alpha} - \beta[\rho_{m\alpha} + \rho_{d\alpha}]$
- 32: $offspring_1 = [\rho_{m1}, \rho_{m2} \dots \rho_{new1} \dots \rho_{dN_{var}}]$
- 33: $offspring_2 = [\rho_{d1}, \rho_{d2} \dots \rho_{new2} \dots \rho_{mN_{var}}]$
- 34: **end for**
- 35: **end procedure**

- 36: **procedure** MUTATION
- 37: $mrow \leftarrow \text{random number}(1, nmut) \times R - 1$
- 38: $mcol \leftarrow \text{random number}(1, nmut) \times N_{var}$
- 39: **for** $ii = 1 : nmut$ **do** ▷ $\mathbf{O}(nmut)$
- 40: $Pop(mrow(ii), mcol(ii)) = (varhi(1, mcol(ii)) - varlo(1, mcol(ii))) \times \text{random number}(0, 1)$
 $+ varlo(1, mcol(ii))$
- 41: **end for**
- 42: **end procedure**
- 43: Evaluate new population ▷ $\mathbf{O}(R)$
- 44: Sort according to best fitness ▷ $\mathbf{O}(R \log(R))$
- 45: **end for**
- 46: **end function**

Algorithm 2 Adaptive Genetic Algorithm pseudo code

```

Require:
 $R$  ▷ Population size
 $M$  ▷ Iteration number
 $N_{\text{var}}$  ▷ Number of variables
 $f(X)$  ▷ Objective function
1: for  $j = 1 : R$  do ▷  $\mathcal{O}(R)$ 
2:   Initialize population ▷  $R \times N_{\text{var}}$  matrix
3:   Evaluate population using  $f(X)$  ▷  $\mathcal{O}(R)$ 
4: end for
5: Sort population according to best fitness ▷  $\mathcal{O}(R \log(R))$ 
6: function GA( $R, M, N_{\text{var}}$ , algorithm 3) ▷  $\mathcal{O}(M)$ 
7:   for  $t = 1 : M$  do
8:      $keep \leftarrow R \times X_{\text{rate}}$  ▷ Population members that will be kept
9:     Run ACROMUSE procedure and obtain values for  $P_c, P_m$  and  $T_{\text{size}}$  ▷  $\mathcal{O}(R) + \mathcal{O}(N_{\text{var}})$ 
10:    procedure EXPLOITATION SECTION
11:      Population to be exploited is  $N_{\text{exploit}} = P_c \times R$ 
12:      Select parents from  $T_{\text{size}}$  and implement tournament competition ▷  $\mathcal{O}(N_{\text{exploit}}/2)$ 
13:      Perform single point crossover with blending method ▷  $\mathcal{O}(R)$ 
14:      Use low rate of mutation of  $P_{\text{mstatic}} = 0.01$  to mutate and generate offspring ▷  $\mathcal{O}(R)$ 
15:    end procedure
16:    procedure EXPLORATION SECTION
17:      Population size to be explored is  $N_{\text{explore}} = (P_c - 1) \times R$ 
18:      Select one individual from tournament selection using  $T_{\text{size}}$  ▷  $\mathcal{O}(R)$ 
19:      Perform mutation using  $P_m$  obtained from ACROMUSE ▷  $\mathcal{O}(R)$ 
20:    end procedure
21:    procedure ELITIST SECTION
22:      Best individual does not get mutated, and survives to next generation
23:    end procedure
24:    Evaluate new population ▷  $\mathcal{O}(R)$ 
25:    Sort according to best fitness ▷  $\mathcal{O}(R \log(R))$ 
26:  end for
27: end function

```

Algorithm 3 ACROMUSE pseudo code

```

1: procedure SPD CALCULATION
2:   Calculate the variable-wise average  $G_n^{\text{ave}}$  ▷  $\mathcal{O}(R)$ 
3:   Calculate  $SPD_i$  ▷  $\mathcal{O}(N_{\text{var}})$ 
4:   Calculate the gene-wise standard deviation  $\sigma(G_n^{\text{ave}})$  ▷  $\mathcal{O}(N_{\text{var}})$ 
5:   Calculate  $SPD$  ▷  $\mathcal{O}(N_{\text{var}})$ 
6: end procedure
7: procedure HPD CALCULATION
8:   Calculate  $w_i$  ▷  $\mathcal{O}(R)$ 
9:   Calculate gene-wise weighted average  $G_n^{\text{W.ave}}$  ▷  $\mathcal{O}(R)$ 
10:  Calculate  $HPD_i$  ▷  $\mathcal{O}(N_{\text{var}})$ 
11:  Calculate the gene-wise standard deviation  $\sigma(G_n^{\text{W.ave}})$  ▷  $\mathcal{O}(R)$ 
12:  Calculate  $HPD$  ▷  $\mathcal{O}(N_{\text{var}})$ 
13: end procedure
14: procedure CALCULATION OF PARAMETERS
15:   Calculate  $P_c$ 
16:   Calculate  $P_m$ 
17:   Calculate  $T_{\text{size}}$ 
18: end procedure

```

for all R individuals in the population as follows:

$$G_n^{\text{ave}} = \frac{1}{R} \sum_{i=1}^R G_{i,n}, \quad (3.22)$$

with each population member i consisting of N_{var} variables with G_{in} referring to the n^{th} variable of the population member i , $G_i = (G_{i,1}, G_{i,2}, \dots, G_{i,N_{\text{var}}})$. From this, SPD_i is calculated. This value refers to the population member i 's contribution to SPD. It is calculated using the euclidean distance between the individual i and G^{ave} shown on Equation (3.23),

$$SPD_i = \sqrt{\sum_{n=1}^{N_{\text{var}}} (G_{i,n} - G_n^{\text{ave}})^2}. \quad (3.23)$$

Though the summation of SPD_i should calculate SPD, the summation cannot be normalized according to the mean. If SPD is not normalized, the SPD measure will vary for different problems and populations. For a normalized SPD, a standard deviation measure must be employed. This is calculated as the gene-wise standard deviation over all R individuals using Equation (3.24),

$$\sigma(G_n^{\text{ave}}) = \sqrt{\frac{1}{R} \sum_{i=1}^R (G_{i,n} - G_n^{\text{ave}})^2}. \quad (3.24)$$

Due to this approach, the standard deviation is expressed relative to the mean, using the coefficient of variation (C_v). This coefficient of variation is used as the measure of SPD, shown on Equation (3.25),

$$SPD = C_v(G^{\text{ave}}) = \frac{1}{N_{\text{var}}} \sum_{j=1}^{N_{\text{var}}} \left(\frac{\sigma(G_j^{\text{ave}})}{G_j^{\text{ave}}} \right). \quad (3.25)$$

The HPD is different from the SPD in that it measures a spread of healthy individuals (i.e. individuals with a high fitness score) across the population, rather than purely looking at the spread of individuals in the search space. This is why HPD is referred to as the fitness-weighted measure of population diversity. The

contribution to diversity in the solution space of each individual is according to its fitness. The first step to calculating the HPD is first to obtain the individual's fitness expressed as a proportion of total fitness using Equation (3.26),

$$w_i = \frac{f_i}{\sum_{k=1}^R f_k}, \quad (3.26)$$

where f represents the fitness value. From this, the weighted average individual is calculated by summing the fitness-weighted gene-wise average across all R individuals using the Equation (3.27),

$$G_n^{\text{W.ave}} = \sum_{i=1}^R w_i G_{i,n}. \quad (3.27)$$

HPD_i is the individual i 's contribution to healthy diversity. This is calculated using the Equation (3.28),

$$HPD_i = w_i \sqrt{\sum_{i=1}^{N_{\text{var}}} (G_{i,n} - G_n^{\text{W.ave}})^2}. \quad (3.28)$$

The normalised HPD, much like the SPD, is computed by calculating the population's gene-wise fitness weighted standard deviation using the Equation (3.29),

$$\sigma(G_n^{\text{W.ave}}) = \sqrt{\sum_{i=1}^R w_i (G_{i,n} - G_n^{\text{W.ave}})^2}. \quad (3.29)$$

Thus HPD is then calculated by obtaining the weighted coefficient of variation using Equation (3.30),

$$HPD = C_v(G^{\text{W.ave}}) = \frac{1}{N_{\text{var}}} \sum_{j=1}^{N_{\text{var}}} \left(\frac{\sigma(G_j^{\text{W.ave}})}{G_j^{\text{ave}}} \right). \quad (3.30)$$

The value of SPD, shown on Equation (3.25) is used to calculate the crossover probability P_c . The crossover probability determines the size of the exploration

and exploitation divisions. A high crossover probability means an increase in the exploitation division. P_c is calculated using Equation (3.31),

$$P_c = \left(\frac{SPD}{SPD_{\max}} \times (K_2 - K_1) \right) + K_1, \quad (3.31)$$

where P_c ranges from $K_1 = 0.4$ to $K_2 = 0.8$. SPD_{\max} is set to 0.4 because according to [9], this is what is regular practice.

Adaptive mutation probability is calculated using two components, mutation due to diversity and mutation due to fitness. Mutation due to diversity is calculated using Equation (3.32),

$$P_m^{\text{Diversity}} = \frac{SPD_{\max} - SPD}{SPD_{\max}} \times K, \quad (3.32)$$

where $K = 0.5$, which is the upper-bound of P_m . Mutation probability due to fitness is calculated using Equation (3.33),

$$P_m^{\text{Fitness}} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \times K, \quad (3.33)$$

where f is the parent fitness. Equation (3.34) describes how the mutation probability is then obtained

$$P_m = \frac{P_m^{\text{Fitness}} + P_m^{\text{Diversity}}}{2}. \quad (3.34)$$

The tournament size is calculated using HPD shown on Equation (3.30). This is done using Equation (3.35),

$$T_{\text{size}} = \frac{HPD}{HPD_{\max}} \times T_{\text{size.max}}, \quad (3.35)$$

where $T_{\text{size.max}}$ refers to the maximum selection pressure applied to the population. This value is calculated as $PopulationSize/6$. HPD_{\max} is the maximum HPD possible and is set to 0.3.

3.2.2 The Flower Pollination Algorithm

The FPA was developed by Yang in 2012 and inspired by the pollination of flowers in nature [60]. Pollination is the transfer of pollen from the male anther to the female stigma of a flower in order for flower reproduction to occur. This can occur by cross-pollination or self-pollination. Cross-pollination, also referred to as biotic or allogamy [60], occurs when pollen grains are transferred from one plant to the next using a pollinator, like an insect. Whereas self-pollination, also referred to as abiotic [60], occurs when the plant does not use a pollinator and sheds its pollen from its anther to its stigma [135]. Cross-pollination is then considered as global-pollination and self-pollination as local-pollination. This is mainly because pollinators, such as bats or birds, can travel long distances while pollinating [60].

As discussed above, there are two methods of pollination. However, 90% of flowering plants are pollinated via cross-pollination or global-pollination [60].

The probability that controls the local and global pollination is a switch probability (p), which has the range $[0,1]$. Physical factors such as wind may increase local pollination probability. However, a probability switch value of 0.8 works best in most applications [60]. The FPA is easier to implement, as compared to other algorithms because it only has a few parameters, namely the switch probability (p) and a scaling factor (α) [136].

The four rules of the FPA are [60]:

1. Biotic or cross-pollination is considered as global-pollination with pollinators traveling using the Lévy flight distribution.
2. Abiotic or self-pollination is considered as local-pollination.
3. Flower constancy, also seen as the reproduction probability, is proportional to the similarity of the two flowers involved in the pollination process.
4. The switch probability controls whether global or local-pollination is performed.

Mathematical derivations are developed from these assumptions and above rules to explain local and global-pollination.

The scaling factor λ value in this study was 1.5, adapted from [60]. However, this study compared two ways to implement the FPA; the static method where switch probability value is 0.8, and the adaptive method where the switch probability changes as the algorithm iterates. Section 3.2.2.1 shows how the dynamic switch probability is calculated.

3.2.2.1 Dynamic Switch Probability

The switch probability (p) is what balances the diversification and intensification of the algorithm. The method of how the dynamic switch probability is calculated is adapted from [78] and shown in Equation (3.36),

$$p = p_{-1} - 0.015 \times \left(\frac{M - t}{M} \right), \quad (3.36)$$

where p_{-1} is the value of the switch probability in the previous iteration, M is the maximum number of iterations, and t is the current iteration of the algorithm. Figure 3.9 shows the change of the switch probability value as the iteration number progresses, based from Equation (3.36). The switch probability starts from 0.9 (90%) and decreases using the step size of 0.015 as the iterations progress in order to increase the probability of local pollination.

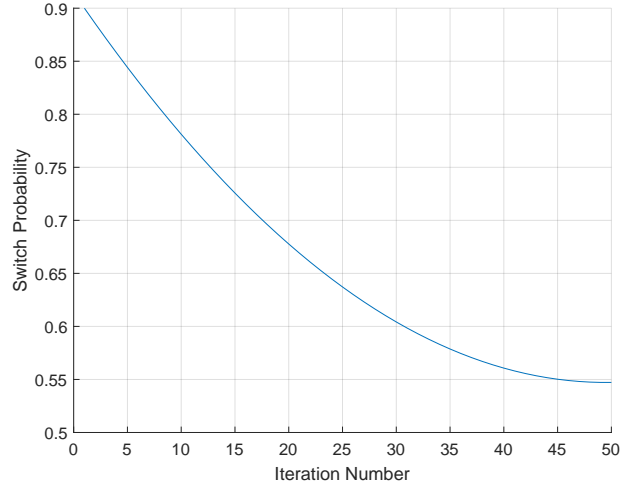


FIGURE 3.9: Diagram showing the change of the switch probability value p as the iteration number progresses

3.2.2.2 Global Pollination

For global pollination, the mathematical representation is as follows:

$$x_i^{t+1} = x_i^t + L(x_i^t - g^*), \quad (3.37)$$

where x_i^t is the pollen i or the solution vector x_i^t at iteration t , g^* is the current best solution found in the current iteration; L is used to represent the step size. There are many different ways to determine step size. One way is using the Lévy flight distribution, which is shown below:

$$L(s) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (3.38)$$

where $\Gamma(\lambda)$ represents the standard gamma function, with this study using $\lambda = 1.5$ adapted from [60]. This distribution is valid for steps $s > 0$.

3.2.2.3 Local Pollination

For local pollination, the mathematical representation is as follows:

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t), \quad (3.39)$$

where $x_j^t - x_k^t$ are pollens from different flowers on the same plant.

Algorithm 4 shows the implementation of the FPA.

Algorithm 4 Flower Pollination Algorithm pseudo code

```

Require:
   $R$  ▷ Population size
   $M$  ▷ Iteration number
   $varhi, varlo$  ▷ Search space bounds for population
   $N_{var}$  ▷ Number of variables
   $f(X)$  ▷ Objective function
1:  $Pop \leftarrow (varhi - varlo) \times \text{random number}(R, N_{var}) + varlo$  ▷ Population initialization  $R \times N_{var}$  matrix
2: for  $j = 1 : R$  do ▷  $O(R)$ 
3:   Evaluate population using  $f(X)$  and find global best solution  $g^*$ 
4: end for
5: function FPA( $R, M, N_{var}$ )
6:    $\rho \leftarrow \text{random number}(0,1)$  ▷ for static switch probability
7:    $\rho \leftarrow \text{dynamic switch probability function } G(M)$ 
8:   for  $t < M$  do ▷  $O(M)$ 
9:     for  $i = 1 : R$  do ▷  $O(R)$ 
10:      if  $\text{random number} < \rho$  then
11:        Draw a  $N_{var}$ -dimensional step vector  $L$  obeys a Lévy distribution
12:        Do global pollination:  $x_i^{t+1} = x_i^t + L(x_i^t - g^*)$  ▷  $O(N_{var})$ 
13:      else
14:        Draw  $\epsilon$  from a uniform distribution in  $[0,1]$ 
15:        Randomly choose  $j$  and  $k$  among all the solutions
16:        Do local pollination:  $x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t)$  ▷  $O(N_{var})$ 
17:      end if
18:      Evaluate new solutions ▷  $O(R)$ 
19:      Update new solutions in population if they are better
20:      Update current best solution  $g^*$ 
21:    end for
22:  end for
23: end function

```

3.2.3 The Teaching-Learning-Based Optimization Algorithm

The Teaching-Learning-Based Optimization Algorithm (TLBO) algorithm is based on the interactions in the classroom between learners and teachers to improve learner marks for different subjects. There are two phases in the algorithm: the teacher phase and the learner phase. The teacher phase is where the learners gain

knowledge from the teacher, and the learner phase is where learners interact and gain knowledge from each other.

3.2.3.1 Teacher Phase

In this phase, the teacher attempt to increase the mean result of the class average [63]. Let R be the total number of learners and N_{var} be the total number of class subjects. In the i th iteration, the performance or mark of the k th learner, where $k = 1, 2, \dots, R$ in the j th subject, where $j = 1, 2, \dots, N_{\text{var}}$ is $X_{k,j,i}$. In this algorithm, the learner who performs the best becomes the teacher. The difference in mean results between the best learner (i.e. teacher) and the average result of the learners, $\bar{\Delta}_{k,j,i}$, is given by the following equation:

$$\bar{\Delta}_{k,j,i} = r_i(X_{k^*,j,i} - T_f M_{j,i}), \quad (3.40)$$

where r_i represents a random number in the range $[0,1]$, $X_{k^*,j,i}$ represents the result of the best learner (k^*) in the subject j and at iteration i , T_f represents the teaching factor which is found to work best when it is either 1 or 2 [63], and $M_{j,i}$ represents the average result of the learners in subject j and at iteration i . From equation (3.40), the solution is updated using the following equation:

$$X'_{k,j,i} = X_{k,j,i} + \bar{\Delta}_{k,j,i}, \quad (3.41)$$

where $X'_{k,j,i}$ is the updated value of $X_{k,j,i}$ and is only accepted if it results in better function value. Otherwise, the old solution is retained. All accepted values in the teacher phase become inputs to the learner phase.

3.2.3.2 Learner Phase

In this phase, the learners gain knowledge by interacting with each other. The learners gain knowledge from other learners who have acquired more knowledge than them. For a population size R and with j_t meaning the result for all subjects,

randomly¹ select two learners P and Q, such that $X'_{P,j_t,i} \neq X'_{Q,j_t,i}$. The learner phase is mathematically represented in the equations below:

$$X''_{P,j,i} = X'_{P,j,i} + r_i(X'_{P,j,i} - X'_{Q,j,i}) \quad \text{if} \quad X'_{P,j_t,i} < X'_{Q,j_t,i} \quad (3.42)$$

$$X''_{P,j,i} = X'_{P,j,i} + r_i(X'_{Q,j,i} - X'_{P,j,i}) \quad \text{if} \quad X'_{Q,j_t,i} < X'_{P,j_t,i} \quad (3.43)$$

This completes one iteration of the TLBO algorithm. This process continues until the algorithm reaches the desired outcome.

The advantage of TLBO is that it only requires non-algorithmic specific parameters, such as the population size and the number of generations. It does not require any algorithm-specific parameters [63]. In this particular study, each learner of the population has four subjects being the PIDN gains. The difference between the standard TLBO described above, and the TLBO algorithm implemented in this study is that there will not be the best learner for each subject. Instead, the best learner will be the one which the combination of the controller gains gives the best fitness function output. This means that (3.40) becomes

$$\bar{\Delta}_{k,i} = r_i(X_{k^*,i} - T_f M_i), \quad (3.44)$$

and (3.41) becomes

$$X'_{k,i} = X_{k,i} + \bar{\Delta}_{k,i}. \quad (3.45)$$

The learner phase equations (3.42) and (3.43) then also change to

$$X''_{P,i} = X'_{P,i} + r_i(X'_{P,i} - X'_{Q,i}) \quad \text{if} \quad X'_{P,i} < X'_{Q,i} \quad (3.46)$$

$$X''_{P,i} = X'_{P,i} + r_i(X'_{Q,i} - X'_{P,i}) \quad \text{if} \quad X'_{Q,i} < X'_{P,i} \quad (3.47)$$

¹The randperm MATLAB function is used to randomly select two learners. There is a remote chance that the selected learners could be the same; however, there have not been any studies on the disadvantages of this. The code implements an if and else loop, thus if the two selected learners are the same, equation (3.43) will be implemented.

Algorithm 5 The Teaching-Learning-Based Optimization Algorithm pseudo code

```

Require:
 $R$                                 ▷ Population size
 $M$                                 ▷ Iteration number
 $N_{\text{var}}$                             ▷ Population dimension
 $f(X)$                             ▷ Objective function
1: for  $j = 1 : R$  do
2:   Initialize population
3:   Evaluate population using  $f(X)$  and find best learner  $k^*$ 
4: end for
5: function TLBO( $R, M, N_{\text{var}}$ )
6:   for  $i = 1 : M$  do                                ▷  $\mathcal{O}(M)$ 
7:     for  $k = 1 : R$  do                                ▷  $\mathcal{O}(R)$ 
8:       procedure TEACHER PHASE
9:          $M_i \leftarrow$  mean student
10:         $T_f \leftarrow$  teaching factor
11:         $r_i \leftarrow$  random number(0,1)
12:        Find difference mean using best:  $\bar{\Delta}_{k,i} = r_i(X_{k^*,i} - T_f M_i)$     ▷  $\mathcal{O}(N_{\text{var}})$ 
13:        Find new solution using:  $X'_{k,i} = X_{k,i} + \bar{\Delta}_{k,i}$                 ▷  $\mathcal{O}(N_{\text{var}})$ 
14:        Evaluate new solution                                          ▷  $\mathcal{O}(R)$ 
15:        Replace old solution if new solution better
16:      end procedure
17:      procedure LEARNER PHASE
18:        Randomly choose two learners P and Q such that  $X'_{P,i} \neq X'_{Q,i}$ 
19:        Update learner P:  $X''_{P,i} = X'_{P,i} + r_i(X'_{P,i} - X'_{Q,i})$  if  $X'_{P,i} < X'_{Q,i}$     ▷  $\mathcal{O}(N_{\text{var}})$ 
20:        Update learner Q:  $X''_{Q,i} = X'_{Q,i} + r_i(X'_{Q,i} - X'_{P,i})$  if  $X'_{Q,i} < X'_{P,i}$     ▷  $\mathcal{O}(N_{\text{var}})$ 
21:      end procedure
22:      Evaluate new solution                                          ▷  $\mathcal{O}(R)$ 
23:      Replace old solution if new solution better
24:    end for
25:    Find the current best learner  $k^*$ 
26:  end for
27: end function

```

3.3 Performance Criteria for Evaluating Gimbal Stabilization System

There must be a desired performance standard which must be defined to analyse how the gimbal stabilization system has performed. This standard inherently evaluates the accuracy and stability of the control system by observing how well-tuned the controller with meta-heuristic techniques is [10].

3.3.1 Transient Response

Standard performance measures are typically based on a step input [10] shown in Figure 3.10. This figure also shows the basic figures of merit used to assess the performance of the time response of the system.

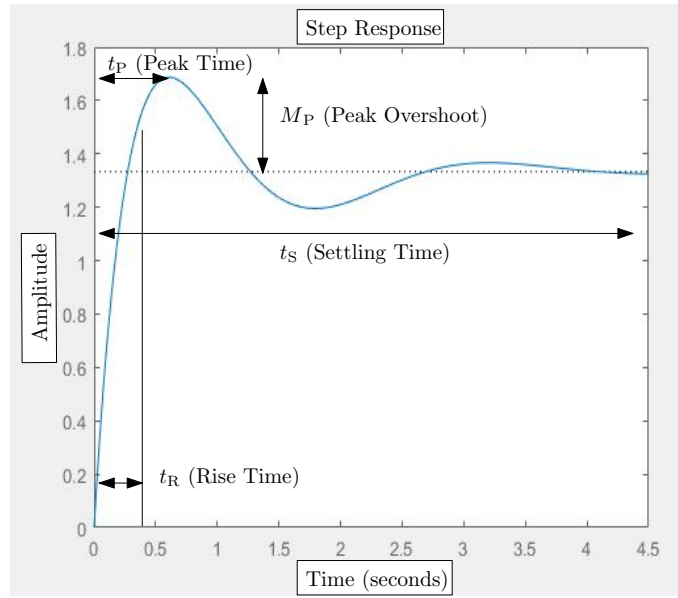


FIGURE 3.10: Basic figures of merit of a control system for the time-domain adapted from [10]

The rise time t_R , and peak time t_P , characterise how fast the system responds to a command input [10] [99]. The rise time is measured differently according to how the system is damped. For under-damped systems, the rise time is measured from 0 to 100% of the final value. However, for over-damped systems, the rise time is measured from 10% to 90% of the final value. The most popular method is using rise time from 10 to 90% of the final value and, thus, this was used for the current study. The peak time is not usually defined in over-damped systems [10] and is not used as a performance measure in this study. How well the system tracks the command input is measured by settling time t_S , percentage overshoot, and the steady-state error e_{ss} [10] [99]. Settling time also characterises the time required for the system response to follow the system input within a specified percentage range [10]. This range is usually within 2% of the input. The settling time was defined in this way in the current study.

The steady-state error refers to the error after the transient response has been diminished, having only the continuous error [10]. This merit provides a good indication of the stability of the system. The degree to which the response is well controlled is determined by the percentage overshoot and peak amplitude [99].

Percentage overshoot can be defined as

$$\text{Percentage overshoot} = \frac{M_P - A_f}{A_f} \times 100,$$

where M_P is the peak value of the time response and A_f is the final amplitude value of the response [10]. A shorter rise time, time constant and peak time may result in a larger overshoot and settling time. This means that these requirements are in contradiction, and thus a compromise must be established [10].

The question of what is deemed as good performance when observing these figures of merit is important. Since this a comparative study, it will focus on comparing the performance of the different algorithms rather than the values they obtain. However, it is still important to observe their performance with other similar systems. According to the study in [137] where the performance of a one-axis gimbal control system was observed, the standard performance requirements for the figures of merit are shown below:

- Rise Time ≤ 0.2 sec
- Settling time ≤ 0.3
- Maximum Overshoot ≤ 20 %

This is a comparison study, so observing how the algorithms perform in comparison to each other was more important than observing how they do in comparison to the performance requirements. However, these requirements were considered in order to compare the performance of the algorithms to the standard performance requirements.

3.3.2 Fitness Function

In an example of an optimization problem where the objective is to minimise the fitness function, let the solution space be S with $S \rightarrow \mathbb{R}$, with each solution represented by $x \subseteq S$. X is defined as the feasible solution space, where the

solutions in X are those that satisfy the constraints defined by the optimization problem. An optimal solution is defined as the solution $x^* \in X$ where the $f(x^*) \leq f(x)$ for all $x \in X$. Thus, the fitness function is vital in improving the performance of a meta-heuristic algorithm because it aids in choosing the best solution x^* . The fitness function is the goal which is intended to be achieved for the system that is being optimized [43].

As stated previously, the fitness function used in this study is shown on Equation (3.48),

$$F = \mu(V_{ISE}) + \alpha(V_{ITAE}), \quad (3.48)$$

with μ and α being the non-negative weights chosen to be 0.6 and 0.4 respectively, and V_{ISE} and V_{ITAE} are the values of the integral of the square of the error and integral time absolute square of the error respectively. How these performance indices are calculated are shown in Equations 3.49 and 3.50 respectively [10],

$$ISE = \int e^2(t)dt \quad (3.49)$$

$$ITAE = \int t | e(t) | dt \quad (3.50)$$

where $e(t)$ represents the error value for each time, and t represents the time.

3.4 Theoretical Analysis Details

The theoretical analysis used in this study is detailed here. This includes statistical and time complexity analysis. How the theoretical analysis was implemented in this study is discussed in this section.

3.4.1 Statistical Analysis

The statistical analysis quantifies whether the differences in the performance of the algorithms are significant. This is done by testing whether the null hypothesis H_0 is accepted or rejected.

The Friedman test answers whether there is a difference in the median or mean values of the populations for a set of k samples, where $k \geq 2$. In this research, k represents the algorithms, and n the problem instances. The data that was used was the mean fitness values of the different algorithms at the different problem instances.

The Friedman analysis was calculated using the program developed by Cardillo [138]. This program is different from the built-in Friedman analysis MATLAB function. This is because when k and n are large (≥ 10), the probability distribution is approximated to chi-square or F distribution. However, if k and n are small, which is the case in this research, this approximation becomes weak, and other means must be used to obtain the required values for this analysis, such as the p-values. The p-values are obtained from tables prepared explicitly for the Friedman analysis. The program in [138] addresses this.

The steps to calculating the Friedman test statistic are as follows:

1. Rank data from the best result (1) to worst (k). These ranks are denoted as $r_i^j (1 \leq j \leq k)$.
2. For each algorithm j , calculate the average ranks obtained for all the problem instances to obtain final rank $R_j = \frac{1}{n} \sum_{i=1}^n r_i^j$.

Once these ranks have been obtained, the Friedman statistic is then calculated by using Equation (3.51).

$$Q = \frac{12}{nk(k+1)} \left[\sum_{j=1}^k (\sum R_j)^2 \right] - 3n(k+1) \quad (3.51)$$

Once this test statistic Q is calculated, it is compared to the critical value obtained to observe if the *null* hypothesis can be rejected. If the Friedman statistic is larger than the critical value, then the *null* hypothesis is rejected, and thus a posthoc analysis is required. The program in [138] does this automatically, by using the posthoc test, which is a method equivalent to Fisher's least significant difference method described in [139].

3.4.2 Time Complexity of Algorithms

For an input n , the complexity was determined by a few rules of thumb obtained from [116]:

- The slowest instruction determines the asymptotic behaviour.
- $O(1)$ is a constant time algorithm.
- $O(n)$ is linear
- $O(n^2)$ is quadratic
- $O(\log(n))$ is logarithmic
- The nested loop in the programs is vital. A single loop over n items becomes $f(n) = n$. A loop within a loop becomes $f(n) = n^2$. A loop within a loop becomes $f(n) = n^3$.
- Sorting algorithm complexity is $O(n\log(n))$

3.5 Method

This section aims to illustrate the methodology followed to obtain the results. The method includes observing the design of the experiments conducted, the research environments used and the implemented tasks.

3.5.1 Computational Experimental Design

The Design of Experiments (DOE) methodology is used in statistics theory to provide clear experimental methods to evaluate stochastic problems. Changing an algorithm parameter one-factor-at-a-time is the most effective way to observe how each parameter affects the performance so that the parameter values are optimized [140]. This method, however, is also time-consuming and in most cases, impractical. This is why there is a need for 2^k factorial designs as they have shown to be more economical [140]. A 2^k factorial design is a basic DOE technique where the experimenter chooses two values for each algorithm parameter (k) that will be tested. These factors can either be quantitative (e.g. two different temperature values) or qualitative (e.g. two types of catalysts).

3.5.2 Common Parameters

The common parameters were optimized and chosen using the DOE methodology.

In this study, the common parameters that were optimized are:

- Search Space Bound (quantitative)
- Population Size (quantitative)

Choosing which two values will be used for the common quantitative parameters depends on the time and computational resources required for this application. The gimbal system that is used in this study is typically applied to the stabilization of an optical system in a helicopter for media coverage [141]. In this application, the gimbal must stabilize the optical system rapidly to ensure that the optical system performs best when covering and tracking what is required. This means there is a reasonable limitation on the time and computational resources given to the algorithm. This was kept in mind when selecting the two values for each parameter, as shown in Table 3.6.

TABLE 3.6: 2-level factorial design bounds

Factor	Level 1	Level 2
Search Space Upper Bound	100	500
Population Size	20	60

TABLE 3.7: Factorial design

A_D	Population Size	Search Space
1	20	100
2	60	100
3	20	500
4	60	500

The detail of each algorithm design is shown in Table 3.7.

Thus, each algorithm design A_D was compared in each experiment in order to find the optimal algorithm design A_D for the problem instance. The 2^k factorial design is also implemented to observe the effect of each algorithm design on the problem instance, so that this gives more insight into the problem and algorithm. Since these algorithms are stochastic, they were run three times for each experiment, and the average, best, and worst performance was observed.

3.5.3 Algorithm-Specific Parameters

The main difference in the Teaching Learning Based Optimization (TLBO), when compared to the other algorithms, lies in the TLBO being ‘parameter-less’. The TLBO algorithm does not contain algorithm-specific parameters, but only contains common parameters.

The main difference in the Flower Pollination Algorithm (FPA), when compared to the other algorithms, lies in the switch probability of the algorithm being what controls whether the algorithm performs exploration or exploitation at each iteration. The TLBO algorithm does not have anything that controls this and performs exploration and exploitation in one iteration.

This is also different from the Genetic Algorithm (GA), which has more than one parameter that controls this. Therefore, the algorithms progress from the GA to the TLBO with the FPA in between, with regards to how much control the practitioner has on it performing exploration or exploitation. The value of choosing to compare these algorithms lies in the ability to see how increasing the practitioner's control on implementing exploration or exploitation affects the performance.

3.5.4 Problem Instances used in the Experiments

The function of the control system was to stabilize the LOS rate of the gimbal to the commanded input, which represents the target movement rate. As seen in Figure 2.1, in Section 2.7 of Chapter 2, there are three problem instances (or scenarios) of the application which the algorithms are tested on.

The first scenario is by inputting a step input function into the gimbal control system without any added additional non-linearity. The only non-linearity which exists in this system is due to the rate gyro modelling, which is described in Section 3.1. This input step function represents a constant velocity of the target to which the gimbal must follow. This means that in this scenario, the target is moving and the rate of movement remains constant at 1 rad/s.

Instead of the step input function, the second scenario consists of a ramp input function as a representation of the target motion. The ramp input function is a dynamic input which represents an increasing target velocity but with a constant acceleration of 1 rad/s². There is still no additional non-linearity, as this experiment only observes how the algorithms and algorithm designs perform with a different input.

The last scenario is where the control system includes additional non-linearity in the control system. This non-linearity is described in Section 3.1.5 and represents

the friction acting on the rotational elements of the gimbal, and the torque disturbance due to base motion and mass unbalance. This scenario tested how the algorithms perform with a step input and added non-linearity.

3.5.5 Research Environments

The operating system that used for this study is the Microsoft Windows 10 64-bit operating system with 8GB RAM and i7 core processor. The research environment chosen for this study was MATLAB/Simulink. The research environment chosen is due to various studies which involved tuning a controller using nature-inspired heuristic algorithms having utilised the MATLAB/Simulink environment.

3.5.6 Work Breakdown Structure

The experiments conducted compared the different elements of this study to observe system response. Figure 3.11 shows the work break down structure (WBS), which describes the tasks required for this study when implementing the experiments and comparisons.

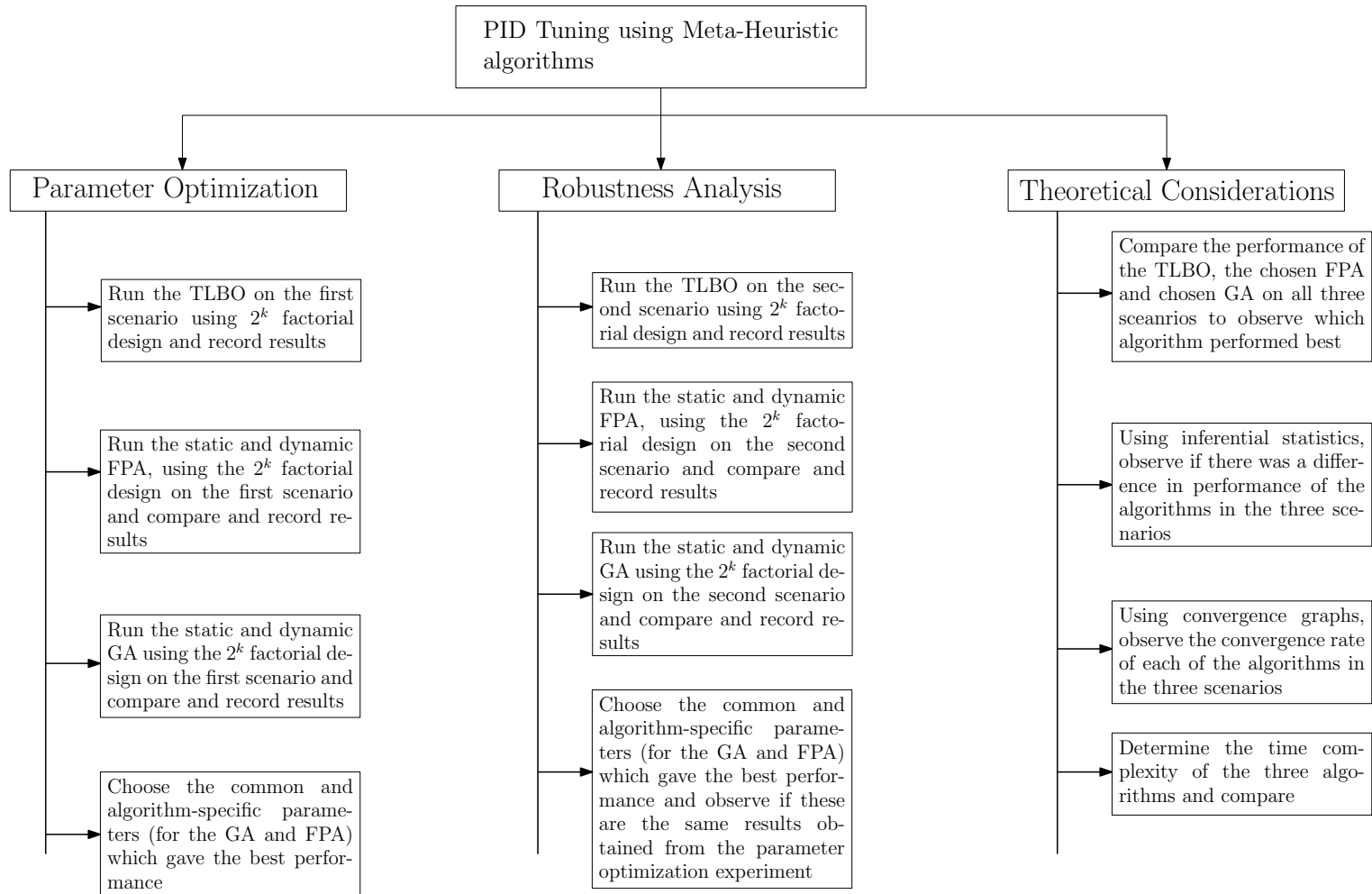


FIGURE 3.11: Work breakdown structure describing tasks required for this study

CHAPTER 4

Parameter Optimization for Nature-Inspired Meta-Heuristic Algorithms

Freedom is not the absence of obligation or
restraint, but the freedom of movement within
healthy, chosen parameters.

Kristin Armstrong

This chapter addresses how the common and algorithm-specific parameters of the three algorithms affect the performance of this problem instance. This problem instance is described as a constant target velocity modelled as a step input, with the LOS rate of the gimbal attempting to follow the commanded input.

The chapter begins by describing the research questions and hypotheses relevant to this experiment. It then continues to describe the results obtained for each algorithm and discussing these results. The chapter then ends with some concluding remarks, which connects the research questions to the results obtained.

4.1 Research Questions

For this particular case study, the relevant research questions which the experiments attempted to answer are as follows:

1. Do common algorithm parameters, specifically the population size and search space bounds, affect the algorithm performance for this problem instance?

2. Should algorithm-specific parameters be adaptive (also referred to as dynamic) or static for this problem instance and do these parameters interact with the common algorithm parameters?

4.2 Parameterless Teaching Learning Based Optimization Algorithm Results

Table 4.1 shows the fitness value descriptive statistical results of the TLBO for this problem instance, where A_D represents the different algorithm designs shown in Table 3.7 in Chapter 3, Section 3.5.2.

TABLE 4.1: Descriptive statistical results illustrating the changes in the fitness value for the different algorithm designs

A_D	Best	Mean	Max	Std Dev	% increase from best to mean	% increase from best to max
1	34.6681	34.6725	34.6807	0.0176	0.0128	0.0363
2	34.5966	34.5981	34.6007	0.0191	0.00424	0.0118
3	34.418	34.4190	34.4201	0.00420	0.00281	0.00610
4	34.4131	34.4152	34.4181	0.00474	0.00610	0.0145

Table 4.1 shows the percentage increase of the fitness value from the best (which is the minimum fitness value) to the mean, and from the best to the maximum. This is to illustrate how much higher the average and maximum fitness values are compared to the minimum value to show the range of the fitness value results.

The highest percentage increase from the best to the maximum is produced by algorithm design one with 0.036%. This means that no solution has given a fitness value above 0.036% higher than the lowest fitness value. The TLBO is a stochastic algorithm which means each run produces a different answer.

However, since the maximum range in the results is less than 1%, this indicates two things: (1) either these solutions are from the same region which the algorithm has the ability to identify at each run or the problem is multi-modal because different solutions produce similar fitness value results, and (2) the lower the range in fitness

value, the higher the probability that this algorithm produces the optimal or near-optimal solution at each run. To determine which one of (1) is true, one must observe the PIDN solutions obtained from the algorithm shown in Table 4.2.

Table 4.2 also shows the time-domain results produced by the solution gains for each algorithm design, with t_R , t_S , M_p , and t_c representing the rise time, settling time, percentage overshoot, and computational time respectively. The PIDN controller gains represent the average controller gains obtained from the three runs implemented.

TABLE 4.2: Time-domain results for the TLBO for the first problem instance

A_D	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p (%)	t_c (s)
1	34.6725	94.740	46.199	0.1883	13.58	0.026	0.0397	0.0393	882.79
2	34.5981	99.896	49.720	82.35	0.2096	0.0252	0.035	0.0329	3273.93
3	34.4190	242.325	48.304	0.03903	20.93	0.0250	0.0403	2.4547	885.21
4	34.4152	294.563	47.709	0.1649	90.25	0.0250	0.0313	1.981	2684.20

Table 4.2 shows that the algorithm designs which produced the lowest rise time are algorithm designs 3 and 4 with 0.025 seconds. These algorithm designs, though, resulted in a high percentage overshoot relative to the other algorithm designs. The solutions of these algorithm designs show higher proportional gains in comparison to algorithm design one and two. This shows that a higher proportional controller gain results in a slower rise time, but runs the risk of experiencing a higher overshoot.

The solutions that are shown in Table 4.2 also indicate that the problem is multi-modal since the changes in the controller gains had little effect to the fitness value, with the range in fitness values being less than 1% as shown in Table 4.1.

It is worthy to note that the results shown in Table 4.2 surpass the standard time-domain results for a gimbal stabilization system stipulated in [137] and shown in Chapter 3 Section 3.3.

Figure 4.1 compares the changes in the PIDN gain values to the changes in the changes in the fitness values for each algorithm design. This figure shows how the

changes in the algorithm design change the PIDN solutions and fitness value, and the effect of this on the time domain responses.

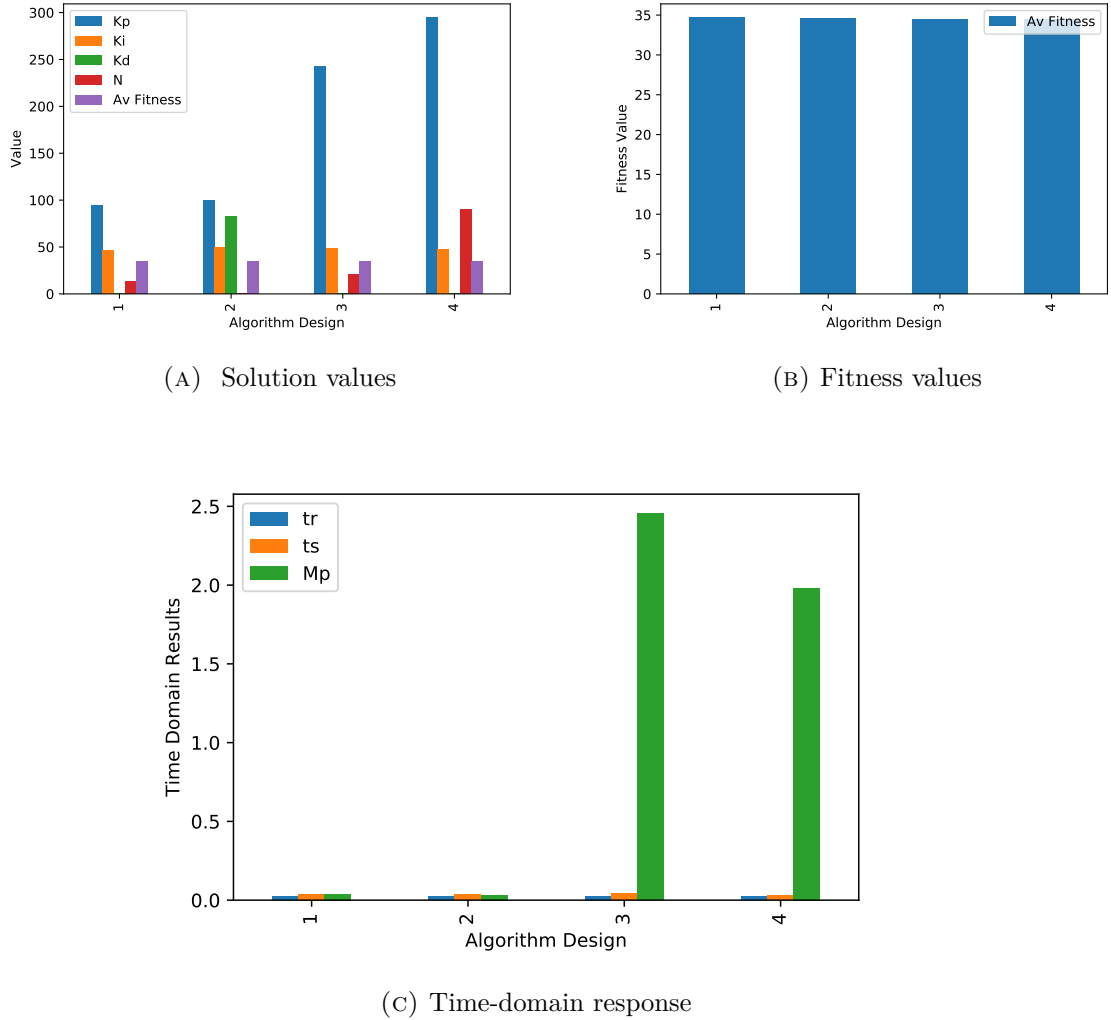


FIGURE 4.1: Bar graph showing the solution and fitness values for each algorithm design for the TLBO

Figure 4.1a confirms further that the changes in the controller gains, particularly the K_p and K_d values, contribute to relatively small changes in the average fitness, with the changes in the average fitness shown more closely in Figure 4.1b.

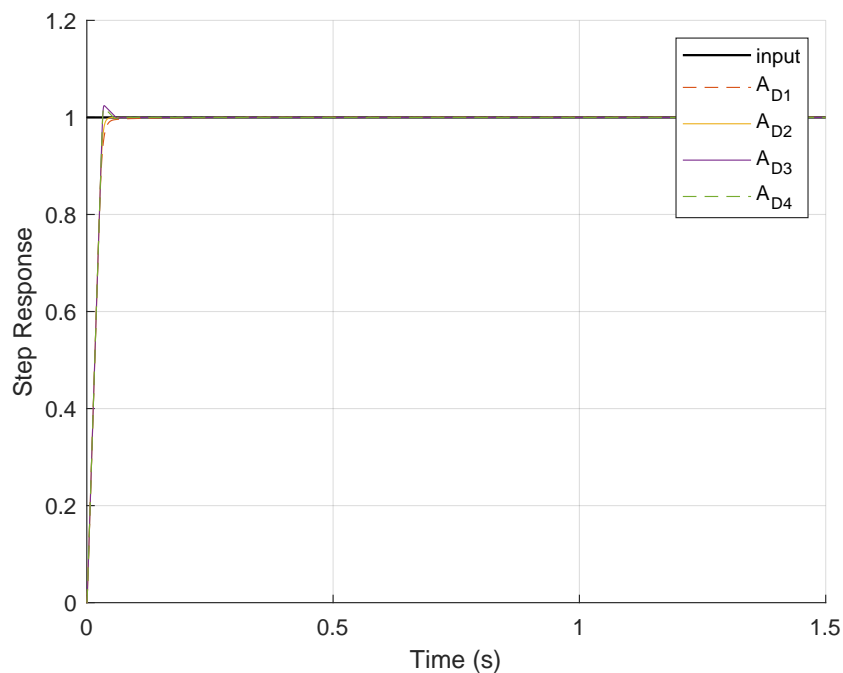
The increase in population size from algorithm design one to two led to an increase in the PID controller gains as seen in Figure 4.1a, and a decrease in the filter coefficient N . This has led to a decrease in the fitness value as seen in Figure 4.1b.

Decreasing the population size and increasing the search space in algorithm design one resulted in the proportional controller gain increasing significantly, along with the filter coefficient while the derivative gain decreases. Algorithm design three has also lead to a significant increase in the percentage overshoot, with a significant decrease in the fitness value. This means an increase in the diversity of solutions since there are fewer population members with a broader range of solutions.

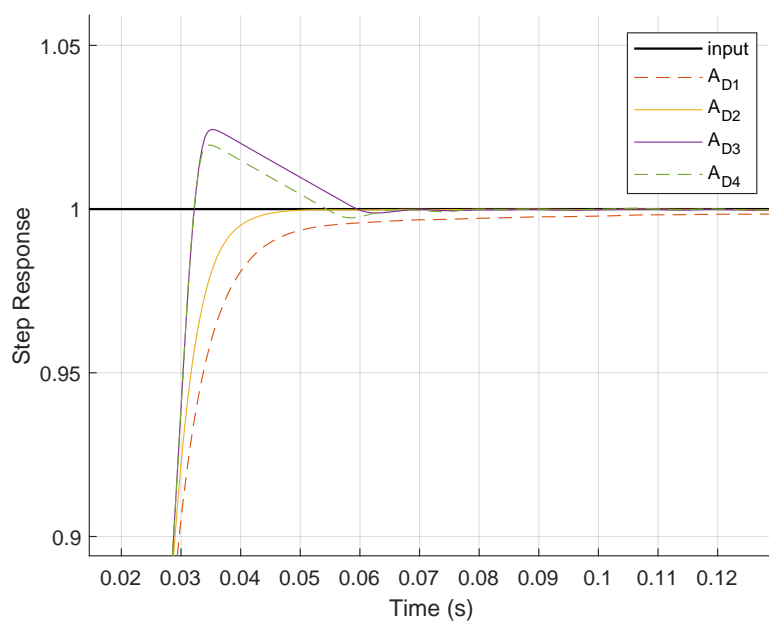
Increasing the population size from algorithm design three to four has lead to a slight decrease in fitness value and percentage overshoot.

The difference between algorithm design two and three is the search space which resulted in a high percentage overshoot. Increasing the population size from design three to four decreased the overshoot percentage. This indicates that a higher search space requires a higher population size and computational resources to find the optimal algorithm design.

Figure 4.2 shows the step response of the different algorithm designs.



(A) Steady-state response



(B) Zoomed-in view of Figure 4.2a showing the transient response

FIGURE 4.2: The step response of the algorithm designs for the TLBO

Figure 4.2a indicates that all the algorithm designs follow the commanded input within the required time of 1.5 seconds. However, 4.2b illustrates the high M_p given by algorithm design three and four, as mentioned previously. The most favourable step response behaviour is algorithm design two, as seen in Figure 4.2b. Table 4.2 shows that this design resulted in the lowest settling time and percentage overshoot while maintaining a reasonably low rise time.

The process of choosing which algorithm design is best suited requires the results to be analysed collectively for the decision to be made. This is because there will not be a clear outstanding result, but a good compromise of the different performance requirements can be obtained. For example, a short rise time usually comes with a higher overshoot, and a small overshoot is accompanied by a longer settling time. Thus it imperative to collectively observe the results.

The fitness function described in Section 3.3.2 places emphasis on reducing the ISE value with its weight being 60%, and the ITAE weight being 40%. The ISE value is given a higher weight because of its ability to reduce rise time because it is important for this application to show a quick response. This is why a solution like the one produced by algorithm design four, though has a high percentage overshoot, is considered a good quality solution because it contains the lowest rise time and gives the lowest fitness value.

The increase in population size from algorithm design three to four resulted in a decrease in percentage overshoot. However, the decrease in this overshoot is not comparable to the low overshoot given by algorithm design one and two. This indicates that for the higher search space bound in the algorithm designs three and four, a suitable solution could be found with a higher population size. This means that a suitable solution requires a larger number of fitness function evaluations and computational resources.

Algorithm design two displays the most favourable response in comparison to the other algorithm designs. The difference between algorithm design one and two is

the higher population size which means higher fitness function evaluations. Algorithm design two displays an improvement of results from algorithm one; however, one must reflect on whether this is worth the increase in the computational cost.

Given the limited resources that this study has, choosing the appropriate design for this context is essential. Though algorithm design four resulted in the lowest fitness value, algorithm design two (which was the algorithm design that had a population number of 60 and a search space of 100) gives the most favourable response and a good compromise for the opposing requirements. Thus this design was chosen as the best performing.

4.3 Dynamic and Static Flower Pollination Algorithm Results

TABLE 4.3: Descriptive statistical results for the FPA for the first problem instance

Algorithm Type	A_D	Best	Mean	Max	Std dev	% Av over best sln	% Max over best sln
Static FPA	1	34.7419	35.3164	35.7037	0.4144	1.654	2.768
	2	34.7207	34.7680	34.8546	0.06130	0.1363	0.3856
	3	34.7934	35.6921	37.013	0.9541	2.583	6.379
	4	34.4621	34.4678	34.4713	0.004084	0.01663	0.02670
Dynamic FPA	1	34.9104	35.0569	35.1935	0.1158	0.4196	0.8109
	2	34.7202	34.7257	34.729	0.003915	0.01584	0.02534
	3	34.4853	34.5036	34.5137	0.01294	0.05297	0.08235
	4	34.4365	34.4446	34.4575	0.009238	0.02342	0.0610

Table 4.3 shows that the algorithm design displaying the lowest fitness value (shown in bold) is algorithm design four for both the static and dynamic FPA. The lowest mean fitness value from the algorithm variations is the dynamic FPA with a fitness value of 34.4446.

Table 4.3 also shows that the average and maximum percentage over the best solution for the static FPA is higher than that of the dynamic FPA, for all the algorithm designs besides algorithm design four. This means that the static FPA, for the most part, produces a larger solution diversity in comparison to the dynamic FPA. This shows the effect of the switch probability since the static FPA keeps this probability at 0.8 (80%), which gives global pollination an advantage and allows for

diversity. In contrast, the dynamic FPA changes this switch probability to increase the likelihood of local pollination to occur and the fine-tuning of solutions in later iterations.

Since the dynamic FPA results in the lowest fitness value, this means that the problem prefers the dynamic FPA and benefits from fine-tuning and exploitation in later iterations rather than a constant higher probability of global pollination throughout all iterations. This is aligned with the conclusions obtained by Ozsoydan, and Baykasoglu [77] who when testing different methods of implementing the switch probability on well-known unconstrained function minimisation problems, found that they should perform exploitation at later iterations. Because this problem benefits from this, this means that small changes in the solutions within the same region can increase performance rather than changing the region entirely.

Table 4.4 shows that the PI controller with an N coefficient is the solution both the static and dynamic FPA algorithm produced for all the algorithm designs, except for algorithm design three. Since the derivative controller is zero for the algorithm designs besides algorithm design three, the N coefficient for those particular designs makes no difference in the performance. The PI controller solution means that the system does not need additional damping and that it is sufficiently damped because the purpose of the D controller is to reduce overshoot and oscillations, i.e. increasing damping.

TABLE 4.4: Time-domain results for the FPA for the first problem instance

	A_D	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	$M_p\%$	t_c (s)
Static FPA	1	35.3164	92.709	45.725	0	17.142	0.0261	0.0393	0.0327	2723.10
	2	34.7680	98.551	46.868	0	53.923	0.0258	0.038	0.0375	3361.03
	3	35.6921	280.80	38.765	32.170	41.389	0.172	0.424	0.2308	1144.82
	4	34.4678	269.84	48.041	0	67.333	0.0250	0.0454	2.9912	2950.97
Dynamic FPA	1	35.0569	94.031	44.599	0	58.756	0.0260	0.0390	0.0278	2208.62
	2	34.7257	99.210	46.390	0	50.734	0.0257	0.0378	0.0355	3316.39
	3	34.5036	463.491	48.279	0.0250	0.0619	0.0250	0.0619	4.7349	2092.01
	4	34.4446	322.725	49.776	0	69.509	0.0250	0.0521	3.6999	7124.08

For algorithm design three, there is a derivative controller for both the static and dynamic FPA, and thus this design gives a PIDN solution. Algorithm design three consists of the larger search space limit with the smaller population size.

Because of this, the solutions are more likely to display a higher diversity. This is why algorithm design three resulted in a derivative controller. When the design changed from three to four, i.e. when the population size increased, this resulted in a solution without the derivative controller.

How the changes in the PIDN controller gains affect the fitness value is shown in Figure 4.3, where the prefix D and S represent the dynamic and static FPA results, respectively.

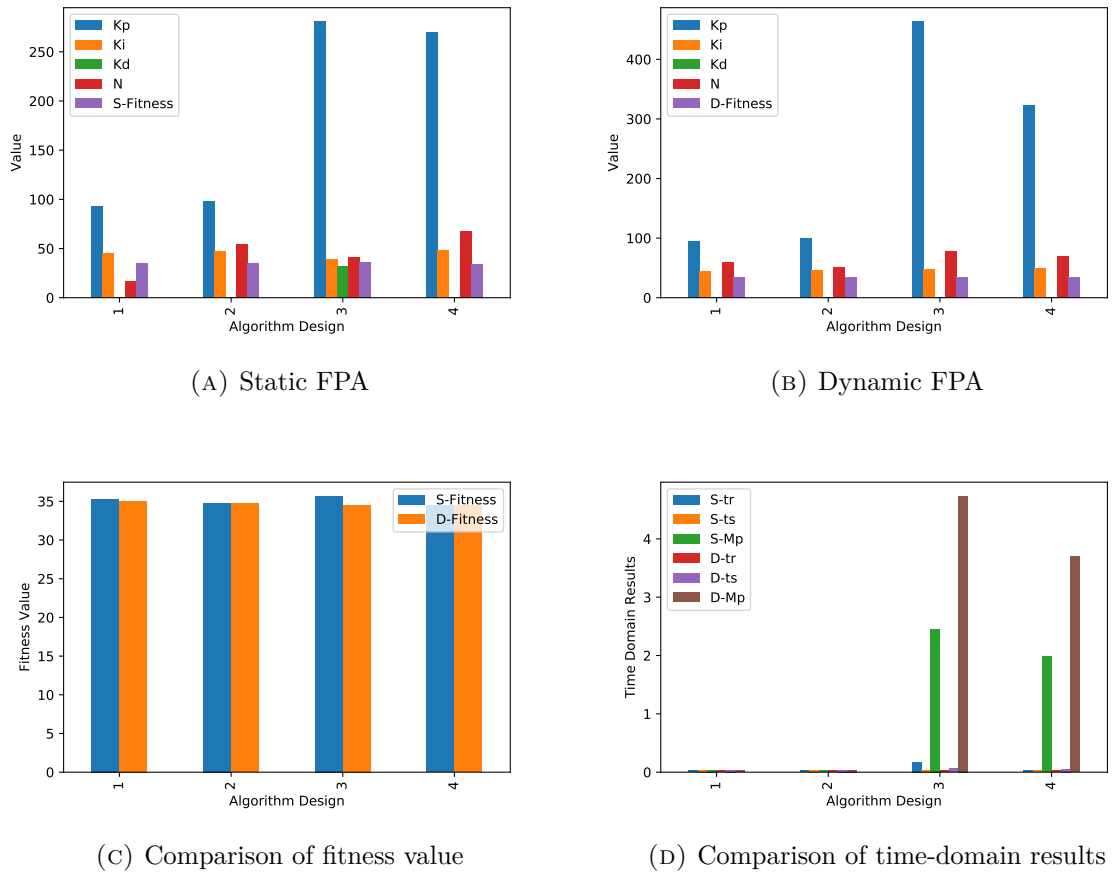
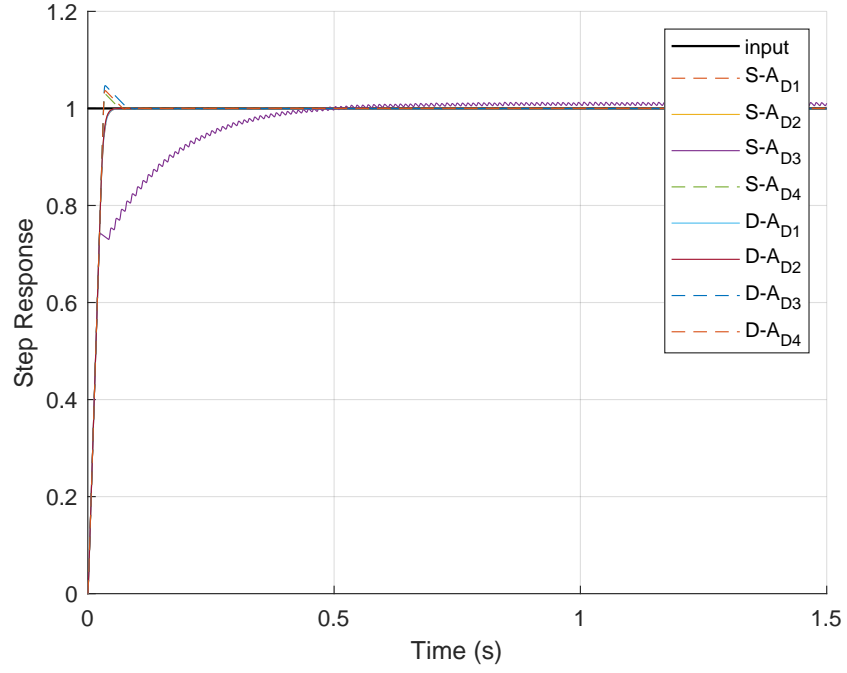


FIGURE 4.3: Bar graph showing the solution and fitness values for each algorithm design for the static and dynamic FPA

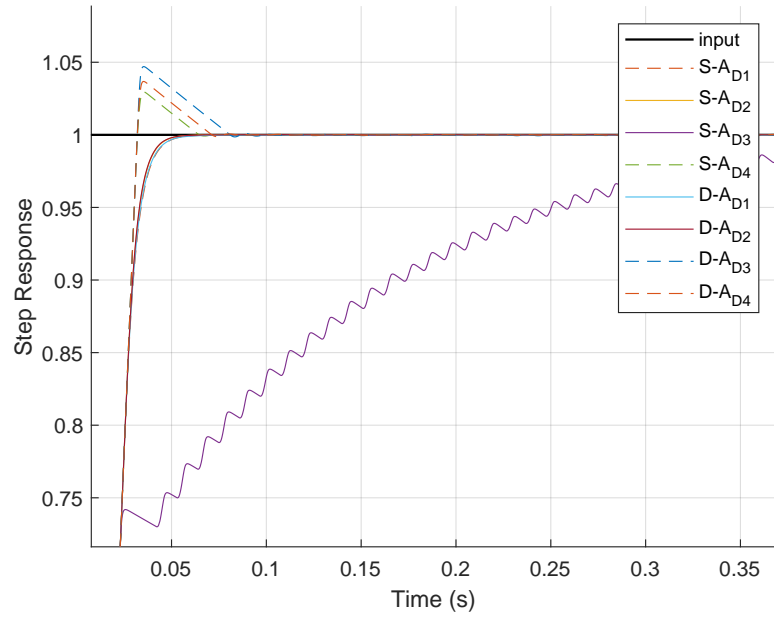
The changes in the proportional gain are similar to the changes observed in the TLBO results, in that the proportional gain greatly increases when the search space bound is increased in algorithm design three, for the dynamic and static

FPA. This results in a lower rise time fitness value with the expense of a larger overshoot.

What is different from the TLBO is the increase in fitness value for the static FPA from algorithm design two to three. This is because of the derivative controller, which was part of the solution produced by algorithm design three, as seen in Table 4.4.



(A) Steady-state Response



(B) Zoomed in view of 4.4a showing the transient response

FIGURE 4.4: The step response of the algorithm designs for the static and dynamic FPA

The step response shown in Figure 4.4 illustrates the high percentage overshoot shown by the algorithm designs four for the static FPA and design three and four for the dynamic FPA.

Algorithm design three for the static FPA shows a different behavioural step response in comparison to the other step response behaviours. The difference in the solution for this design is the relatively large derivative controller value coupled with a relatively large filter coefficient. A large derivative allows for noise to enter the system, hence the need for a filter coefficient. However, a large filter coefficient means a large cut-off frequency as the filter coefficient informs the cut-off frequency. This means that a large cut-off frequency allows for more noise to enter the system which is shown in this response. The PIDN gain solutions obtained from design three of the static FPA show a response with noise throughout the 1.5 seconds in Figure 4.4. This may be due to the higher cut-off frequency value, which allows for high-frequency noise.

When observing Figure 4.4a, algorithm design one and two gave favourable and similar step response behaviour, for both the static and dynamic FPA. The effect of population size is seen when comparing algorithm design one to two, and algorithm design three to four. Table 4.4 shows that algorithm design two resulted in the lowest fitness value for both the static and dynamic FPA, and so did algorithm design four. This confirms that at any search space limit, the algorithm design with the higher population improves the performance.

The lowest mean fitness value when comparing the dynamic and static FPA, and observing the algorithm design two is the dynamic FPA. The similarity between the TLBO and the dynamic FPA is that they encourage exploitation as much they do exploration, whereas the static FPA focuses more on the exploration.

The difference between the dynamic FPA and the TLBO is that the dynamic FPA increases the probability of exploitation occurring in later iterations. In contrast, the TLBO does not operate with probability but rather guarantees exploitation in each iteration with the learner phase of the algorithm.

For algorithm design two, the dynamic FPA resulted in the lowest fitness value. Thus, design two was chosen as the best performing, and this design, along with a dynamic switch probability, was used in the algorithm comparison in Chapter 6.

4.4 Dynamic and Static Genetic Algorithm Results

Table 3.5 in Chapter 3, Section 3.2.1 illustrates the differences in the dynamic and static GA. This table shows that along with comparing the quantitative parameters, such as the mutation rate and crossover probability, this experiment also compares a qualitative parameter of the GA being the selection type. The static GA is one that has a 100% probability of crossover, which means that this GA favours exploitation rather than exploration. Also, the dynamic GA maintains a certain level of diversity, which means that this GA favours exploration.

Algorithm design four gives the lowest fitness value for both the static and dynamic GA, as seen in Table 4.5, with the static GA giving the lowest fitness value. The highest maximum over best solution percentage is given by algorithm design one for both the static and dynamic GA, with the highest being the dynamic GA at 1.65%. This is expected as the dynamic GA allows for maintenance of high diversity individuals, and thus would be the algorithm with the higher range in performance.

The corresponding solutions and time domain responses for the mean fitness values shown in Table 4.6 further illustrate that the lowest rise times, which were obtained by algorithm design four for the static GA and algorithm design three and four for the dynamic GA, are coupled with a relatively large percentage overshoot. This reflects how the fitness function was weighted.

The difference in the GA solutions compared to the FPA solutions discussed in Section 4.3, is that the GA solutions contain the full PIDN controllers and not only the PI controller. This indicates that this application could be multi-modal and the GA has been able to locate other regions within the search space, which

TABLE 4.5: Descriptive statistical results for the GA for the first problem instance

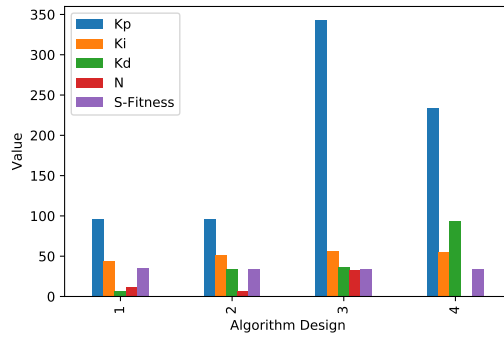
	A_d	Best	Mean	Max	Std Dev	% Av over best sln	% Max over best sln
Static GA	1	34.6685	34.8308	35.108	0.1969	0.4682	1.268
	2	34.6238	34.6730	34.7143	0.03736	0.1420	0.2614
	3	34.4337	34.4861	34.552	0.04922	0.1523	0.3436
	4	34.4213	34.4354	34.4635	0.01989	0.04087	0.1226
Dynamic GA	1	34.7719	34.9649	35.346	0.2695	0.5550	1.651
	2	34.7046	34.7191	34.7404	0.01537	0.04188	0.1032
	3	34.4498	34.6076	34.7876	0.1388	0.4580	0.9806
	4	34.4284	34.5078	34.6351	0.09095	0.2305	0.6004

TABLE 4.6: Time-domain results for the GA for the first problem instance

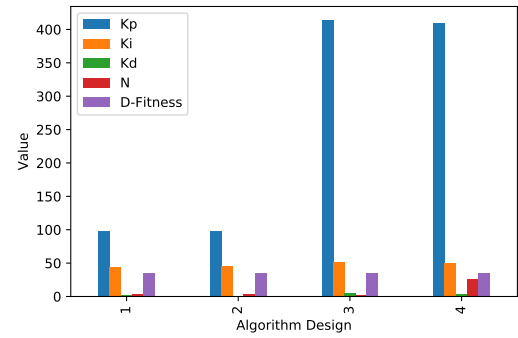
	A_d	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p (%)	t_c (s)
Static GA	1	34.8308	96.194	44.240	6.428	12.316	0.0257	0.239	0.0987	932.11
	2	34.6730	95.787	51.150	33.586	7.382	0.0252	0.680	0.0222	2765.62
	3	34.4861	342.763	56.735	36.674	32.442	0.147	0.401	0.200	1117.40
	4	34.4354	234.309	54.873	94.169	0.263	0.0250	0.0436	2.799	2827.08
Dynamic GA	1	34.9649	97.624	43.775	2.180	3.716	0.0256	0.0379	0.0267	954.69
	2	34.7191	98.634	46.065	0.67976	3.4715	0.0257	0.0379	0.0269	3455.12
	3	34.6076	413.701	50.9061	5.5924	2.4306	0.0250	0.0597	4.5017	977.71
	4	34.5078	409.622	50.102	3.8476	25.643	0.0250	0.0757	0.0983	7709.43

can produce quality results which are different from the solutions produced by the FPA.

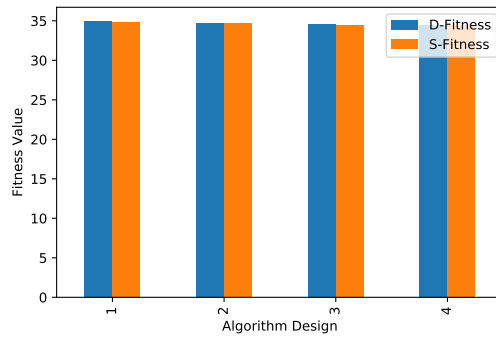
Figure 4.5a and 4.5b indicate the rise in the proportional gain controller from algorithm design 2 to algorithm design three, as seen in the TLBO and FPA results. Figure 4.5c shows the decrease in fitness value, which has been seen in both the dynamic and static GA.



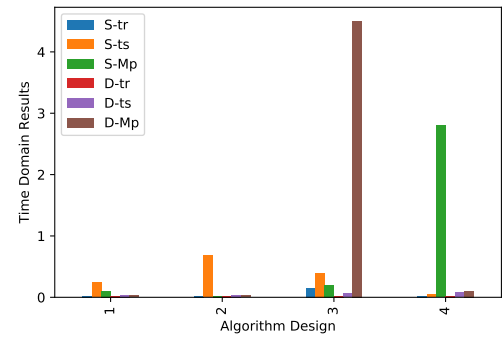
(A) Static GA



(B) Dynamic GA

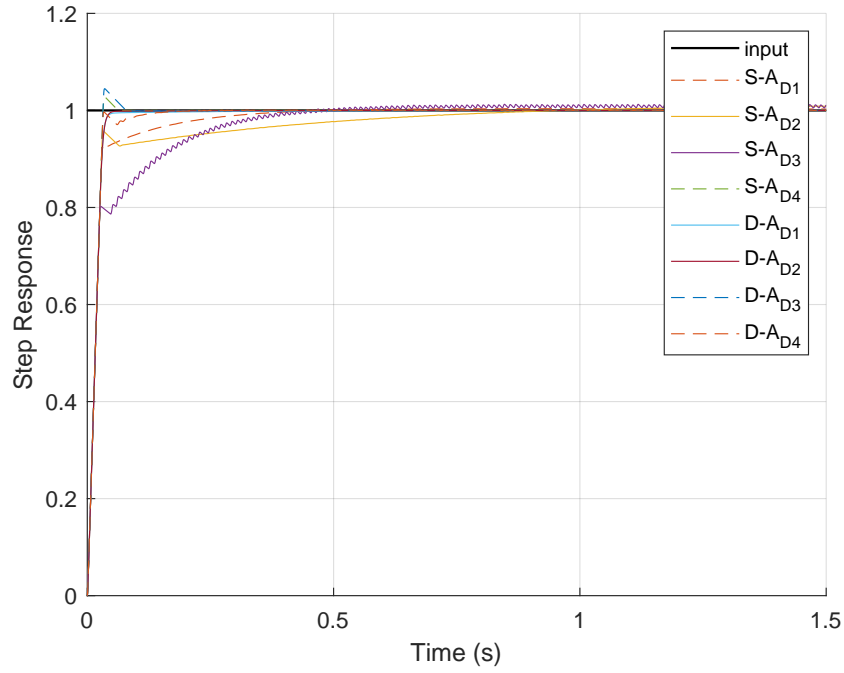


(C) Change of fitness value

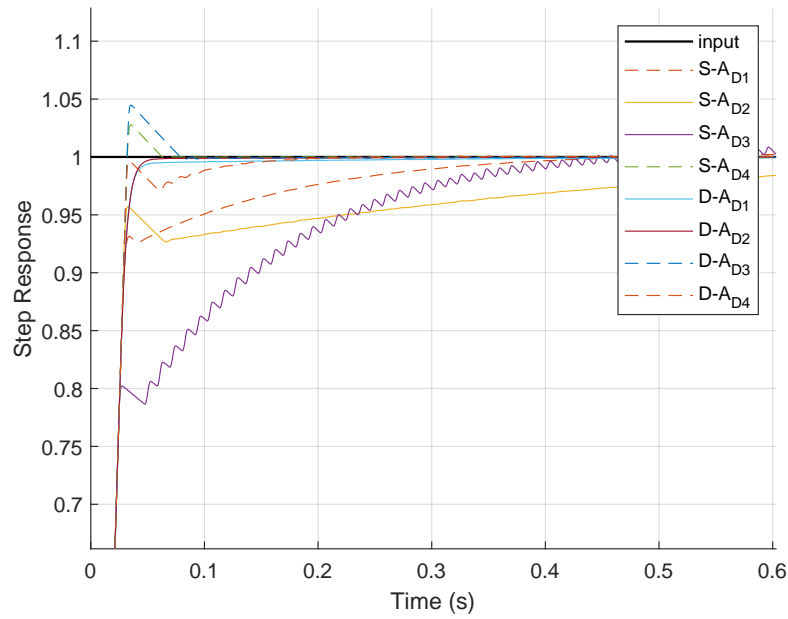


(D) Change of time domain results

FIGURE 4.5: Bar graph showing the solution and fitness values for each algorithm design for the static and dynamic GA



(A) Steady-state response



(B) Transient response

FIGURE 4.6: The step response of the algorithm designs for the static and dynamic GA

The most favourable response is algorithm design one and two of the dynamic GA shown in Figure 4.6. These solutions contained relatively low derivative controllers, which effectively is a PI controller, as seen in Table 4.6. A larger overshoot is seen by algorithm design three for the dynamic GA and algorithm design four for the static GA. The static GA response shows a similar response to the static FPA when both algorithms were using algorithm design three parameters. The PID solutions to these responses are also very similar, with the filter co-efficient and derivative controller gain both being large in value simultaneously. This confirms that this behaviour is because of the noise that the larger derivative gain and large filter coefficient allow. Other algorithm designs resulted in a behaviour indicating a longer settling time, as seen in Figure 4.6. Table 4.6 also shows that even though the proportional gain can be high, increasing the derivative controller decreases potential overshoot, but with the expense of increasing settling time.

This is why the static GA, though gives the lowest fitness value, was not chosen to be used as it displays the long settling time. The dynamic GA, with the motive of constantly maintaining diversity in the algorithm, obtains favourable behaviour by making use of algorithm design one and two, with algorithm design two being most favourable and thus this algorithm design was chosen.

4.5 Concluding Remarks

This section describes the conclusions obtained and suggestions derived from the results discussed above.

4.5.1 Conclusions

The following conclusions are made:

- In this instance, since the fitness function favoured a shorter rise time, solutions with the shortest rise time, though contained large overshoots and longer settling times, were considered to be good by the fitness function since this resulted in the lowest fitness value. However, these solutions did not produce favourable step responses due to their overshoot and settling time. The algorithm designs with the most favourable step responses and the best compromise of the different objectives were chosen.
- There is an important relationship between the search space limit and the population size. A larger search space upper bound can be beneficial if there is a larger population to explore the search space fully. A large search space is not beneficial if there are not enough population members to explore it. Thus, if there are limited computational resources (i.e. population size), instead choose the limited search space upper bound which still suits the application.
- When observing the fitness value as a form of performance measure, the dynamic FPA, which encourages exploitation, gave the lowest fitness value in this problem when compared to the static FPA. The static GA, which also allows for exploitation, gave the lowest fitness value in comparison to the dynamic GA. This means that this problem, with this fitness function, preferred algorithms which implement a higher probability of intensification as well, rather than those that favoured diversification. The importance of

fine-tuning solutions rather than constantly looking for other solutions in different search space regions is shown here. This also indicates that the problem is not multi-modal, and thus the algorithm must be able to focus on a specific region of the search space.

- Algorithm design two resulted in the most suitable step response with a reasonable compromise between the requirements for all the algorithms. This means that when observing all the requirements (i.e. fitness function value and favourable response), algorithm design two resulted in the best performing. This means that the algorithms applied in this problem instance performed best with a higher population size of 60 but limited search space bound of 100.

4.5.2 Suggestions for Future Practitioners

From these conclusions, some considerations or suggestions can be drawn for future meta-heuristic algorithm practitioners when tackling this kind of problem with these kinds of algorithms:

1. If the application requires a fast rise time (this translates to an increase in the weight of the objective which applies to rise time in the fitness function), ensure that the maximum number of population members are used while limiting the search space. Limiting the search space creates a constraint for settling time and overshoot, while the higher population members ensure the search space is fully explored to find a solution which gives the fastest rise time.
2. Ensure that the algorithm-specific parameters chosen give a higher probability for exploitation in later iterations as this problem favoured that.
3. The static GA gave the lowest fitness value but was not chosen as the best performing because of the settling time. The settling time was present because of the high derivative controller value. Since the best solutions for all

the algorithms resulted in high PI controller gains. But with low derivative controller gains or filter coefficients, it would be good to reduce the search space upper bound of the derivative controller to something less than that of the other controller gains (i.e. less than 100) so that this may increase the number of viable solutions in the search space.

CHAPTER 5

Robustness Investigation of Nature-Inspired Meta-Heuristic Algorithms

This chapter determines which problem instances affect the common and algorithm-specific parameters of the algorithm, and thus the system performance. The robustness analysis refers to whether the algorithms can perform well with changing problem instances, and if there is a need to change the algorithm parameters to suit the problem.

The chapter begins by stating the research questions which are relevant for these experiments. It continues to describe and discuss the results obtained from the three algorithms on the two different problem instances. The chapter then ends with some concluding remarks.

5.1 Research Questions

The relevant research questions for this study are as follows:

1. Can the algorithms perform when presented with a different input and added non-linearity in the problem?
2. Do the common and algorithm-specific parameters affect the performance of the algorithms in these problem instances?

5.2 Dynamic Input

This problem instance is described as the target of interest moving at a constant acceleration, thus an increasing velocity. This movement is modelled as a ramp input where the gimbal LOS rate tries to follow the commanded input. The three algorithms were tested on this problem instance, with changing the algorithm designs from one to four.

5.2.1 Parameterless Teaching Learning Based Optimization Results

Table 5.1 shows that algorithm design four obtained the lowest fitness value.

TABLE 5.1: Descriptive statistical results for the TLBO for the second problem instance

A_d	Best	Mean	Max	Std Dev	% increase from av to max	% increase from best to max
1	5.8445	5.8688	5.8858	0.01761	0.4152	0.7066
2	5.8001	5.8217	5.8466	0.01913	0.3718	0.8017
3	1.7782	1.7817	1.7876	0.004196	0.1968	0.5286
4	1.7603	1.7662	1.7719	0.004738	0.3352	0.6590

The table also shows that there is also not a vast range of solutions, with the standard deviation not exceeding 0.02 and the maximum percentage over the best solution not exceeding 0.9%.

Table 5.2 shows the average solutions obtained from each algorithm design, and the computational time used to obtain these results. Since this is a ramp input design, time-domain results cannot be utilised to assess performance as these are only applicable to step inputs.

The results shown in Table 5.2 are different from the results observed in Table 4.2 with regards to the PIDN controller gains produced. Table 5.2 shows that the TLBO, in this problem instance, produced a PIDN controller with the larger integral controller solutions producing smaller fitness values.

TABLE 5.2: PIDN solution results for the TLBO for the second problem instance

A_D	Mean Fitness	K_p	K_i	K_d	N	$t_c(s)$
1	5.8688	55.590	55.485	13.416	98.672	1214.84
2	5.8217	55.253	56.989	13.282	99.816	6313.58
3	1.7817	20.638	500	1.5931	15.883	984.45
4	1.7662	16.546	499.73	3.7085	58.287	2939.56

Table 5.2 also shows that the increase in population size from algorithm design 1 and 2 resulted in a high increase in the computational time without a significant decrease in the fitness value. However, when changing the search space bound to the higher value from algorithm design 2 to 3, this caused a significant decrease in the fitness value. This is due to the high increase in the integral controller gain with a lowered proportional and derivative gain, respectively.

Figure 5.1 confirms that the large integral gain from algorithm design 2 to 3 has significantly dropped the fitness value.

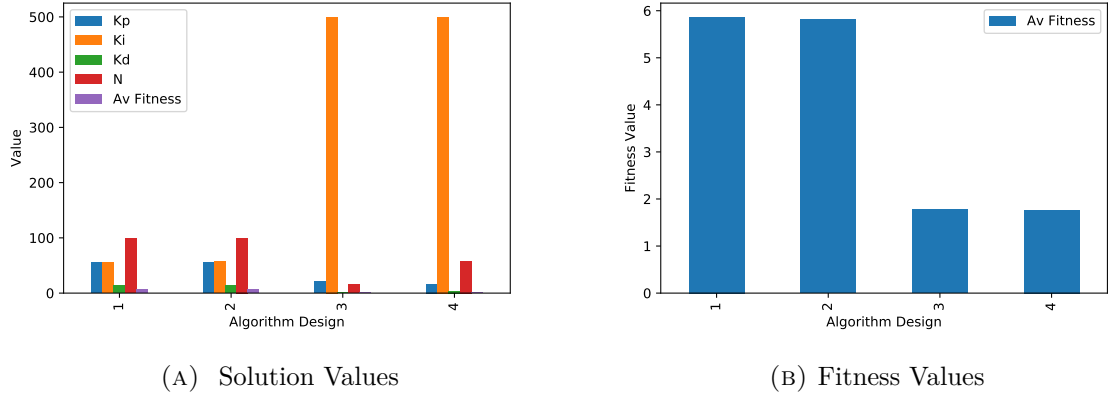


FIGURE 5.1: Bar graph showing the solution and fitness values for each algorithm design for TLBO on the control system on the second problem instance

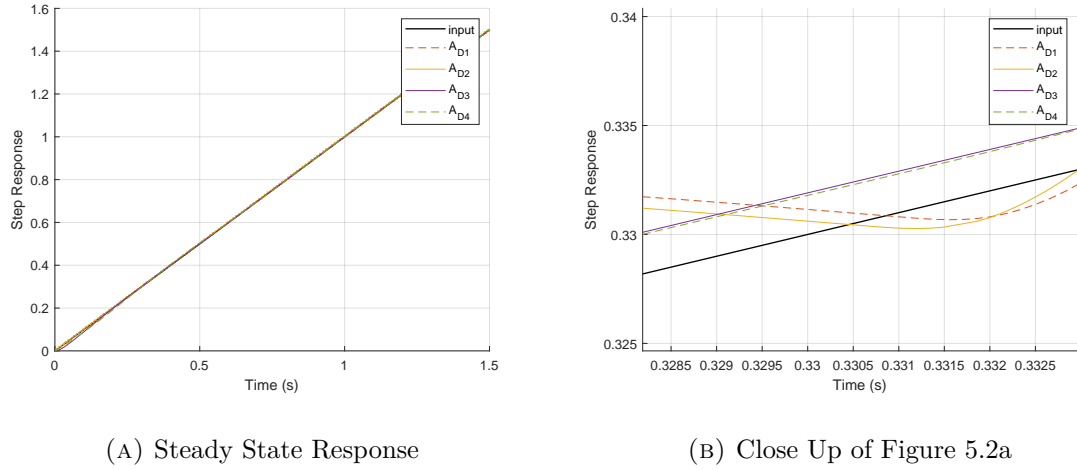


FIGURE 5.2: The ramp response of the algorithm designs for the TLBO

This integral gain has improved ramp performance behaviour, as seen in Figure 5.2. Figure 5.2b shows that algorithm design one and two resulted in an oscillation around the input, whereas algorithm design three and four were able to track commanded input without the oscillation. This gives reason as to why the fitness value for algorithm design three and four has decreased in comparison to algorithm design one and two.

The proportional controller functions by observing the error term between the commanded input and output and attempts to reduce this error proportionally. Since this is a ramp input and not a step input, the error difference at the beginning of the simulation is zero as both input and output begin at zero. This is why there is no requirement for a high proportional gain. The integral controller is used to sum the error term over time to reduce the steady-state error. Algorithm design one and two followed the ramp input, but oscillating around it and causing a high steady-state error. The increase in the integral gain reduced this steady-state error which resulted in a decreased fitness value.

However, the question is whether the decrease in the fitness value was a result of solely increasing the integral gain, or the combination of increasing the integral gain and decreasing the proportional, derivative, and filter coefficient value. In other words, would the performance be similar if the ratio between the PIDN

values for algorithm design one and two were the same as the ratio between these values for algorithm design three and four, even though the actual values are not the same? Obtaining similar ratio's between these algorithm designs would mean increasing the integral gain and decreasing the other controller gains.

Upon investigation, it was found that the ratio's between the controller gains are important, and there are solutions which could have decreased the oscillations in the performance within the search space bounds of 100, i.e. using algorithm design one or two. However, these solutions result in a higher fitness value because they have a large steady-state error which accumulates to a larger fitness value than the oscillations. This means that the ratio's of the controller gains, as well as the actual controller gain values, are important in improving the system.

Since this experiment resulted in algorithm design four performing best with the lowest fitness value and best response behaviour, this means that the algorithm design must change from algorithm design two to algorithm design four when the input changes from a unit step to a unit ramp for the TLBO. This shows that to achieve optimal performance, the parameters of the algorithm must change to suit the problem instance. The difference between algorithm design two and four is the search space bound since the population size is the same. This problem instance required a larger search space bound because of the large integral value.

5.2.2 Dynamic and Static Flower Pollination Algorithm Results

Table 5.3 shows the descriptive results of the FPA for the ramp input. The main difference in these results in comparison to the TLBO is the FPA illustrating a higher range of solutions in some algorithm designs, such as in algorithm design three having a maximum over best solution percentage of 113.52% for the static FPA, and algorithm design three having a maximum over best solution percentage of 137.93% for the dynamic FPA.

TABLE 5.3: Descriptive statistical results of the fitness value for the FPA for the second problem instance

	A_D	Best	Mean	Max	Std Dev	% increase from best to mean	% increase from best to max
Static FPA	1	5.9893	6.0283	6.067	0.03172	0.6517	1.297
	2	5.9434	5.9586	5.9702	0.01122	0.2552	0.4509
	3	1.7912	2.5121	3.8246	0.9296	40.2468	113.5217
	4	1.9285	2.0599	2.3004	0.1703	6.8153	19.2844
Dynamic FPA	1	5.9841	6.0467	6.1432	0.0693	1.046	2.659
	2	5.8848	5.9151	5.9502	0.0269	0.5143	1.1113
	3	1.8479	2.7479	4.3967	1.1675	48.706	137.93
	4	1.8959	1.9352	1.9893	0.03955	2.0711	4.926

The lowest mean fitness values were obtained using algorithm design four for both the static and dynamic FPA, with the lowest of the two being obtained using the dynamic FPA. Table 5.4 shows the mean resulting in PIDN solutions and corresponding computational time.

TABLE 5.4: Time domain results for the FPA for the second problem instance

	A_D	Mean Fitness	K_p	K_i	K_d	N	$t_c(s)$
Static FPA	1	6.0283	40.1764	81.8759	18.885	73.44	1524.62
	2	5.9586	63.2204	62.4606	14.018	92.984	4770.29
	3	2.5121	35.5488	439.103	0	89.712	1326.10
	4	2.0599	23.6209	499.912	0.7366	50.310	4708.81
Dynamic FPA	1	6.0467	21.6075	50.2750	18.4842	70.8343	2245.45
	2	5.9151	45.8924	67.3278	15.434	86.261	34128.27
	3	2.7479	184.644	500	0	89.077	1096.75
	4	1.9352	28.435	500	0	94.729	3246.95

Similar output to the one produced by the TLBO results on Section 5.2.1 shows that from algorithm design two to three shows a significant increase in the integral controller gain value and a decrease in the proportional and derivative controller gains. The resulting solution shows a PI controller for algorithm design three and four, which is different from the TLBO results, which showed a PIDN controller output. The increase in the integral gain is evident in Figure 5.3.

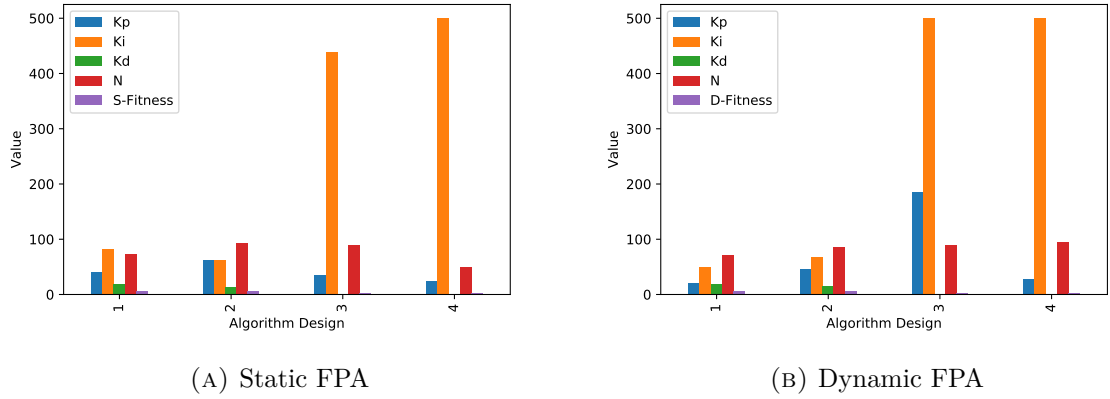


FIGURE 5.3: Bar graph showing the solution and fitness values for each algorithm design for static and dynamic FPA on the second problem instance

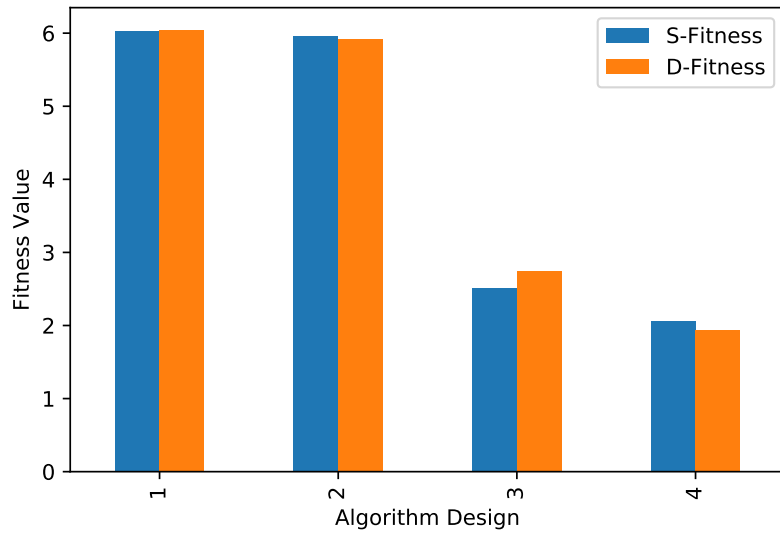


FIGURE 5.4: Bar graph showing the fitness values for each algorithm design for FPA on the second problem instance

The fitness value decreases for each algorithm design, as shown in Figure 5.3, with the lowest being on algorithm design four for both the static and dynamic FPA. The dynamic FPA obtains the lowest of the two algorithms. The behaviour of the algorithms follow the commanded input, but some algorithm designs show an oscillating response, as seen in Figure 5.5b. These are algorithm designs one and two for both the static and dynamic FPA. The algorithm showing the best

behaviour would be algorithm design three from the dynamic FPA since it resulted in following the commanded input the closest and fastest.

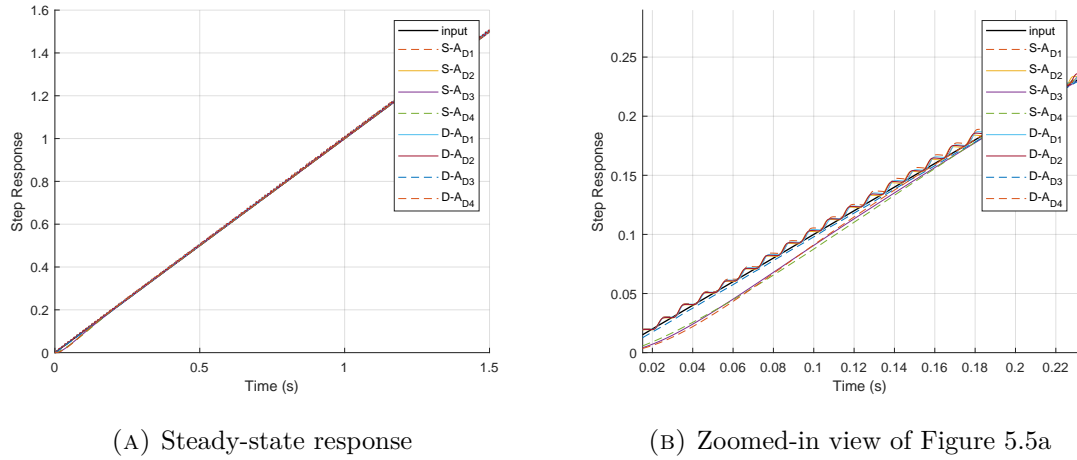


FIGURE 5.5: The ramp response of the algorithm designs for the FPA

Since this problem favoured the dynamic FPA instead of the static FPA, it may be that this problem is not multi-modal and thus requires fine-tuning of a specific region in the search space rather than searching for other regions in the search space. Also, the increase in the search space greatly reduced the fitness function value. This problem instance may require a higher search space bound, which is also what was found with the TLBO results.

5.2.3 Dynamic and Static Genetic Algorithm Results

Table 5.5 shows that the lowest fitness value was obtained by algorithm design four for the static GA, and algorithm design three for the dynamic GA, with the lowest of the two being the static GA. The highest standard deviation obtained was by algorithm design one for the static GA with a value of 0.1118, and algorithm design four for the Dynamic GA with a value of 0.3211.

TABLE 5.5: Descriptive statistical results of the fitness value for the GA for second problem instance

	A_D	Best	Mean	Max	Std Dev	% increase from best to mean	% increase from best to max
Static GA	1	5.8788	6.0265	6.1493	0.1118	2.5130	4.6013
	2	5.819	5.8851	5.9591	0.05747	1.1354	2.4076
	3	1.7981	1.8645	1.95	0.06346	3.6946	8.4478
	4	1.7851	1.8055	1.8297	0.01841	1.1409	2.4985
Dynamic GA	1	5.9386	6.0641	6.1784	0.09822	2.1139	4.0380
	2	5.8286	5.8897	5.9499	0.04952	1.0477	2.0811
	3	1.8052	1.9381	2.0947	0.1194	7.3602	16.0370
	4	1.8499	2.0991	2.5525	0.3211	13.4728	37.9804

Table 5.6 shows the PIDN solutions from the different algorithm designs. This table shows that both GA variations obtained PIDN solutions for all the algorithms designs.

TABLE 5.6: Time-domain results for the GA for the second problem instance

	A_D	Mean Fitness	K_p	K_i	K_d	N	$t_c(s)$
Static GA	1	6.0265	46.6182	63.7817	27.2353	55.4945	1193.14
	2	5.8851	73.2471	56.6878	17.4954	72.4885	3440.20
	3	1.8645	11.6549	487.486	2.56039	6.05713	2189.95
	4	1.8055	14.5861	493.192	17.5416	1.96974	3926.19
Dynamic GA	1	6.0641	52.4165	65.0099	35.1302	44.8876	1159.19
	2	5.8897	91.6233	95.4977	17.2062	74.87	3610.61
	3	1.9381	17.9465	489.75	3.5627	14.71	1042.06
	4	2.0991	14.2973	495.80	13.0905	3.240	5218.73

Algorithm designs three and four both resulted in a higher integral gain and relatively low proportional and derivative gain, similar to what was observed for the FPA and TLBO results. The change in integral gain is also observed in Figure 5.6 for both variations of the GA.

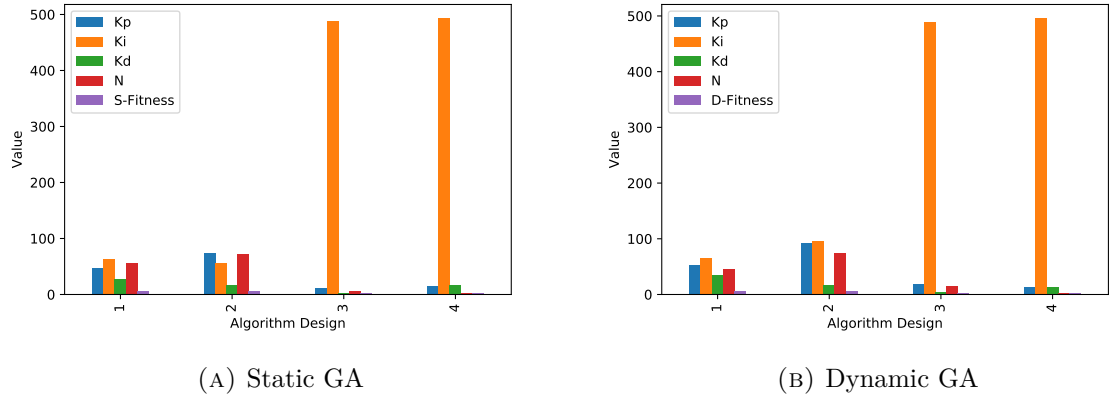


FIGURE 5.6: Bar graph showing the solution and fitness values for each algorithm design for GA on the second problem instance

The algorithm that resulted in the lowest fitness value for all the algorithm designs is indeed the static GA, as shown in Figure 5.7.

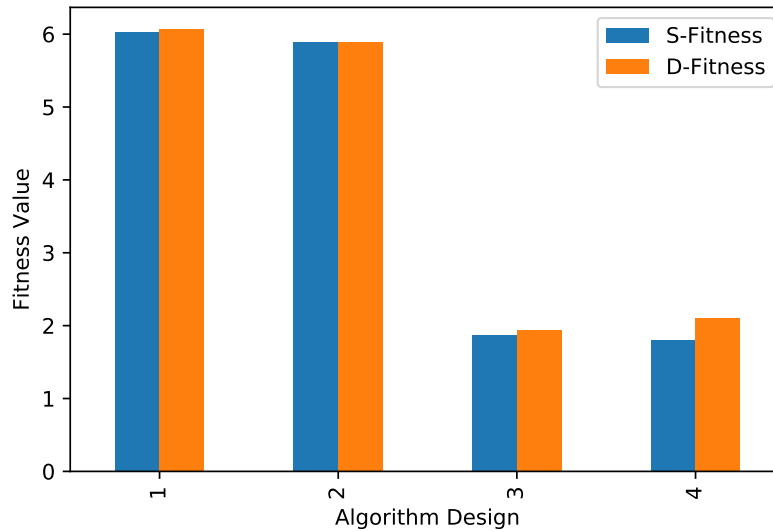


FIGURE 5.7: Bar graph showing the change of fitness value with changing algorithm design for the GA algorithm on the second problem instance

The ramp response in Figure 5.8 gives the same trend in that the algorithm designs one and two provide oscillatory responses. Though algorithm design three for the dynamic GA resulted in tracking the commanded input appropriately, this result also displayed small oscillations, as seen in Figure 5.8. This is because of the

higher proportional gain seen on algorithm design three as compared to four for the dynamic GA.

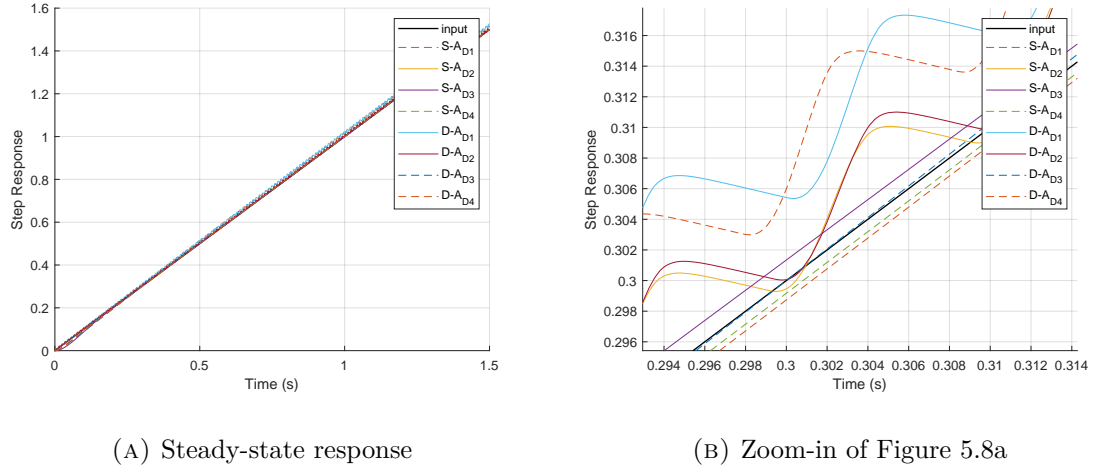


FIGURE 5.8: The ramp response of the algorithm designs for the GA

Because of this, the static GA is most appropriate for this instance, as opposed to the dynamic GA. The static GA also resulted in the lowest fitness value obtained.

This is a similar result obtained by the FPA results shown on Section 5.2.2 as this section also favoured the algorithm-specific parameters which show more exploitation rather than exploration, as seen by the static GA. Algorithm design four has been observed to be the best performing since this resulted in the lowest fitness value and appropriate ramp response behaviour.

This further confirms that the problem is not multi-modal, and the optimal solution is found in one region. This is why this problem benefits from fine-tuning the solution rather than exploring other regions. This also means that the algorithm was capable of obtaining the optimal region in the search space, as fine-tuning will only be beneficial when the optimal region is found.

5.3 Additional Non-Linearities

This problem instance for this experiment is described as the target of interest moving at a constant velocity modelled as a step input, but with additional non-linearity including base motion and friction. The three algorithms were tested on this problem instance, with changing the algorithm designs from one to four.

5.3.1 Parameterless Teaching Learning Based Optimization Results

Table 5.7 shows that the TLBO maintains its robustness with the maximum percentage over the best solution, not exceeding 0.4%. The lower this percentage is, the higher the probability of achieving the same or similar solution after each independent run, and the more robust the algorithm is, by this measure.

TABLE 5.7: Descriptive statistical results for the TLBO for the last problem instance

A_d	Best	Mean	Max	Std Dev	% Av over best sln	% Max over best sln
1	32.6425	32.6858	32.757	0.05072	0.1328	0.3508
2	32.6297	32.6304	32.6317	0.000899	0.002247	0.006129
3	32.6945	32.717	32.7392	0.01825	0.06882	0.1367
4	32.6267	32.6669	32.6932	0.02889	0.1233	0.2038

TABLE 5.8: Time-domain results for the TLBO for the last problem instance

A_D	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p %	t_c (s)
1	32.69	0.02926	0.001432	2.7849	9.4272	0.4696	1.4391	3.4600	1250.69
2	32.63	0	0	2.77	9.53	0.4742	1.4387	3.4433	3607.96
3	32.72	0	0	2.7228	10.0261	0.4856	1.4372	3.3897	1166.06
4	32.67	0	0	2.8121	9.3537	0.4663	1.4391	3.4608	3489.32

What is interesting to note is that the solutions in Table 5.8 mostly show that the control system is comprised of the derivative controller and filter coefficient, i.e. a DN solution, except for algorithm design one. The lowest fitness value is achieved

by algorithm design two, with a fitness value of 32.63. Figure 5.9b also illustrates this.

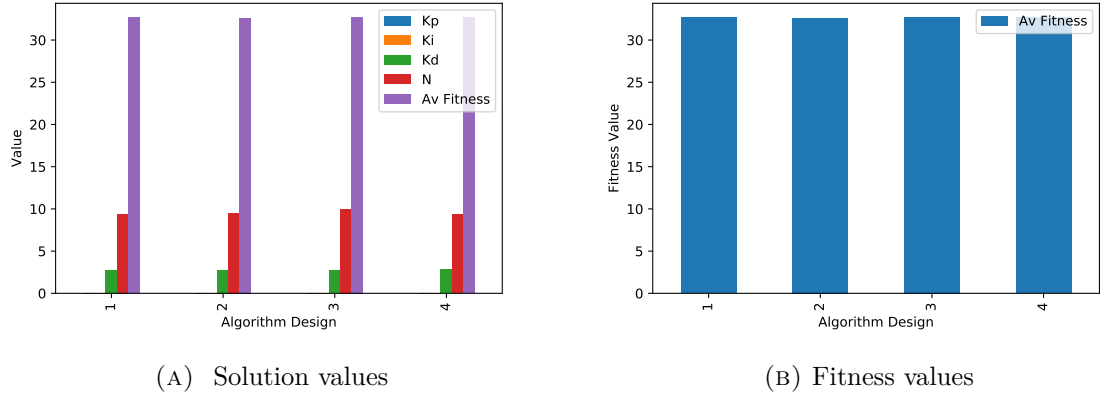


FIGURE 5.9: Bar graph showing the solution and fitness values for each algorithm design for TLBO on the last problem instance

Figure 5.10 illustrates the time domain response and shows that the change in algorithm design did not make a significant difference in response. The step response in Figure 5.11 confirms that the behaviour of the solutions for all the algorithm designs is similar.

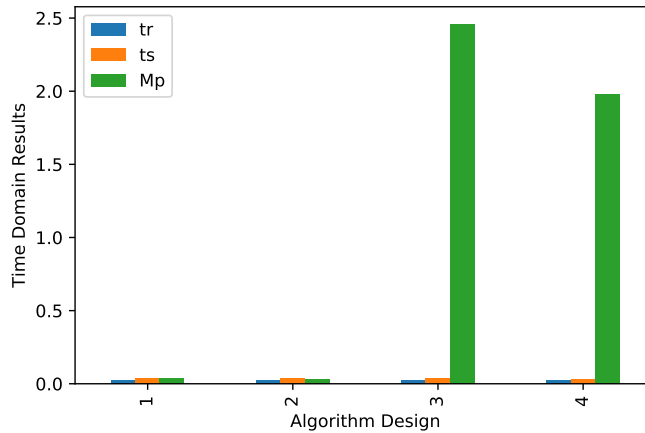


FIGURE 5.10: Bar graph showing the time domain responses with changing algorithm design for the TLBO on the last problem instance

This step response also shows that the solutions were not able to overcome the torque disturbance produced by added non-linearity in the system.

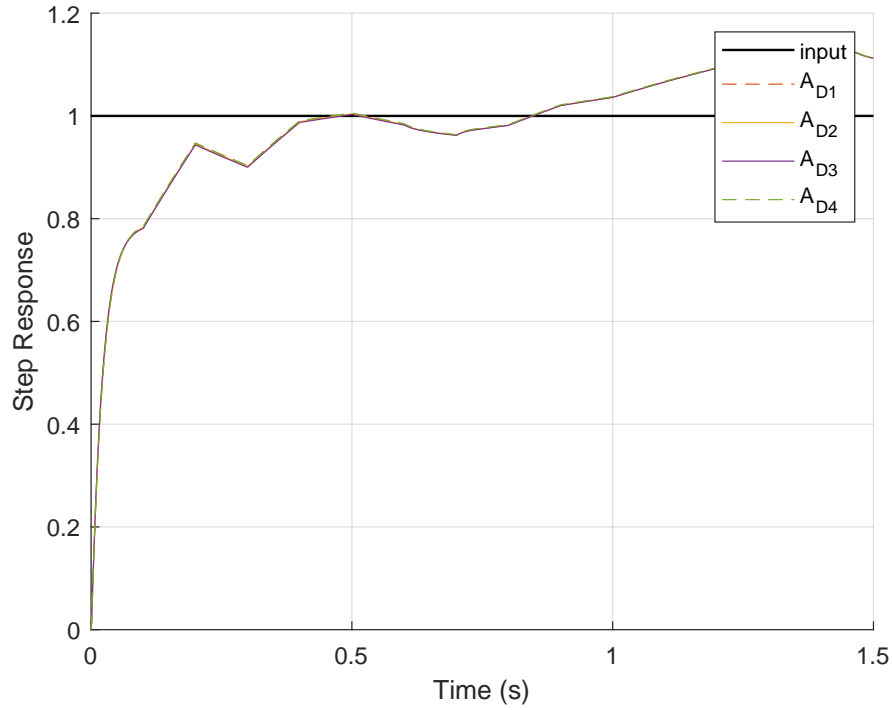


FIGURE 5.11: The step response of the algorithm designs for the TLBO on the last problem instance

This indicates that either the optimal solutions are not present in the search space are not present at all, or that the algorithm was not capable of finding these solutions.

Removing the voltage limit in the control system resulted in a significant improvement in the results, as shown in Figure 5.12. This result was obtained using the algorithm design two since it was chosen as the best fitting for the step response study in the first problem instance experiments. Since this problem instance requires a higher voltage for the system to perform, it means it requires a higher motor torque order overcome the disturbance torque.

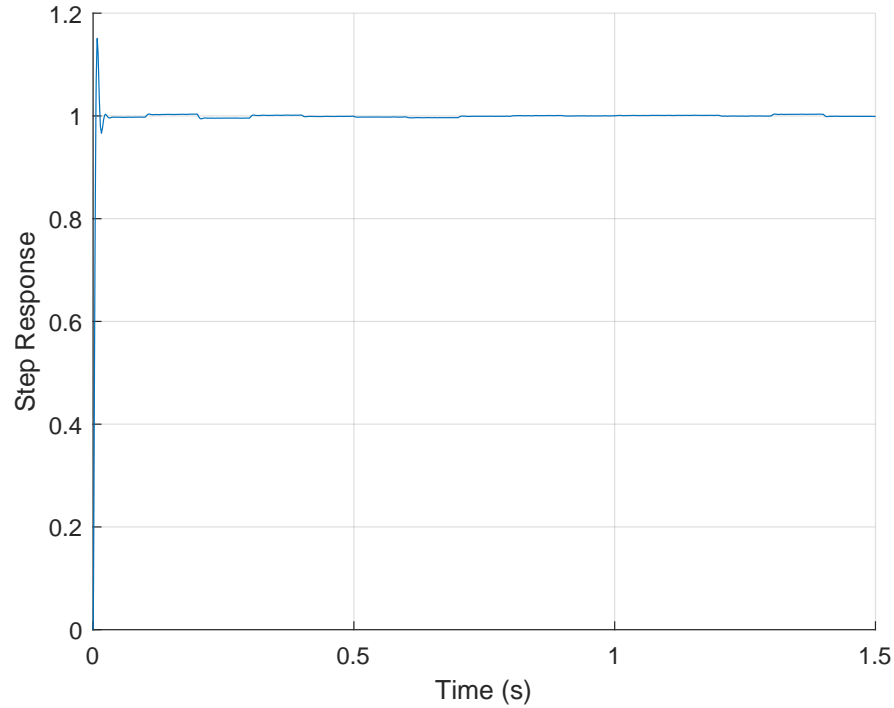


FIGURE 5.12: The step response showing the TLBO algorithm without the voltage constraint of the motor

The required torque can be achieved by choosing a different motor which satisfies these requirements. However, a larger motor which has a larger voltage limit also comes with its disadvantages such as increased size and weight of that motor which inevitably increases the size and weight of the system.

Another solution is to input gears into the system so that this increases the torque output the gimbal. However, there are disadvantages to inputting gears such as reaction torques from a geared actuator which causes torque disturbance [12]. Hilkert [12] states that gearing arrangements are highly likely to introduce additional friction and torsional resonances in the system, which is why the direct drive was chosen for this system. Also, increasing the torque results in decreasing the velocity, which then means that the gimbal will respond slower to the commanded input.

5.3.2 Dynamic and Static Flower Pollination Algorithm Results

TABLE 5.9: Descriptive statistical results for the FPA for the last problem instance

	A_d	Best	Mean	Max	Std Dev	% Av over best sln	% Max over best sln
Static FPA	1	33.586	37.2782	39.2475	2.6127	10.993	16.8567
	2	35.2428	37.5304	38.698	1.6177	6.4911	9.8040
	3	38.49	39.3525	39.8738	0.6143	2.2409	3.5952
	4	39.6738	39.8154	40.0393	0.1602	0.3569	0.9213
Dynamic FPA	1	35.7665	37.8904	39.2142	1.5169	5.9381	9.6395
	2	37.30	38.9643	38.9643	0.6845	4.4588	4.4588
	3	38.8103	40.1025	40.1025	0.5373	3.3295	3.3295
	4	38.6823	39.5351	39.5351	0.3483	2.2046	2.2046

TABLE 5.10: Time-domain results for the FPA for the last problem instance

	A_D	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p %	t_c (s)
Static FPA	1	37.28	54.77	5.151	6.424	23.920	0.1937	1.4460	3.7808	1524.62
	2	37.53	24.38	0	11.76	19.88	0.3689	1.446	3.7939	3604.63
	3	39.35	345.68	72.685	20.731	35.964	0.1683	1.4460	3.7756	1326.10
	4	39.82	230.39	89.175	14.128	46.725	0.1785	1.4459	3.7747	4708.81
Dynamic FPA	1	37.8904	65.824	1.306	6.5731	26.775	0.1917	1.4460	3.7756	1355.88
	2	38.9643	99.2391	5.03345	6.4573	28.143	0.1712	1.4459	3.7723	3496.21
	3	40.1025	390.261	3.90398	40.6506	27.273	0.1853	1.4461	3.7870	1356.44
	4	39.5351	499.960	10.2658	28.7254	32.559	0.1636	1.4459	3.7748	3634.05

The FPA descriptive fitness value solutions for this problem instance is shown in Table 5.9, illustrating a higher range of solutions as compared to the TLBO, shown in Table 5.7. The lowest fitness value was obtained by algorithm design one for both the static and dynamic FPA, with the static FPA being the lowest. Furthermore, the solutions are comprised of a PIDN controller shown in Table 5.10 rather than the DN solutions shown in Table 5.8 by the TLBO. Figure 5.13 shows the increase in the proportional gain when the algorithm design changed from two to three, which has not made a great impact on the fitness value shown in Figure 5.14 and has increased the fitness value.

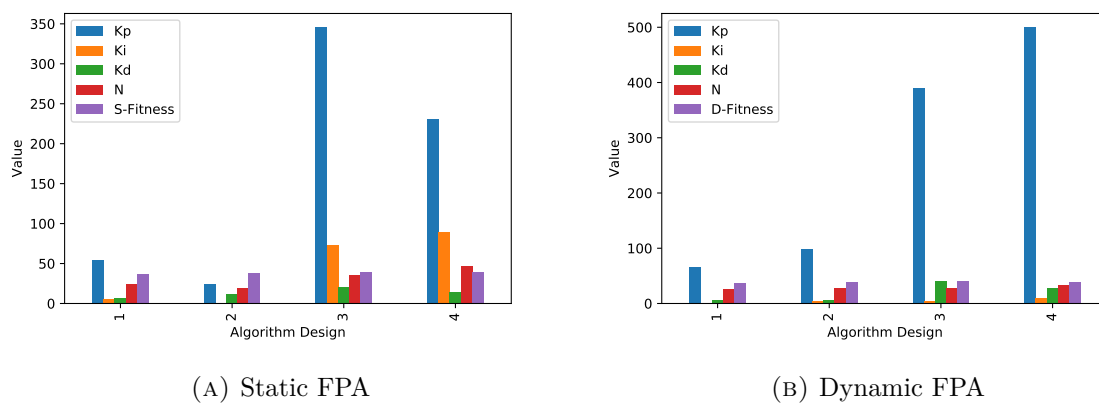


FIGURE 5.13: Bar graph showing the solution and fitness values for each algorithm design for FPA and the last problem instance

The time-domain results have also remained relatively the same throughout the algorithm designs, as shown in Figure 5.15.

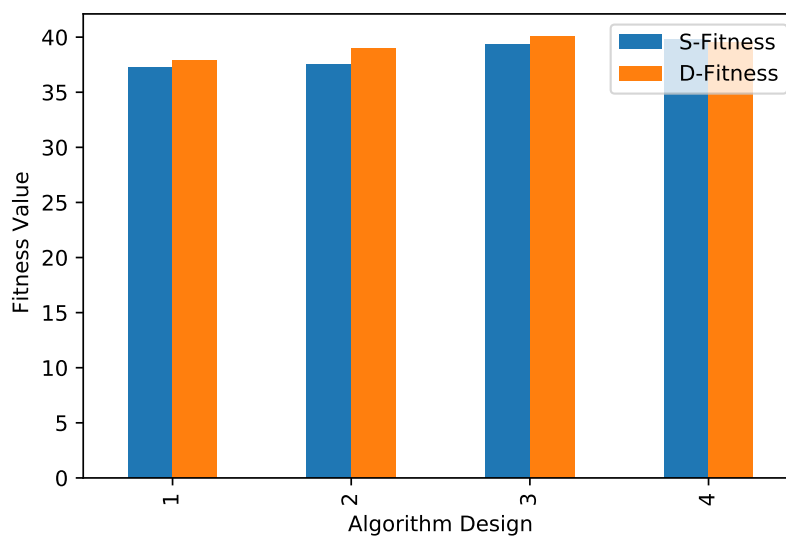


FIGURE 5.14: Bar graph showing the change of fitness value with changing algorithm design for the FPA algorithm on the last problem instance

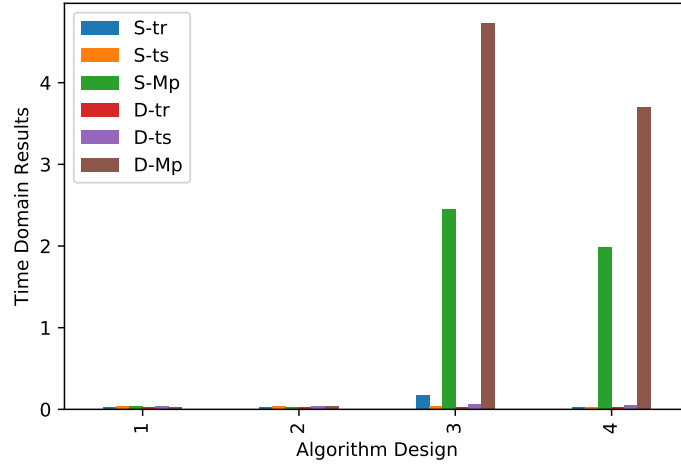


FIGURE 5.15: Bar graph showing the change of time-domain responses with changing algorithm design for the static and dynamic FPA on the last problem instance

The step response shown in Figure 5.16 shows a bigger range in behaviour as compared to the TLBO, but still shows no real improvement or satisfactory performance.

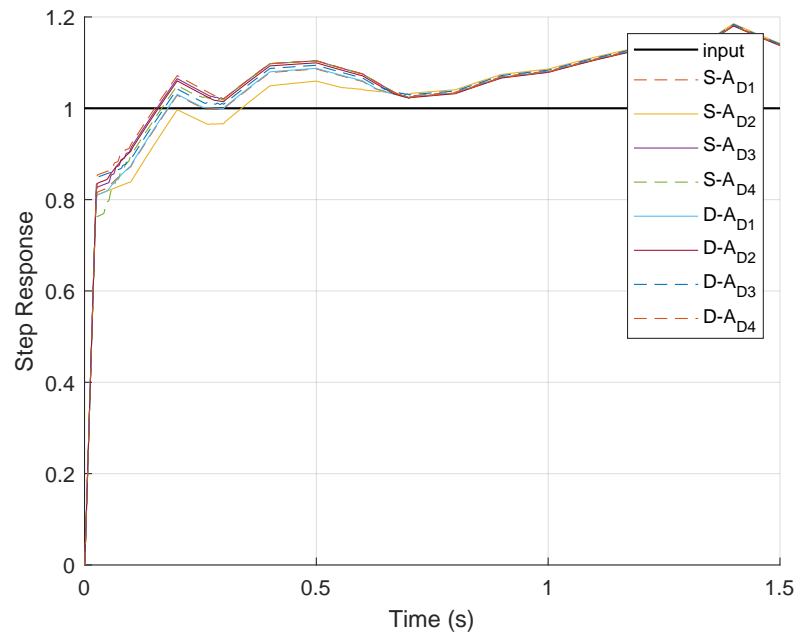


FIGURE 5.16: The step response of the algorithm designs for the FPA with the last problem instance

This further indicates that the problem may not be the algorithm itself, but the constraints of the control system and search space.

Figure 5.17 shows the result of the Dynamic FPA without a voltage limit, using algorithm design two.

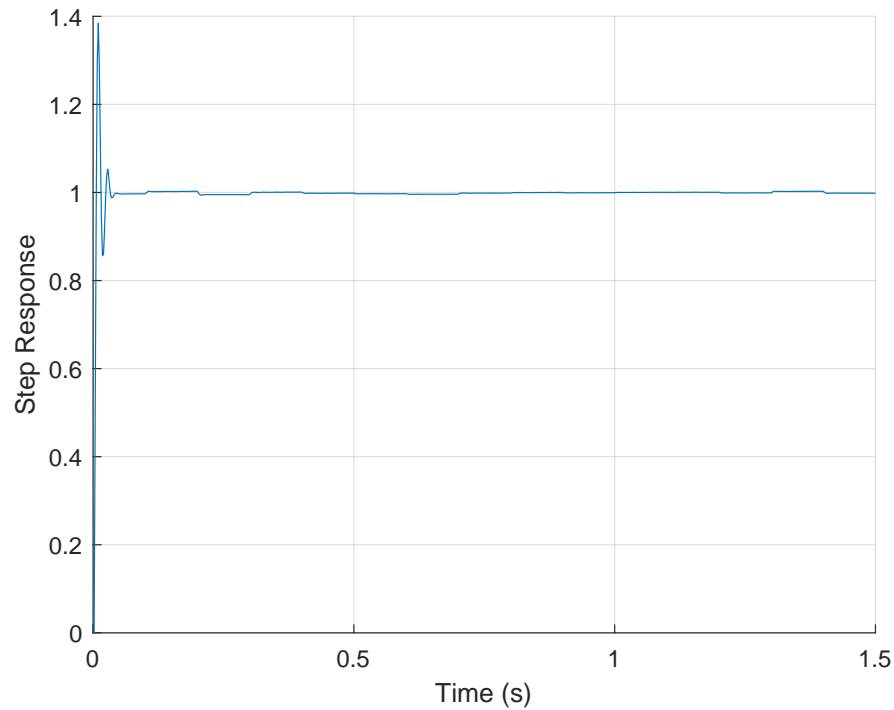


FIGURE 5.17: The step response showing the Dynamic FPA algorithm without the voltage constraint of the motor

Though this indicates a large overshoot, this can be reduced by decreasing the proportional gain. Compared to 5.16, removing the voltage has increased performance.

5.3.3 Dynamic and Static Genetic Algorithm Results

Table 5.11 showing the descriptive statistical results illustrates that the static GA achieved the lowest fitness values, and the dynamic GA achieved the highest. Table 5.12 shows that the algorithm which gave the lowest fitness value is the static

TABLE 5.11: Descriptive statistical results for the GA on the last problem instance

Algorithm Type	A_D	Best	Mean	Max	Std Dev	% Av over best sln	% Max over best sln
Static GA	1	34.8727	35.5638	36.6387	0.7704	1.982	5.064
	2	34.2518	34.5305	34.7232	0.2018	0.8136	1.376
	3	39.2131	39.3275	39.4161	0.08486	0.2917	0.5177
	4	39.1988	39.3025	39.5019	0.1410	0.2645	0.7732
Dynamic GA	1	36.1146	36.7911	37.4648	0.5512	1.873	3.739
	2	34.6821	35.5134	36.125	0.6092	2.3969	4.160
	3	39.2968	39.4862	39.6646	0.1504	0.4821	0.9360
	4	39.1272	39.1898	39.2219	0.04429	0.1601	0.2420

GA using algorithm design two. Algorithm design two gave the lowest mean fitness value for both the static and dynamic GA.

TABLE 5.12: Time-domain results for the GA for the last problem instance

	A_D	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p %	t_c (s)
Static GA	1	35.56	3.5573	2.8851	3.8855	18.129	0.4726	1.4336	3.2872	1196.77
	2	34.53	3.76	0.34	3.52	21.53	0.8299	1.4289	3.0935	3748.57
	3	39.3	408.48	24.792	23.017	31.738	0.1621	1.4459	3.7735	1138.60
	4	39.30	391.06	116.93	21.655	31.280	0.1585	1.4459	3.7734	3463.39
Dynamic GA	1	36.79	14.8797	0.8995	6.9688	31.4849	0.8099	1.4372	3.4158	1129.06
	2	35.51	5.16	1.65	2.74	23.92	0.8168	1.4396	3.4989	3361.75
	3	39.49	437.475	47.5729	27.8541	28.0786	0.1634	1.4460	3.7805	1130.83
	4	39.19	330.992	24.2029	15.8751	36.1856	0.1591	1.4459	3.7709	3356.82

The GA also results in a full PIDN solution rather than the DN solution observed by the TLBO algorithm.

Figure 5.18 shows a rise in the proportional gain value from algorithm design two to three, which has resulted in a rise in the fitness value, as shown in Figure 5.19.

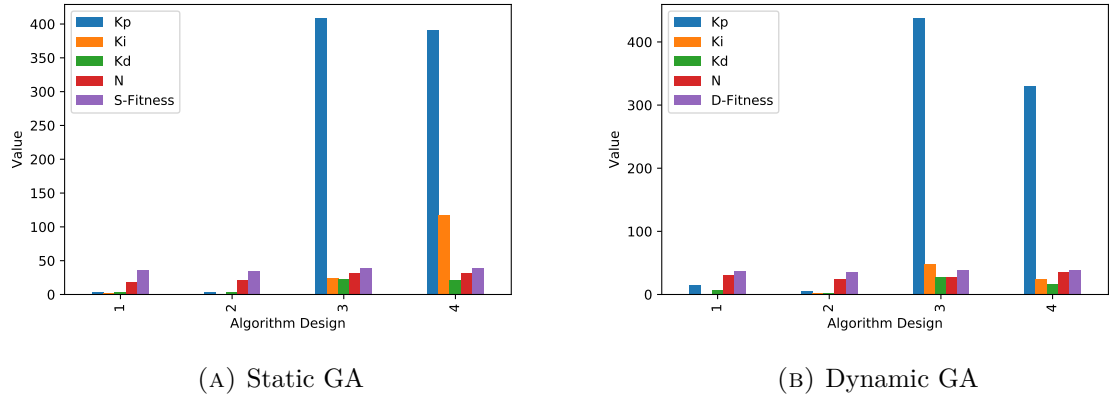


FIGURE 5.18: Bar graph showing the solution and fitness values for each algorithm design for FPA on the last problem instance

This rise in proportional gain resulted in a decrease in the rise time for both the static and dynamic GA but was accompanied by the rise in percentage overshoot, as shown in Figure 5.20. The settling time has remained the same, which is the duration of the simulation itself, being 1.5 seconds. This indicates that the signal did not follow the input command within the time frame of 1.5 seconds.

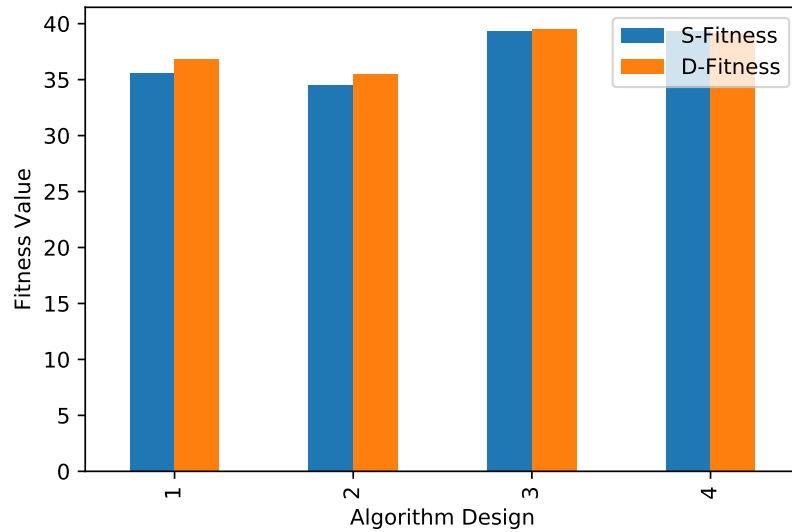


FIGURE 5.19: The change of fitness value with changing algorithm design for the static and dynamic GA on the last problem instance

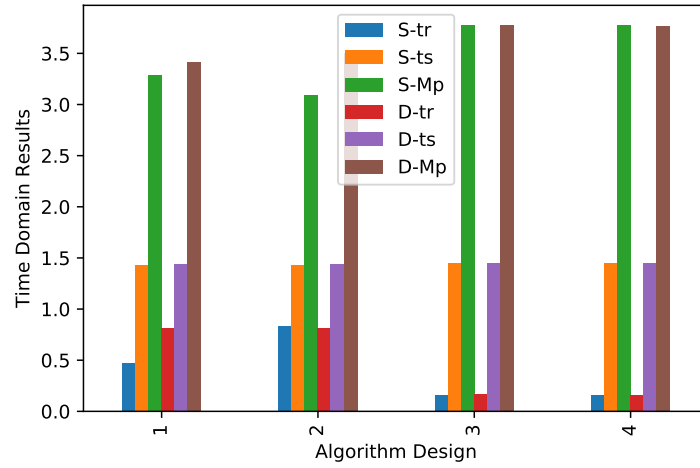


FIGURE 5.20: The change of the time domain responses with changing algorithm design for the static and dynamic GA algorithm on the last problem instance

The step response of the algorithms is shown in Figure 5.21. This further indicates that the GA was not able to find a suitable solution to the control system.

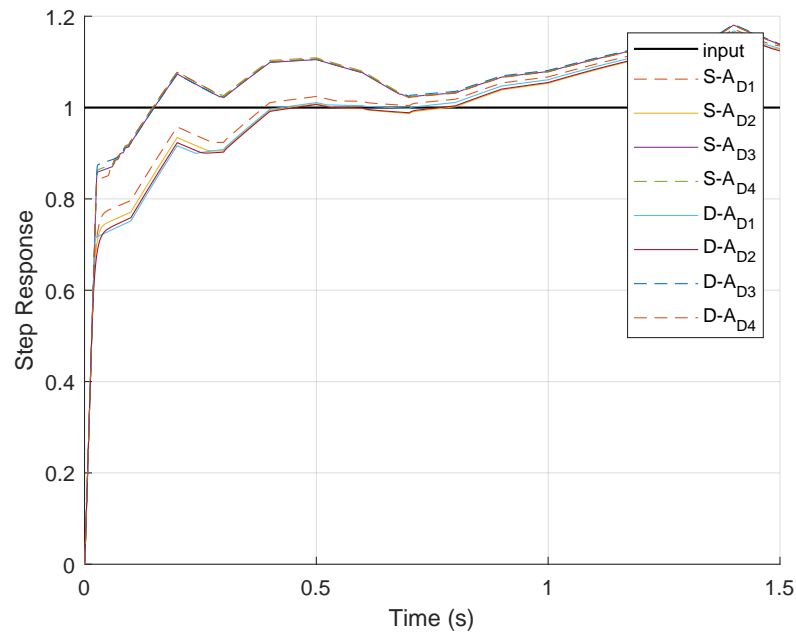


FIGURE 5.21: The step response of the algorithm designs for the GA on the last problem instance

Figure 5.22 shows that no voltage limit has increased the gimbal response when using the dynamic GA with algorithm design two.

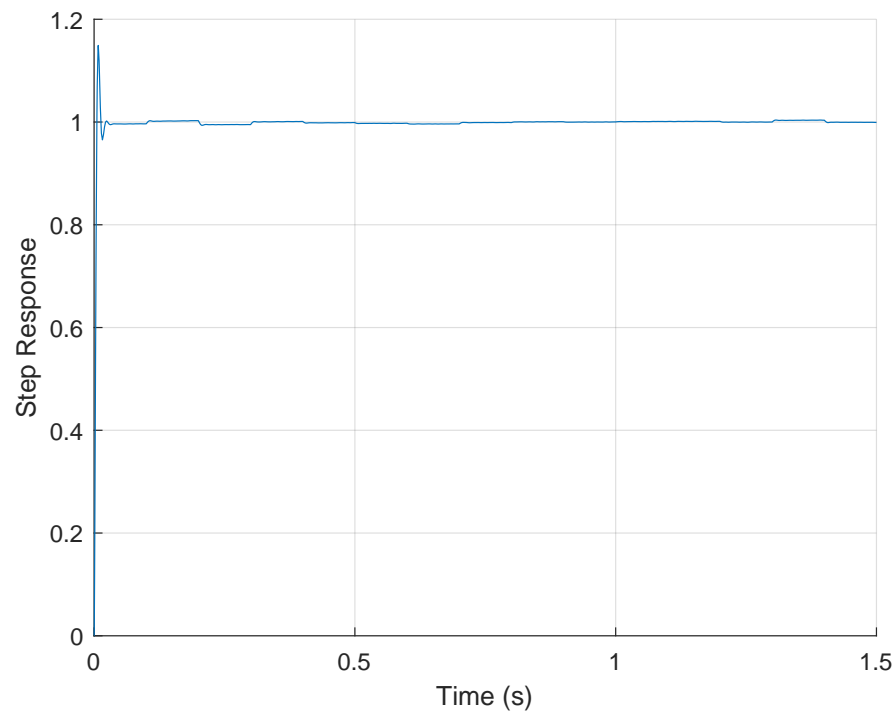


FIGURE 5.22: The step response showing the dynamic GA algorithm without the voltage constraint of the motor on the control system

With the three algorithms not being able to perform when the system has added non-linearity further indicates that the problem is not with the algorithms but with the constraints of the control system itself, as stated in Section 5.3.1 when the performance of the TLBO was discussed.

5.4 Concluding Remarks

5.4.1 Dynamic Input Conclusions

From the experiments conducted, the following concluding remarks are made:

- There is a shift in the importance of the controller gain as the input of the control system changes. The integral controller becomes more critical in tracking the commanded input is tracked while reducing overshoot.
- The algorithm design with the higher search space upper bound resulted in the lowest fitness value for this problem instance. This was seen in algorithm design three and four with algorithm design four, producing the lowest fitness value for all the algorithms.
- The ramp input also requires more exploitation rather than exploration. Hence this problem instance favoured dynamic FPA and static GA. This may be due to this problem also not being multi-modal, and thus the algorithm must be able to focus on a specific region of the search space.
- The initial common parameter design that was chosen to be the best in Chapter 4 (algorithm design two) has changed in this Chapter to algorithm design four. These means that these parameters are not robust and must be changed with changing problem instances.
- Algorithm design four contains a larger search space bound of 500, with a larger population size of 60.

5.4.2 Additional Non-Linearity Conclusions

The experiments conducted allowed for the following concluding remarks to be made:

- The overall performance of the control system is not satisfactory for all algorithms. This indicates one of two possibilities: the algorithms cannot operate under problems with additional non-linearity, or there is no solution for this control system under the constraints given in this problem instance.
- Removing the voltage limit of the motor improves the performance of all the algorithms and system. This may mean that there is not enough voltage, and thus not enough torque in order for the actuator to overcome the disturbance torque.
- However, removing the voltage limitation of the motor makes the system non-practical as that limit is used as part of modelling the motor closer to real-life.

5.4.3 Suggestions for Future Practitioners

From these conclusions, some considerations or suggestions can be drawn for future heuristic algorithm practitioners when tackling this kind of problem with this kind of meta-heuristic algorithms:

1. If the target is already moving with a constant velocity (step input) when the simulation begins, then proportional gain should be high since the gimbal is starting from zero velocity and needs to reach the target velocity. However, if the target begins from zero as well and moves with constant acceleration (ramp input), then the integral gain is significant which shows the error over time because the gimbal moves with the target rather than attempting to reach the target motion.
2. If the system is required to overcome torque disturbance, consider adding arrangements that increase motor torque in the system in order to increase the performance of the algorithms and solution quality. The search space bound must adjust accordingly.

CHAPTER 6

Performance Evaluation

This chapter compares the performance of the algorithms, not only by using the performance evaluations shown in Chapter 4 and 5 but by other crucial performance evaluations which are relevant to meta-heuristic algorithms.

The chapter begins by comparing the performance of the meta-heuristic algorithms in the different problem instances. It continues to conduct a statistical analysis of the performance of the algorithms, to quantify whether there was a difference in performance. The chapter then focuses on the convergence of algorithms, along with the time complexity analysis. The chapter ends off with concluding remarks.

6.1 Research Questions

The research questions for this particular case study are as follows:

1. How does the performance of the algorithms when observing their fitness value, behaviour, statistical analysis and convergence for the different problem instances?

6.2 Comparison of Algorithm Performance using Fitness Value and Behavioural Response

Table 6.1 compares the performance for the three chosen best-performing algorithms from Chapter 4 for the first problem instance, which is described as a step input with no additional non-linearity.

TABLE 6.1: Comparison of time-domain results for a step input

Algorithm Type	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	% OS	t_c (s)	% increase from best to max
TLBO	34.5981	99.8965	49.7202	82.3452	0.2096	0.0252	0.035	0.0329	3273.93	0.01185
Dynamic FPA	34.7257	99.2101	46.3896	0	50.7338	0.0257	0.0378	0.0355	3316.39	0.02535
Dynamic GA	34.7191	98.6337	46.0652	0.6798	3.4715	0.0257	0.0379	0.0269	3455.12	0.6004

The table shows that the algorithm with the lowest fitness value for this particular instance is the TLBO. The table also shows that the TLBO has the lowest rise and settling time, with the lowest overshoot obtained by the dynamic GA. The TLBO also resulted in the lowest range of solutions, with the percentage increase from the lowest to highest fitness value being 0.0118%. This makes the TLBO more likely to achieve a solution of similar quality at each run, in comparison to the other algorithms. In other words, the TLBO has a higher success rate when compared to the other algorithms for this problem.

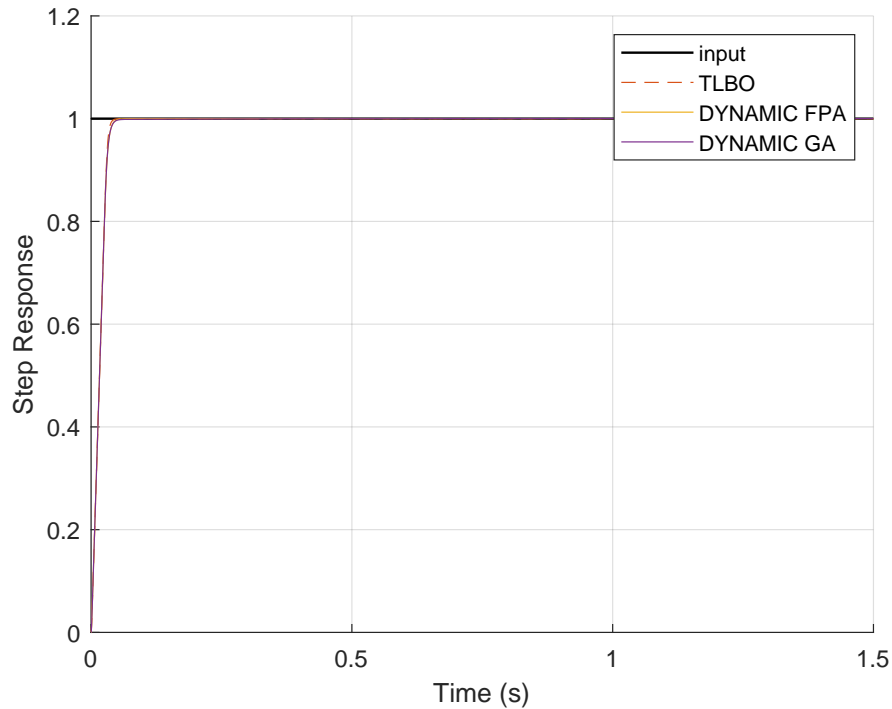
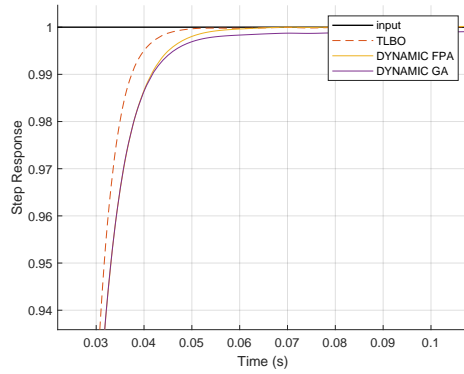
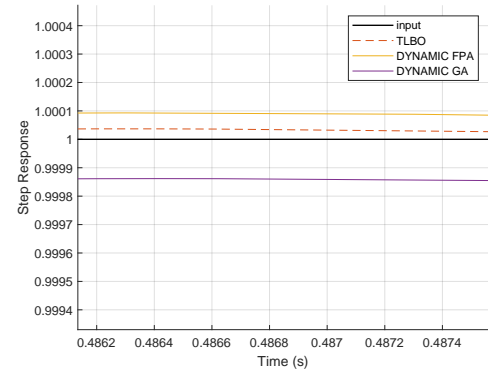


FIGURE 6.1: The step response comparing the algorithm performance for the plant with step input and no additional friction



(A) Close up showing transient response



(B) Close up showing steady-state response

FIGURE 6.2: The step response comparing the algorithm performance for the control system with step input

Figure 6.1 shows that the step response behaviour of all the algorithms is very similar in performance. The differences are shown in Figure 6.2 as the TLBO reaches the commanded input faster (because it contains the lowest rise time) as seen in Figure 6.2a, and also follows the commanded input closer in comparison to the other algorithms, as seen in Figure 6.2b.

As mentioned previously, the difference between the TLBO and the other algorithms lies in the TLBO being ‘parameterless’. The algorithm-specific parameters are used to control the exploration and exploitation of the algorithms. The TLBO does not have these parameters because there is a guarantee that the algorithm performs exploration and exploitation. Exploration and exploitation control how the algorithm moves through the search space to obtain the optimal solution. The TLBO performing best in the first problem instance shows that guaranteeing both the exploration and exploitation is better than assigning parameters which control the probability of which one occurs.

Since the exploration and exploitation occur in one iteration, each iteration fine-tunes the solutions which give a reason as to why the TLBO has a higher success rate, when compared to the other algorithms.

Table 6.2 shows that for the problem instance involving a ramp input, which shows that the TLBO algorithm has the lowest fitness value.

TABLE 6.2: Comparison of fitness value and algorithm solutions

Algorithm Type	Mean Fitness	K_p	K_i	K_d	N	$t_c(s)$	% increase from best to max
TLBO	1.7662	16.5460	499.73	3.7085	58.2873	2939.56	0.6590
Dynamic FPA	1.9352	28.4353	500	0	94.7292	3246.95	4.9264
Static GA	1.8055	14.5861	493.192	17.542	1.9697	3926.19	2.4985

The TLBO also resulted in the lowest fitness value and range of solutions. The highest fitness value was obtained by the dynamic FPA, along with the highest range of solutions.

Figure 6.3 showing the behaviour of the algorithm performance illustrates that the best performing algorithm is the dynamic FPA with it following the ramp input closest.

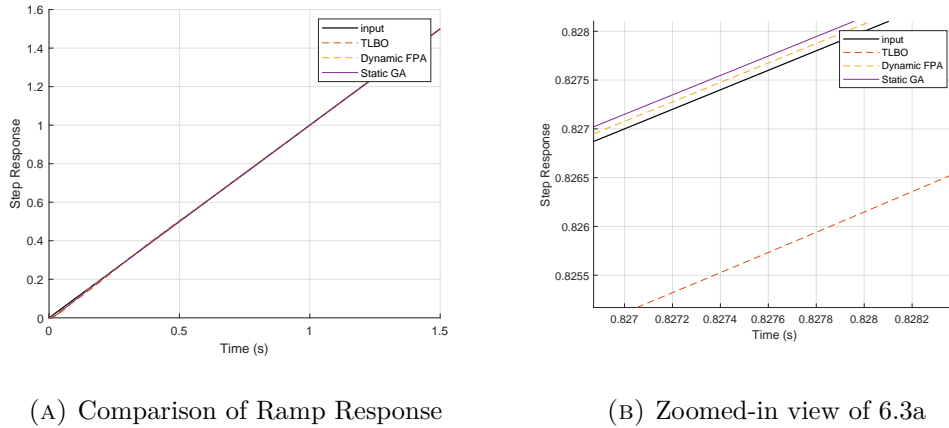


FIGURE 6.3: The comparison of the ramp response of the algorithm performance

Table 6.3 compares the time-domain results for the problem instance with added non-linearity.

TABLE 6.3: Comparison of time-domain results for a step input and added non-linearity

Algorithm Type	Mean Fitness	K_p	K_i	K_d	N	t_R (s)	t_S (s)	M_p %	t_c (s)
TLBO	32.63	0	0	2.77	9.53	0.4742	1.4387	3.4433	3607.96
Static FPA	37.53	24.38	0	11.76	19.88	0.3689	1.4462	3.7939	3604.63
Dynamic GA	35.51	5.16	1.65	2.74	23.92	0.8168	1.4396	3.4989	3361.75

From this table, it is seen that the lowest fitness value is produced by the TLBO, which also produced the solution with the lowest settling and overshoot percentage time. The lowest rise and computational time are obtained from the dynamic GA.

Figure 6.4 compares the step response behaviour for the system with added non-linearity.

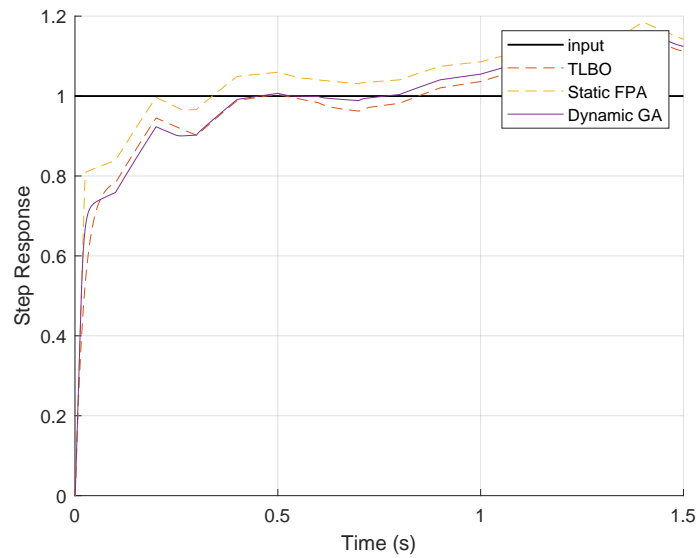


FIGURE 6.4: The comparison of the step response of the algorithm performance on the control system with additional non-linearity

As seen in Figure 6.4, the step response behaviour is very similar to the algorithm performance being affected by the non-linearity similarly presented in the system.

6.3 Statistical Analysis

Table 6.4 shows the mean fitness values obtained from the three different problem instances. The mean fitness is then used as the data required in order to implement the Friedman inferential statistical analysis.

TABLE 6.4: Data for the Friedman Test

Plant Type	TLBO	FPA	GA
Problem Instance 1 (step input)	34.60	34.77	34.72
Problem Instance 2 (ramp input)	1.77	1.94	1.81
Problem Instance 3 (added non-linearity)	32.63	37.53	34.53

The Friedman test results are shown in Table 6.5.

TABLE 6.5: Friedman Test results

Problem Instances	Algorithms	Sum of Squared Ranks	Q	cv	DF ($k - 1$)	α
3	3	42	18	6	2	0.05

Since the computed Q , which is the Friedman statistic is higher than the critical value cv , the null hypothesis is rejected, which means that the algorithms do not have identical effects. There is a significant difference in performance.

Tables 6.6, 6.7, and 6.8 show the results of the posthoc tests.

TABLE 6.6: Posthoc comparison of absolute difference among mean ranks

	TLBO	FPA	GA
TLBO	0	0	0
FPA	6	0	0
GA	3	3	0

This means that there is only one pair-wise comparison of the algorithms which showed a significant difference in the performance, which is the comparison between the TLBO and the FPA with the second problem instance. The statistical

analysis observed the performance of these two algorithms in this problem instance as the only significant difference since this is the only comparison where the p-value is less than the chosen level of significance α .

TABLE 6.7: Posthoc comparison of p-values

	TLBO	FPA	GA
TLBO	0	0	0
FPA	0.0093	0	0
GA	0.3056	0.3056	0

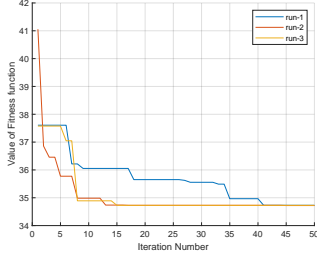
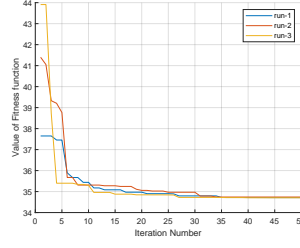
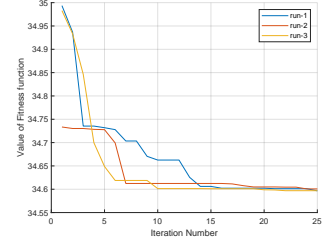
TABLE 6.8: Posthoc comparison of p-values $< \alpha$

	TLBO	FPA	GA
TLBO	0	0	0
FPA	1	0	0
GA	0	0	0

According to the statistical analysis, the algorithms performed relatively the same with each problem instance, except for the TLBO and FPA on the second problem instance.

6.4 Convergence

Figure 6.5 shows the run-length distributions of the chosen algorithms and algorithm design for all the problem instances. TLBO was compared for 25 iterations because this algorithm performs fitness evaluations twice in one iteration, rather than once in one iteration like the other algorithms.

(A) Dynamic FPA
with step input(B) Dynamic GA with
step input

(C) TLBO with step input

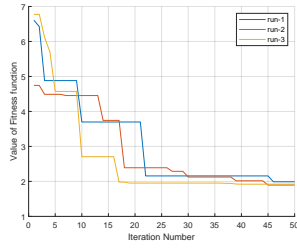
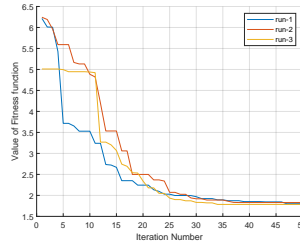
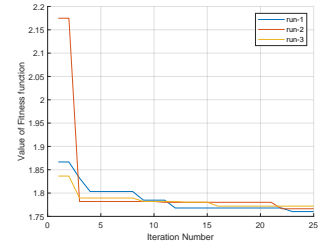
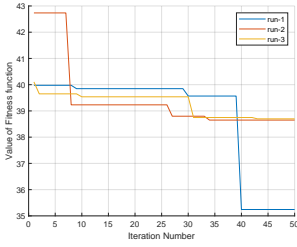
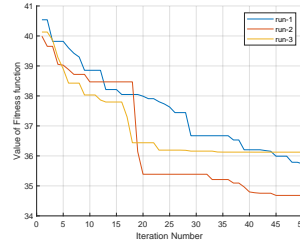
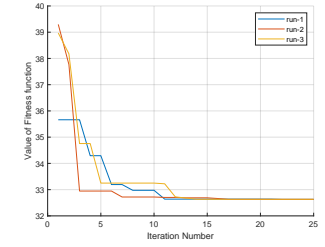
(D) Dynamic FPA
with ramp input(E) Static GA with ramp
input(F) TLBO with ramp in-
put(G) Static FPA
with added non-
linearity(H) Dynamic GA with
added non-linearity(I) TLBO with added non-
linearity

FIGURE 6.5: Run-length distributions showing the convergence performance for different Simulation conditions

Observing the performance of the algorithms for Figures 6.5a, 6.5b, and 6.5c, all three runs converged to a similar point. However, the TLBO shows the fastest convergence rate with the algorithm converging to the lowest fitness value at 25 iterations. These figures also show that the dynamic FPA converged before 40 iterations and the dynamic GA converging before 35 iterations.

Figures 6.5d, 6.5e and 6.5f show the convergence behaviour when a ramp function is the command input for the three algorithms. These figures show that the lowest fitness value is again, obtained by the TLBO. Figure 6.5f also shows that the TLBO drops to a lower fitness value in comparison to the other algorithm convergence behaviours, which gradually decrease. This further confirms that the TLBO converges faster in this problem instance as well.

Figures 6.5g, 6.5h, and 6.5i show the convergence behaviour of the algorithms with a control system that has added non-linearity. Figure 6.5i again shows that TLBO converges to the lowest fitness value and faster. The dynamic GA does not seem to have converged, as seen in Figure 6.5h as the three runs produced different final behaviours. This is also seen in Figure 6.5g, where the convergence behaviour of run 1 shows that the static FPA may have converged locally in the other two runs.

6.5 Time Complexity

The time complexity analysis begins by observing the measured computational time for the three problem instances using the chosen algorithm design.

Figure 6.6 shows the measured computational time for the three problem instances and algorithms.

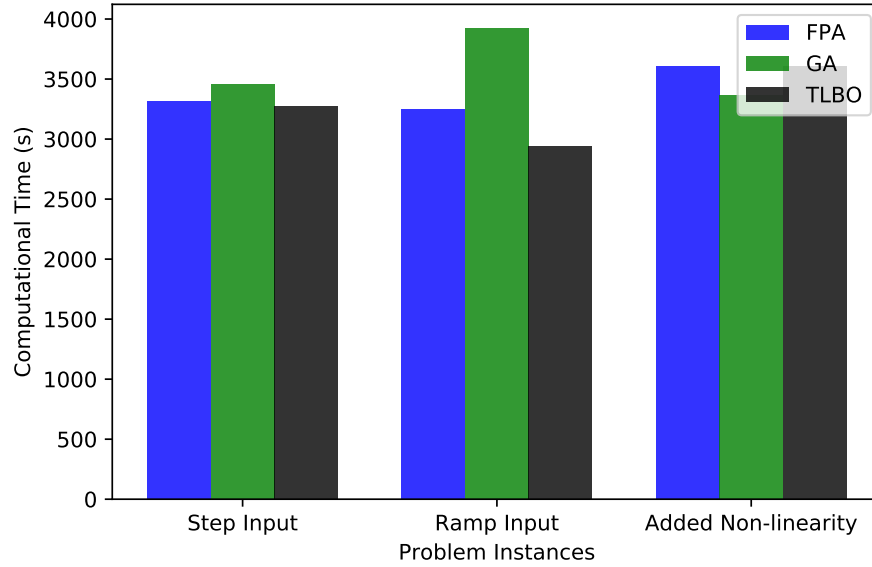


FIGURE 6.6: Bar graph comparing the computational time required for all three runs

The measured computational time shows that the TLBO is the lowest for the first two problem instances. The GA is the lowest in the last problem instance, though the GA is the highest in the first two problem instances. These measured results are dependent on the research environments shown in Section 3.5.5 of Chapter 3, and on how the code was implemented.

The analysis of the complexity of the algorithms was first done by analyzing each step of the algorithm, as shown in their respective pseudo-codes found in Chapter 3, Section 3.2. How the time complexity is analysed is shown in Appendix B. The complexity is then calculated by cascading the individual complexities of the steps.

Table 6.9 shows the result from the time complexity analysis.

TABLE 6.9: Time complexity result for each algorithm analysed in appendix B

Algorithm Type	Complexity Result
TLBO	$O(MR^2)$
FPA	$O(MR^2)$
Static GA	$O(MR \log R)$
Dynamic GA	$O(MR \log R)$

Note that M and R are the values for the iteration number and population size. Also, the complexity of both the static and dynamic FPA is the same. This means that the deterministic calculation of the switch probability does not add to the complexity.

The complexity analysis shows that the complexity of the FPA and TLBO is higher than that of the GA. Thus, the GA has a lower asymptotic bound, which makes it the most favourable when compared to the other algorithms.

6.6 Concluding Remarks

6.6.1 Conclusions

From the sections stated above, the concluding remarks are as follows:

- When observing the algorithm fitness value as a measure of performance, the TLBO performed the best due to this algorithm giving the lowest fitness value for all three problem instances.
- When considering repeatability of the obtained solution at each run as a robustness measure, the TLBO is robust. It is robust because of its ability to conduct exploration and exploitation in one iteration rather than other algorithms which have algorithm-specific parameters that use probability to choose whether exploration or exploitation occurs in each iteration.
- The TLBO algorithm also resulted in the lowest range of fitness values, thus, with the highest probability of achieving quality solutions at each run. This is important for meta-heuristic algorithms since these algorithms contain random seeds, and thus each run is highly more likely to produce a different result.
- When observing the statistical analysis, using a significant level $\alpha = 0.05$, this shows that there is one significant difference in the algorithm performance which is between the TLBO and FPA for the ramp input problem instance. It was found that the TLBO performs significantly better than the FPA in this problem instance. Other than this pair, however, the performance of the algorithms using the chosen parameters, are all the same.
- The TLBO converges to the optimal solution with the smallest number of iterations of 25 compared to the iteration number of 50. This algorithm also resulted in the lowest measured time for both the step and ramp input.
- When observing the time complexity of the algorithms, it was found that the GA resulted in the lesser complexity, as compared to the FPA and TLBO.

6.6.2 Suggestions for Future Practitioners

From these conclusions, some considerations can be drawn for future heuristic algorithm practitioners:

1. Since the TLBO resulted in obtaining the lowest fitness function values and obtained a higher success rate in comparison to the other algorithms, this algorithm must be considered for future experimentation. However, this is with caution as this algorithm also resulted in a higher time complexity result, which means for applications which require large iteration numbers or population size, it will be slow and the results obtained may not be worth the computational effort.

CHAPTER 7

Conclusions and Future Work

In literature and in life we ultimately pursue,
not conclusions, but beginnings.

Sam Tanenhaus
Literature Unbound

7.1 Research Summary

The main aim for the research was to answer two research questions; *"How does applying meta-heuristic algorithms to tune a PID Controller, namely the Flower Pollination Algorithm (FPA), the Genetic Algorithm (GA) and the Teaching-Learning-Based Optimization Algorithm (TLBO), affect the one-axis inertial stabilization system? How do these algorithms compare in performance?"* and *"Do the parameters of the algorithms and the characteristics of the one-axis inertial stabilization system affect the performance? If so, how?"*. The research began by conducting a literature review in Chapter 2, where various aspects of the study were combined and surveyed. The purpose was to propose how to conduct this study, including what to consider, based on the current standing of literature.

Chapter 3 describes the background and preliminaries on the fundamental technical theory and models used. That included a description of the gimbal stabilization system, the algorithms used with the different common and algorithm-specific parameters, and the chosen performance criteria in order to create a strong foundation for the following chapters. The following chapters described the parameter

optimization procedure, the results obtained from each problem instance, and the comparison of the algorithms using some theoretical aspects, each with research questions.

7.2 Research Contributions

Below is a summary of the research contribution. The reader is referred to Section 1.5 in Chapter 1 for more detail.

This research was a comparison study. Thus, the underlying contribution lies in its ability to conduct a comprehensive study of the algorithms on an existing real-life problem to observe how these algorithms behave when given different problem instances. In doing so, questions such as what parameters affect performance for changing problem instances, and whether to use static or dynamic parameters, are then able to be answered.

7.3 Research Conclusions

Below are the research conclusions drawn from the research contributions and results obtained:

- According to the inferential statistical methods used, the significant performance from the algorithms is between the TLBO and FPA for the ramp input problem instance in that the TLBO performed significantly better than the FPA. Other than this, there was no significant difference in the performance of the three algorithms. However, the TLBO resulted in the lowest fitness value and reached the near-optimal solution with the lowest iteration number.
- Since there is no significant difference in the algorithm performances (except for that one instance), this may indicate that all algorithms have reached the

‘optimal’ solution and that the reason one cannot obtain a better solution is because of constraints that lie in the control system itself.

- The common algorithms do have an effect on the performance of the system. It was found that increasing the upper bound of the search space results in a performance with a high overshoot for the simple control system design.
- The common parameters of the algorithms are not robust. This was seen when changing the parameters of the control system resulting in an unsatisfactory response.
- Lastly, the insignificant difference in the performance of the algorithms indicates that there is no particular better algorithm class (i.e. no better between SI or EA). However, the performance of (any) algorithm highly depends on how it is tuned, and the problem it is solving.

7.4 Possible Future Work

There are different avenues that need further investigation. How these algorithms perform for a two or three-axis gimbal stabilization system is crucial for future work. It is crucial because more often than not, an inertial stabilization platform requires two or more gimbal axes. Thus, exploring this would be beneficial. However, there are complexities which two-axis systems introduce, such as coupling, and observing how these algorithms perform in this setting is essential.

On this note, there are other non-linearity which could be explored in this type of application. More of these are illustrated in Figure 9 of [12], which include structural flexibility like bending, and actuator and gear reactions.

This work could be expanded by introducing other methods for adaptively tuning the algorithm parameters, whether common or algorithm-specific. The burden of choosing what values to use for the parameters can be a cumbersome and unnecessary burden when the main point is to solve the problem at hand. The

studies in [142] and [143] move towards implementing parameterless GA to solve an optimization problem, and perhaps that is the way forward.

The fitness function was an important aspect of this study as it quantifies the performance. The effect of the weight value chosen for the different objectives for this study was evident. There are two ways of combining multiple and conflicting objectives for optimization; using a weighted cost function (such as the one used in this study) or making use of a Pareto front. When plotting two conflicting objectives against one another and observing the fitness value, the Pareto front is the set of points on this graph, where one objective cannot be improved without sacrificing the other. This then would give the weight values of the fitness function rather than making assumptions on what the weight values should be. This may improve the fitness function and thus, the overall performance on the system.

Stiction plays a significant role in the initial response of the system rather than in the long term. Inserting sinusoidal command in the control system to observe how the algorithms perform with a constant change in the direction would be of interest, as this input represents a constant change in direction which is when stiction is more important to observe.

There are other factors which can and should be considered in future work, which were not considered in this work. These factors include modelling the gimbal system with additional non-linearity, which brings it closer to reality, such as adding an integral wind-up algorithm for the integral controller. Other factors also include adding the frequency domain response as a performance measure rather than solely looking at the time domain response for the control system, and observing space-complexity rather than only looking at the time-complexity of the algorithms.

Appendix A

Model Validation

This section aims to describe the method used to ensure that the model used in this study is the same as the one adapted from [5] for validation purposes. This will be done by comparing the step response result obtained for the simple one-axis control system, using the elements and models stipulated in Section 3.1.

Figure A.1 shows the step response obtained on the study in [5] and Figure A.2 shows the simulated solution.

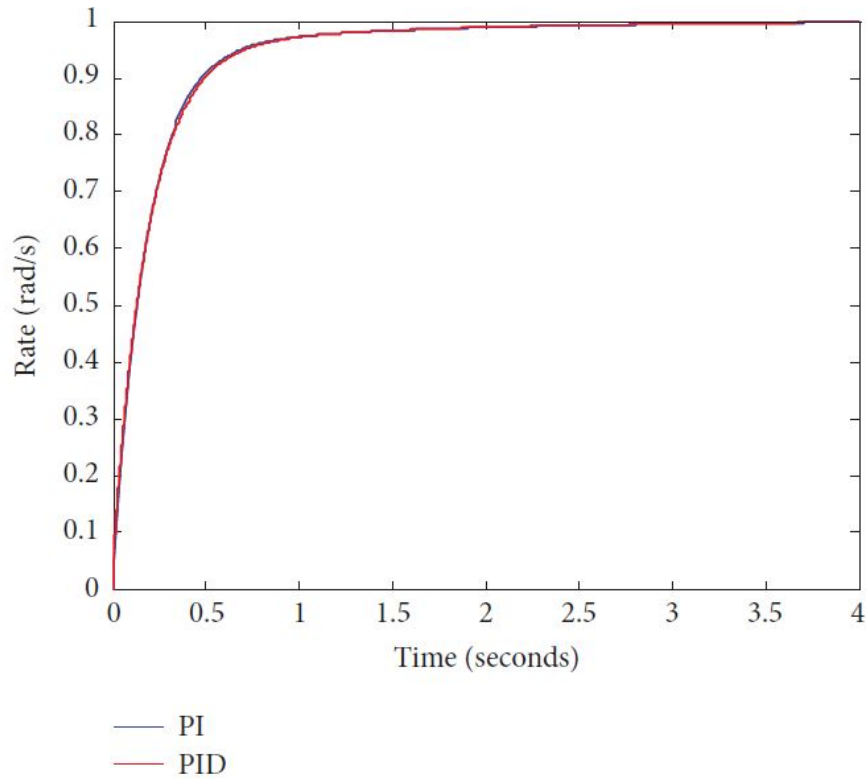


FIGURE A.1: Diagram showing the results for the step response system shown in Figure 11 of [5]

The study in [5] was comparing two different controllers (i.e. the PI and the PID controller) as seen in Figure A.1. The performance behaviour is quite similar; however, the difference that was stipulated in the paper, is the rise time.

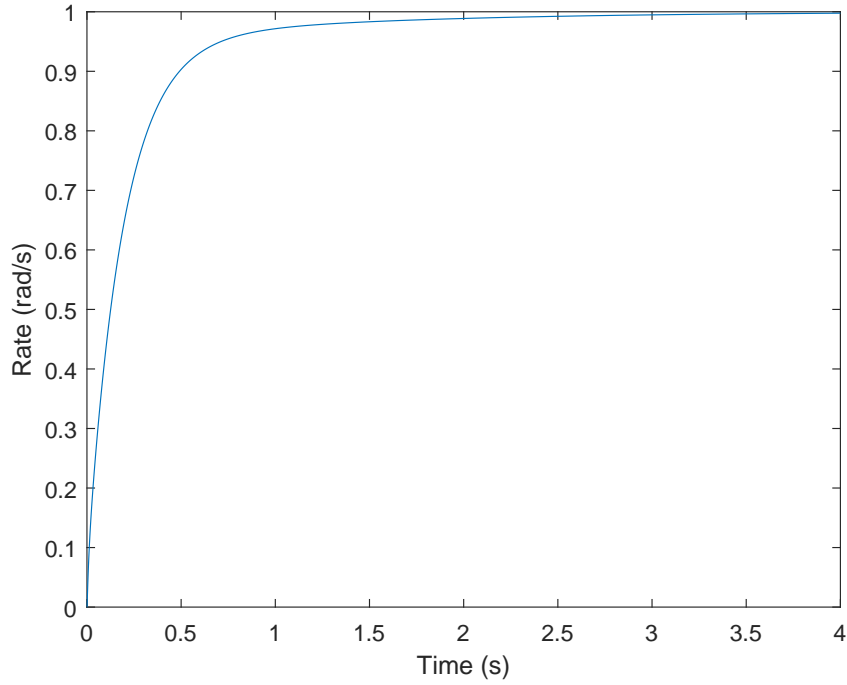


FIGURE A.2: Diagram showing the simulated results for the step response of the system from the model

Table A.1 compares the rise time from the paper for both the PI and PID controllers with the rise time obtained from this simulation, where a PID controller was used.

TABLE A.1: Showing the maximum, minimum and average fitness value at each iteration value.

Taken from	Rise Time (s)
Journal Paper PI	0.456
Journal Paper PID	0.485
Simulink Simu- lation (PID)	0.4753
Transfer Func- tion Simulation (PID)	0.4826

The results are very similar when comparing the PID controller response from the journal paper, with the PID controller response from the simulations conducted in this research; however, there is a slight difference. The journal paper presented a Simulink model together with the deduced transfer function of the model. The result obtained from using the transfer function in the simulation of this research is most similar with the journal paper result. This may indicate that the journal paper obtained results from using the transfer function. However, the importance of using Simulink for simulation lies in the ability of Simulink to model nonlinearities to which transfer functions are not able to. If the journal paper study did make use of the Simulink model rather than the transfer function, then the difference in the rise time performance may be due to the difference in Simulink solvers used or whether the filter coefficient was used and the difference in its value, if it was. These two parameters are not stipulated in the journal paper and thus can contribute to the difference in results. Nonetheless, the result obtained is satisfactory, and thus, the control system model has been validated and was used in this study.

Appendix B

Time Complexity Analysis

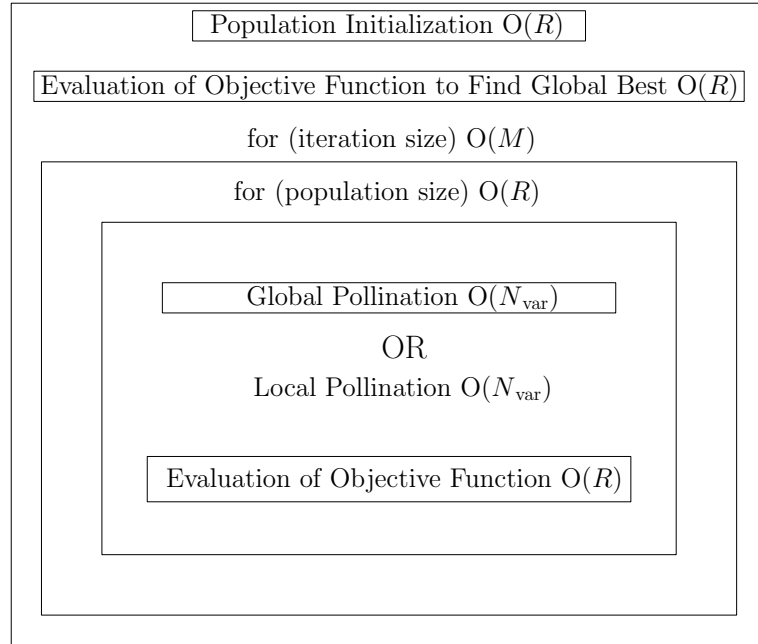


FIGURE B.1: Time complexity of FPA

$$O(R + M(R(N_{\text{var}} + R))) = O(MR^2) \quad (\text{B.1})$$

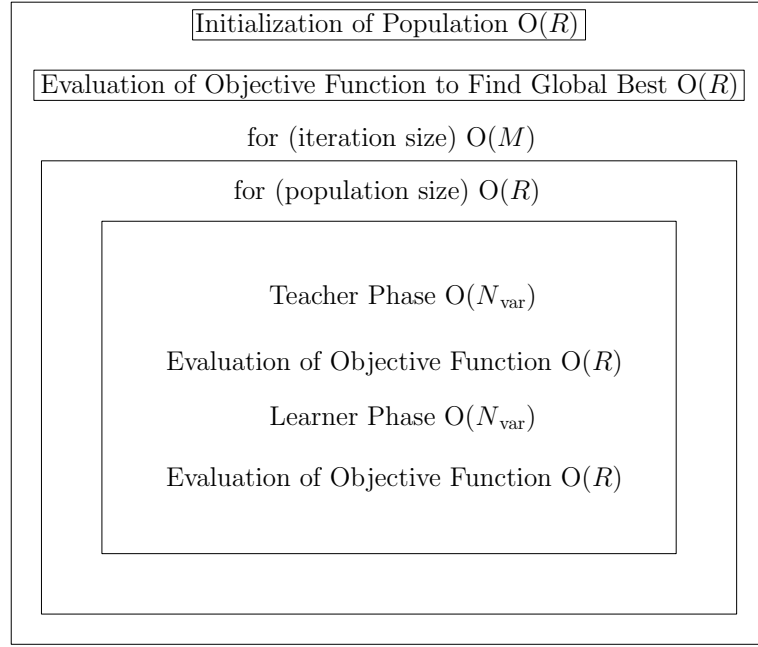


FIGURE B.2: Time complexity of TLBO

$$O(R + R + M(R(N_{\text{var}} + RN_{\text{var}} + R))) = O(MR^2) \quad (\text{B.2})$$

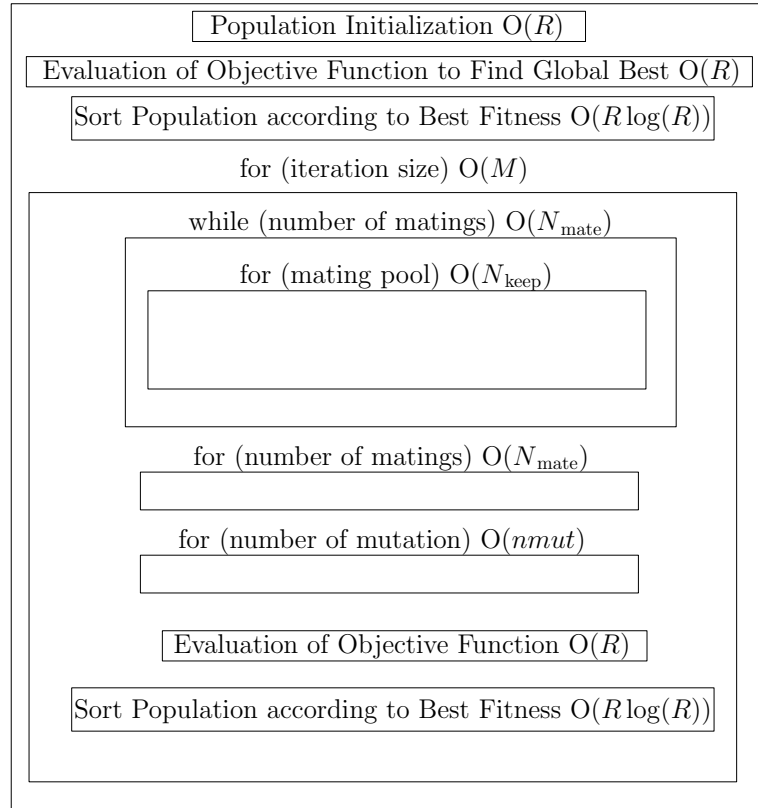


FIGURE B.3: Time complexity of Static GA

$$O(R + R + M(N_{\text{mate}}N_{\text{keep}} + N_{\text{mate}} + nmut + R + R \log R) = O(MR \log R) \quad (\text{B.3})$$

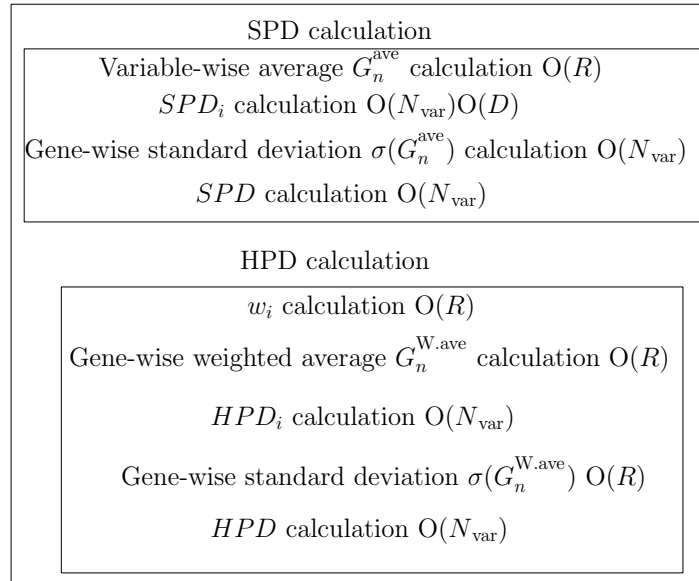
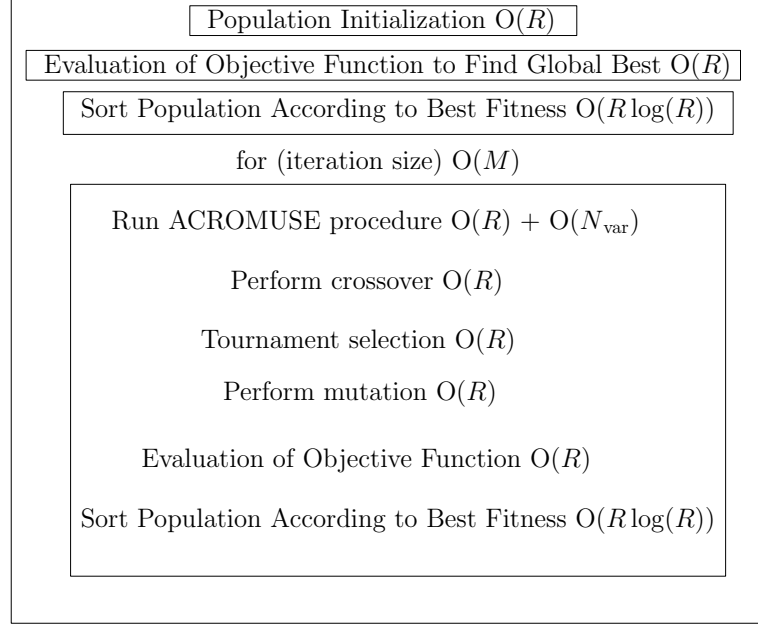


FIGURE B.4: Time complexity of Dynamic GA

$$O(R + R + R \log R + M(R + N_{\text{var}} + R + R + R + R \log R)) = O(MR \log R) \quad (\text{B.4})$$

References

- [1] “R-3 Series Short-Range Air-to-Air Missile | Military-Today.com.” [Online]. Available: <http://www.military-today.com/missiles/r3.htm>
- [2] “Parts for the Hubble Space Telescope.” [Online]. Available: <https://www.darpa.mil/about-us/dar/hubble>
- [3] “ATMOS 2000,” Version ID: 908685875, Jul. 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=ATMOS_2000&oldid=908685875.
- [4] A. E. Eiben and S. K. Smit, “Parameter tuning for configuring and analyzing evolutionary algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, Mar. 2011.
- [5] R. Jia, V. Nandikolla, G. Haggart, C. Volk, and D. Tazartes, “System performance of an inertially stabilized gimbal platform with friction, resonance, and vibration effects,” *Journal of Nonlinear Dynamics*, vol. 2017, pp. 1–20, 2017, doi: <https://doi.org/10.1155/2017/6594861>.
- [6] N. Layshot and Xiao-Hua Yu, “Modeling of a gyro-stabilized helicopter camera system using artificial neural networks,” in *2011 IEEE International Conference on Information and Automation*, 2011, pp. 454–458.
- [7] K. A. Tehrani and A. Mpanda, “Pid control theory,” in *Introduction to PID Controllers*, R. C. Panda, Ed. Rijeka: IntechOpen, 2012, ch. 9, pp. 213–228. [Online]. Available: <https://doi.org/10.5772/34364>.
- [8] L. Marton and B. Lantos, “Modeling, identification, and compensation of stick-slip friction,” *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 511–521, Feb. 2007.

- [9] B. McGinley, J. Maher, C. O’Riordan, and F. Morgan, “Maintaining healthy population diversity using adaptive crossover, mutation, and selection,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 692–714, 2011.
- [10] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 9th ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2001.
- [11] M. Abdo, A. R. Vali, A. R. Toloei, and M. R. Arvan, “Modeling control and simulation of two axes gimbal seeker using fuzzy PID controller,” in *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*. Tehran, Iran: IEEE, May 2014, pp. 1342–1347.
- [12] J. M. Hilkert, “Inertially stabilized platform technology concepts and principles,” *IEEE Control Systems Magazine*, vol. 28, no. 1, pp. 26–46, Feb. 2008.
- [13] M. Abdo, A. R. Vali, A. Toloei, and M. R. Arvan, “Research on the cross-coupling of a two axes gimbal system with dynamic unbalance,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 10, Oct. 2013, doi: <https://doi.org/10.5772/56963>.
- [14] M. M. Abdo, A. R. Vali, A. R. Toloei, and M. R. Arvan, “Stabilization loop of a two axes gimbal system using self-tuning PID type fuzzy controller,” *ISA Transactions*, vol. 53, no. 2, pp. 591–602, Mar. 2014.
- [15] J. M. S. Ribeiro, M. F. Santos, M. J. Carmo, and M. F. Silva, “Comparison of PID controller tuning methods: Analytical/classical techniques versus optimization algorithms,” in *2017 18th International Carpathian Control Conference (ICCC)*, May 2017, pp. 533–538.
- [16] V. Chopra, S. K. Singla, and L. Dewan, “Comparative analysis of tuning a pid controller using intelligent methods,” *Acta Polytechnica Hungarica*, vol. 11, no. 8, pp. 235–249, 2014.

- [17] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, “Metaheuristic algorithms in modeling and optimization,” in *Metaheuristic Applications in Structures and Infrastructures*, A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, Eds. Oxford: Elsevier, 2013, pp. 1–24.
- [18] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, 2nd ed. Berlin, Heidelberg: Springer-Verlag, 2007.
- [19] A. Memari, R. Ahmad, and A. R. A. Rahim, “Metaheuristic algorithms: guidelines for implementation,” *Journal of Soft Computing and Decision Support Systems*, vol. 4, no. 7, pp. 1–6, 2017.
- [20] J. N. Hooker, “Testing heuristics: We have it all wrong,” *Journal of Heuristics*, vol. 1, no. 1, pp. 33–42, Sep. 1995.
- [21] K. Ogata, *Modern Control Engineering*, 5th ed. Boston: Prentice-Hall, 2010.
- [22] S. Seshan and A. Seshan, “Using the gyro sensor and dealing with drift.” [Online]. Available: <https://stemrobotics.cs.pdx.edu/sites/default/files/Gyro.pdf>.
- [23] S. Li, Y. Gao, G. Meng, G. Wang, and L. Guan, “Accelerometer-based gyroscope drift compensation approach in a dual-axial stabilization platform,” *Electronics*, vol. 8, no. 5, May 2019.
- [24] E. A. Carmona, “Development of active camera stabilization system for implementation on uav’s.” [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.939.6007&rep=rep1&type=pdf>.
- [25] R. J. Rajesh and P. Kavitha, “Camera gimbal stabilization using conventional PID controller and evolutionary algorithms,” in *2015 International Conference on Computer, Communication and Control (IC4)*. Indore: IEEE, Sep. 2015, doi: 10.1109/IC4.2015.7375580.

- [26] M. Baskin and K. Leblebicioğlu, “Robust control for line-of-sight stabilization of a two-axis gimbal system,” *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 25, pp. 3839–3853, 2017.
- [27] V. Sangveraphunsiri and K. Malithong, “Control of inertial stabilization systems using robust inverse dynamics control and sliding mode control,” in *The 6th International Conference on Automotive Engineering (ICAE-6)*, BITEC, Bangkok, Thailand, Mar 29, 2010, p. 8.
- [28] H.-P. Lee and I.-E. Yoo, “Robust control design for a two-axis gimbaled stabilization system,” in *2008 IEEE Aerospace Conference*, 1 March 2008, doi: 10.1109/AERO.2008.4526568.
- [29] B. W. Bujela, “Investigation into the robust modelling, control and simulation of a two-dof gimbal platform for airborne applications,” MSc thesis, University of the Witwatersrand, Johannesburg, SA, 2013.
- [30] K. Kawada, T. Shiino, T. Yamamoto, M. Komichi, and T. Nishioka, “Data-Driven PD Gimbal Control,” in *2008 International Conference on Computational Intelligence for Modelling Control Automation*, Dec. 2008, pp. 993–998.
- [31] X. Zhou, C. Yang, and T. Cai, “A model reference adaptive control/PID compound scheme on disturbance rejection for an aerial inertially stabilized platform,” *Journal of Sensors*, vol. 2016, 2016, doi: 10.1155/2016/7964727.
- [32] I. D. Landau, R. Lozano, and M. M’Saad, “Introduction to adaptive control,” in *Adaptive Control*, ser. Communications and Control Engineering, I. D. Landau, R. Lozano, and M. M’Saad, Eds. Springer: London, 1998, pp. 1–30.
- [33] M. Khayatian and P. K. Aghaei, “Adaptive control of a two-axis gimbal system using modified error,” in *The 3rd International Conference on Control, Instrumentation and Automation*, Tehran, Iran, Dec. 28, 2013, pp. 1–5.

- [34] R. Caponetto and M. G. Xibilia, “Fractional order PI control of a gimbal platform,” in *2017 European Conference on Circuit Theory and Design (ECCTD)*, Catania, Italy, Sep. 4, 2017, pp. 1–4.
- [35] X. Zhou, B. Zhao, and G. Gong, “Control parameters optimization based on co-simulation of a mechatronic system for an UA-based two-axis inertially stabilized platform,” *Sensors*, vol. 15, no. 8, pp. 20 169–20 192, Aug. 2015.
- [36] B. Singh and N. Joshi, “Tuning techniques of PID controller: A review,” *International Journal on Emerging Technologies*, vol. 8, no. 1, pp. 481–485, 2017.
- [37] F. Haugen, “Basic Dynamics and Control,” Skien, Norway: TechTeach, 2010.
- [38] O. Garpinger, “Analysis and design of Software-Based Optimal PID Controllers,” PhD thesis, Lund University, Lund, Sweden, 2015.
- [39] F. T. Asal and M. Coggun, “EE 402 Discrete Time Systems Project Report PI , PD , PID Controllers,” 2012. [Online]. Available: <https://pdfs.semanticscholar.org/d149/c7ad9deba6d0f3a7a615a48fbd11fc16a1a8.pdf>.
- [40] P. Dash, L. C. Saikia, and N. Sinha, “Flower pollination algorithm optimized PI-PD cascade controller in automatic generation control of a multi-area power system,” *International Journal of Electrical Power & Energy Systems*, vol. 82, pp. 19–28, Nov. 2016.
- [41] M. A. Johnson and M. H. Moradi, Eds., *PID Control: New Identification and Design Methods*. London: Springer-Verlag, 2005.
- [42] J. H. Chow, F. F. Wu, and J. A. Momoh, Eds., *Applied Mathematics for Restructured Electric Power Systems: Optimization, Control, and Computational Intelligence*. NY, USA: Springer, 2005.
- [43] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, N.J: John Wiley & Sons, 2009.

- [44] H. Gozde, M. Cengiz Taplamacioglu, and İ. Kocaarslan, “Comparative performance analysis of Artificial Bee Colony algorithm in automatic generation control for interconnected reheat thermal power system,” *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 167–178, Nov. 2012.
- [45] M. G. Villarreal-Cervantes and J. Alvarez-Gallegos, “Off-line PID control tuning for a planar parallel robot using DE variants,” *Expert Systems with Applications*, vol. 64, pp. 444–454, Dec. 2016.
- [46] K. Jagatheesan, B. Anand, S. Samanta, N. Dey, V. Santhi, A. S. Ashour, and V. E. Balas, “Application of flower pollination algorithm in load frequency control of multi-area interconnected power system with nonlinearity,” *Neural Computing and Applications*, vol. 28, no. 1, pp. 475–488, Dec. 2017.
- [47] D. Fister, I. Fister Jr, I. Fister, and R. Šafarič, “Parameter tuning of PID controller with reactive nature-inspired algorithms,” *Robotics and Autonomous Systems*, vol. 84, pp. 64–75, Oct. 2016.
- [48] M. M. Sabir and J. A. Khan, “Optimal design of PID controller for the speed control of DC motor by using metaheuristic techniques,” *Advances in Artificial Neural Systems*, vol. 2014, pp. 1–8, 2014, doi: 10.1155/2014/126317.
- [49] A. Y. Jaen-Cuellar, R. de J. Romero-Troncoso, L. Morales-Velazquez, and R. A. Osornio-Rios, “PID-controller tuning optimization with genetic algorithms in servo systems,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 9, p. 324, Sep. 2013, doi: 10.5772/56697.
- [50] W. Huang and H. N. Lam, “Using genetic algorithms to optimize controller parameters for HVAC systems,” *Energy and Buildings*, vol. 26, no. 3, pp. 277–282, 1997.
- [51] V. Rajinikanth and S. C. Satapathy, “Design of controller for automatic voltage regulator using teaching learning based optimization,” *Procedia Technology*, vol. 21, pp. 295–302, Jan. 2015.

- [52] C. Fonlupt, D. Robilliard, P. Preux, and E.-G. Talbi, “Fitness landscapes and performance of meta-heuristics,” in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voß, S. Martello, I. H. Osman, and C. Roucairol, Eds. Boston, MA: Springer, 1999, pp. 257–268. [Online]. Available: https://doi.org/10.1007/978-1-4615-5775-3_18.
- [53] T. J. Krüger, “Development, implementation and theoretical analysis of the bee colony optimization meta-heuristic method,” Dissertation, Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia, 2017.
- [54] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine Learning*, vol. 3, no. 2, pp. 95–99, Oct. 1988.
- [55] D. Shiffman, “The nature of code,” 2012. [Online]. Available: <http://wtf.tw/ref/shiffman.pdf>.
- [56] S. Das and P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
- [57] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Oct. 1995, pp. 39–43.
- [58] M. Mitchell, “Genetic algorithms: An overview,” *Complexity*, vol. 1, no. 1, pp. 31–39, Sep. 1995.
- [59] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, 2nd ed. Hoboken, N.J: John Wiley and Sons, Inc., 2004.
- [60] X.-S. Yang, “Flower pollination algorithm for global optimization,” in *Unconventional Computation and Natural Computation*, J. Durand-Lose and N. Jonoska, Eds., vol. 7445. Berlin, Heidelberg: Springer, 2012, pp. 240–249.

- [61] Z. A. A. Alyasseri, A. T. Khader, M. A. Al-Betar, M. A. Awadallah, and X.-S. Yang, “Variants of the flower pollination algorithm: A review,” in *Nature-Inspired Algorithms and Applied Optimization*, X.-S. Yang, Ed. Cham: Springer International Publishing, 2018, vol. 744, pp. 91–118.
- [62] X. He, X.-S. Yang, M. Karamanoglu, and Y. Zhao, “Global convergence analysis of the flower pollination algorithm: A discrete-time markov chain approach,” *Procedia Computer Science*, vol. 108, pp. 1354–1363, 2017.
- [63] R. V. Rao, *Teaching-Learning-Based Optimization and Its Engineering Applications*. Cham, Switzerland: Springer, 2016.
- [64] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
- [65] D. H. Wolpert and W. G. Macready, “Coevolutionary free lunches,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, Dec. 2005.
- [66] C. L. Karr and E. Wilson, “A self-tuning evolutionary algorithm applied to an inverse partial differential equation,” *Applied Intelligence*, vol. 19, no. 3, pp. 147–155, Nov. 2003.
- [67] A. E. Eiben, R. Hinterding, and Z. Michalewicz, “Parameter control in evolutionary algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, Jul. 1999.
- [68] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, “Parameter control in evolutionary algorithms: Trends and challenges,” *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 167–187, Apr. 2015.
- [69] H. Gozde, M. Cengiz Taplamacioglu, and İ. Kocaarslan, “Comparative performance analysis of Artificial Bee Colony algorithm in automatic generation

- control for interconnected reheat thermal power system,” *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 167–178, Nov. 2012.
- [70] W. Huang and H. N. Lam, “Using genetic algorithms to optimize controller parameters for HVAC systems,” *Energy and Buildings*, vol. 26, no. 3, pp. 277–282, Jan. 1997.
- [71] T. Bäck, “Optimal mutation rates in genetic search,” in *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 2–8.
- [72] E. Goldberg, “Toward a better understanding of mixing in genetic algorithms,” *J. SICE*, vol. 32, no. 1, pp. 10–14, 1993.
- [73] J. D. Schaffer and A. Morishima, “An adaptive crossover distribution mechanism for genetic algorithms,” in *Proceedings of the Second International Conference on Genetic Algorithms and Their Application*. J. J. Grefenstette, Ed. Hillsdale, NJ: Lawrence Erlbaum Associates Inc., 1987, p. 36–40.
- [74] G. Lin and G. Liu, “Tuning PID controller using adaptive genetic algorithms,” in *2010 5th International Conference on Computer Science Education*, Aug. 2010, pp. 519–523.
- [75] T. Park and K. R. Ryu, “A dual-population genetic algorithm for adaptive diversity control,” *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 865–884, Dec. 2010.
- [76] X.-S. Yang, “Metaheuristic optimization: Nature-inspired algorithms and applications,” in *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, X.-S. Yang, Ed. Berlin, Heidelberg: Springer, 2013, vol. 427, pp. 405–420.

- [77] F. B. Ozsoydan and A. Baykasoglu, “Analysing the effects of various switching probability characteristics in flower pollination algorithm for solving unconstrained function minimization problems,” *Neural Computing and Applications*, vol. 31, no. 11, pp. 7805–7819, Nov. 2019.
- [78] W. Li, Z. He, J. Zheng, and Z. Hu, “Improved flower pollination algorithm and its application in user identification across social networks,” *IEEE Access*, vol. 7, pp. 44 359–44 371, 2019.
- [79] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu, “A framework for self-tuning optimization algorithm,” *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2051–2057, Dec. 2013.
- [80] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [81] D. E. Goldberg, K. Deb, and J. H. Clark, “Genetic algorithms, noise, and the sizing of populations,” *Complex Systems*, vol. 6, pp. 333–362, 1992.
- [82] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller, “The gambler’s ruin problem, genetic algorithms, and the sizing of populations,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, Apr. 1997, pp. 7–12.
- [83] D. Thierens, “Dimensional analysis of allele-wise mixing revisited,” in *Parallel Problem Solving from Nature — PPSN IV*, ser. Lecture Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Heidelberg, Berlin: Springer, 1996, pp. 255–265.
- [84] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Heidelberg: Springer-Verlag, 1996.
- [85] P. Bujok, J. Tvrdík, and R. Poláková, “Comparison of nature-inspired population-based algorithms on continuous optimisation problems,” *Swarm and Evolutionary Computation*, vol. 50, Jan. 2019.

- [86] E. Ridge, “Design of experiments for the tuning of optimisation algorithms,” PhD thesis, Dept. of Computer Science, University of York, York, UK, 2007, doi: <https://doi.org/10.1016/j.swevo.2019.01.006>.
- [87] M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, Eds., *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Providence, Rhode Island: American Mathematical Society, Dec. 2002, vol. 59.
- [88] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, 1st ed. Amsterdam, Boston: Elsevier, 2014.
- [89] K. Socha, “The influence of run-time limits on choosing ant system parameters,” in *Genetic and Evolutionary Computation — GECCO 2003*, ser. Lecture Notes in Computer Science, E. Cantú-Paz, J. A. Foster, K. Deb, L. D. Davis, R. Roy, and O’Reilly, Eds. Berlin, Heidelberg: Springer, 2003, pp. 49–60.
- [90] N. Zlobinsky and L. Cheng, “SAM: a meta-heuristic algorithm for single machine scheduling problems,” *SAIEE Africa Research Journal*, vol. 109, pp. 58 – 68, Mar. 2018. [Online]. Available: http://www.scielo.org.za/scielo.php?script=sci_arttext&pid=S1991-16962018000100006&nrm=iso.
- [91] C.-C. Wong, S.-A. Li, and H.-Y. Wang, “Optimal PID controller design for AVR System,” *Tamkang Journal of Science and Engineering*, vol. 12, no. 3, pp. 259–270, 2009.
- [92] X.-S. Yang, M. Karamanoglu, and X. He, “Flower pollination algorithm: A novel approach for multiobjective optimization,” *Engineering Optimization*, vol. 46, no. 9, pp. 1222–1237, Sep. 2014.
- [93] Z. L. Gaing, “A particle swarm optimization approach for optimum design of PID controller in AVR system,” *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 384–391, Jun. 2004.

- [94] F. Pan and L. Liu, “Research on different integral performance indices applied on fractional-order systems,” in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 324–328.
- [95] C. C. McGeoch, “Feature Article - Toward an experimental method for algorithm simulation,” *INFORMS Journal on Computing*, vol. 8, no. 1, pp. 1–15, Feb. 1996.
- [96] M. Birattari and M. Dorigo, “How to assess and report the performance of a stochastic algorithm on a benchmark problem: *mean* or *best* result on a number of runs?” *Optimization Letters*, vol. 1, no. 3, pp. 309–311, May 2007.
- [97] A. E. Eiben and M. Jelasity, “A critical note on experimental research methodology in EC,” in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02*, vol. 1. Honolulu, HI, USA: IEEE, May 12 2002, pp. 582–587.
- [98] R. J. Rajesh and C. M. Ananda, “PSO tuned PID controller for controlling camera position in UAV using 2-axis gimbal,” in *2015 International Conference on Power and Advanced Control Engineering (ICPACE)*. Bengaluru, India: IEEE, Aug. 2015, pp. 128–133.
- [99] P. B. Jackson, “Overview of missile flight control systems,” *Johns Hopkins APL Technical Digest*, vol. 29, no. 1, 2010.
- [100] A. Singh, S. Chatterjee, and R. Thakur, “Design of tracking of moving target using PID controller,” *International Journal of Engineering Trends and Technology*, vol. 15, no. 8, pp. 403–406, Sep. 2014.
- [101] M. Chiarandini, L. Paquete, M. Preuss, and E. Ridge, “Experiments on metaheuristics: methodological overview and open issues.” [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.207.4291&rep=rep1&type=pdf>.

- [102] T. Davidović and T. Jakšić Krüger, “Convergence analysis of swarm intelligence metaheuristic methods,” in *International Conference on Optimization Problems and Their Applications*, Jun. 2018, pp. 251–266.
- [103] J. Chen, J. Ni, and M. Hua, “Convergence analysis of a class of computational intelligence approaches,” *Mathematical Problems in Engineering*, vol. 2013, 2013, doi: 10.1155/2013/409606.
- [104] T. J. Krüger and T. Davidović, “Model convergence properties of the constructive bee colony optimization algorithm,” in *XLI Simpozijum o operacionim istrazivanjima*, Sept. 16, 2014, pp. 340–345.
- [105] J. He and L. Kang, “On the convergence rates of genetic algorithms,” *Theoretical Computer Science*, vol. 229, no. 1-2, pp. 23–39, Nov. 1999.
- [106] G. Rudolph, “Convergence analysis of canonical genetic algorithms,” *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, Jan. 1994.
- [107] M. J. Mahmoodabadi and R. Ostadzadeh, “CTLBO: Converged teaching–learning–based optimization,” *Cogent Engineering*, vol. 6, no. 1, Aug. 2019, doi: 10.1080/23311916.2019.1654207.
- [108] J. K. Pickard, J. A. Carretero, and V. C. Bhavsar, “On the convergence and origin bias of the Teaching-Learning-Based-Optimization algorithm,” *Applied Soft Computing*, vol. 46, pp. 115–127, Sep. 2016.
- [109] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, Mar. 2011.
- [110] A. K. Qin and P. N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” in *2005 IEEE Congress on Evolutionary Computation*, vol. 2, Sep. 2005, pp. 1785–1791.

- [111] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.
- [112] M. M. Alhato and S. Bouallègue, "Direct power control optimization for doubly fed induction generator based wind turbine systems," *Mathematical and Computational Applications*, vol. 24, no. 3, Aug. 2019, doi 10.3390/mca24030077.
- [113] S. Mohammed, K. M. Fayçel, and B. B. Rochdi, "Statistical analysis of harmony search algorithms in tuning PID controller," *International Journal of Intelligent Engineering and Systems*, vol. 9, no. 4, pp. 98–106, Dec. 2016.
- [114] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.
- [115] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, Mar. 1940.
- [116] D. Zindros, "A Gentle Introduction to Algorithm Complexity Analysis," <https://discrete.gr/complexity/>.
- [117] J. P. Gibson, "MSP: Mathematical foundations - MAT7003/Introduction.1 MAT 7003 : mathematical foundations (for software engineering)," <https://slideplayer.com/slide/2529946/>, 2012.
- [118] A. Maniuk, "Introduction in big-O notation," Software Development, June 5, 2019. [Online]. Available: <https://www.maniuk.net/2019/06/introduction-in-big-o-notation.html>.
- [119] D. Sudholt and C. Thyssen, "Running time analysis of Ant Colony Optimization for shortest path problems," *Journal of Discrete Algorithms*, vol. 10, pp. 165–180, Jan. 2012.

- [120] R. Strange, A. Y. Yang, and L. Cheng, “Discrete flower pollination algorithm for solving the symmetric travelling salesman problem,” in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, Xiamen, China, Dec. 6, 2019.
- [121] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart Jr., “Designing and reporting on computational experiments with heuristic methods,” *Journal of Heuristics*, vol. 1, no. 1, pp. 9–32, Sep. 1995.
- [122] M. M. Amini and M. Racer, “A rigorous computational comparison of alternative solution methods for the generalized assignment problem,” *Management Science*, vol. 40, no. 7, pp. 868–890, July 1994.
- [123] A. A. Trusov, “Overview of MEMS gyroscopes: History, principles of operations, types of measurements,” 2011. [Online]. Available: <http://alexandertrusov.com/uploads/pdf/2011-UCI-trusov-whitepaper-gyros.pdf>.
- [124] A. M. Shkel, “Type I and Type II micromachined vibratory gyroscopes,” in *2006 IEEE/ION Position, Location, And Navigation Symposium*, Apr. 2006, pp. 586–593.
- [125] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella, “Gyroscope technology and applications: a review in the industrial perspective,” *Sensors*, vol. 17, no. 10, Oct. 2017, doi 10.3390/s17102284.
- [126] Cineflex, “Cineflex media: Stabilized broadcast & ENG camera system.” [Online]. Available: <http://pdf.directindustry.com/pdf/axsys-technologies/cineflex-media/36088-395321.html#open>.
- [127] D. Lakshmi, A. P. Fathima, and R. Muthu, “A novel flower pollination algorithm to solve load frequency control for a hydro-thermal deregulated power system,” *Circuits and Systems*, vol. 7, no. 4, pp. 166–178, Apr. 2016.
- [128] H. Yang, Y. Zhao, M. Li, and F. Wu, “The static unbalance analysis and its measurement system for gimbals axes of an inertial stabilization platform,” *Metrology and Measurement Systems*, vol. 22, no. 1, pp. 51–68, Mar. 2015.

- [129] J. Johnson and D. Priser, “Vibration levels in army helicopters - measurement and recommendations and data,” U.S. Army Aeromedical Research Laboratory, Fort Rucker, Alabama 36362, USAARL Report 81-5, Sep. 1981.
- [130] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Y. Leong, “Crossover and mutation operators of genetic algorithms,” *International Journal of Machine Learning and Computing*, vol. 7, no. 1, pp. 9–12, Feb. 2017.
- [131] K. Q. Zhu, “A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows,” in *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, Nov. 2003, pp. 176–183.
- [132] R. K. Ursem, “Diversity-guided evolutionary algorithms,” in *International Conference on Parallel Problem Solving from Nature — PPSN VII*, ser. Lecture Notes in Computer Science, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañás, Eds. Heidelberg, Berlin: Springer, 2002, pp. 462–471.
- [133] T. Bäck and F. Hoffmeister, “Extended selection mechanisms in genetic algorithms,” in *4th International Conference on Genetic Algorithms*, 1991, pp. 92–99.
- [134] D. Curran, C. O’Riordan, and H. Sorensen, “The effects of lifetime learning on the diversity and fitness of populations,” in *GECCO ’07 Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, July 2007, p. 337.
- [135] Diffen, “Cross-Pollination vs Self-Pollination.” [Online]. Available: https://www.diffen.com/difference/Cross_Pollination_vs_Self_Pollination.
- [136] T. Kathuria, A. Gupta, J. Kumar, V. Kumar, and K. P. S. Rana, “Study of optimization methods for tuning of PID gains for three link manipulator,” in *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*. Noida, India: IEEE, Jan. 2017, pp. 99–104.

- [137] A. R. Vali, M. Abdo, and M. R. Arvan, "Modeling, control and simulation of cascade control servo system for one-axis gimbal mechanism," *International Journal of Engineering*, vol. 27, no. 1, Jan. 2014.
- [138] G. Cardillo, "MyFriedman: Friedman test for non parametric two way ANalysis Of VAriance," 2009. [Online]. Available: <https://kr.mathworks.com/matlabcentral/fileexchange/25882-myfriedman?focused=6e2d1502-b544-76e1-0ef3-141808fb0823>.
- [139] W. J. Conover, *Practical Nonparametric Statistics*, 3rd ed., ser. Wiley Series in Probability and Statistics. Applied Probability and Statistics Section. New York: Wiley, 1999.
- [140] T. Bartz-Beielstein and S. Markon, "Tuning search algorithms for real-world applications: A regression tree based approach," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. Portland, OR, USA: IEEE, 2004, pp. 1111–1118.
- [141] Aerial Camera Systems, "Cineflex HD V14." [Online]. Available: https://www.aerialcamerasystems.com/_resource/_pdf/HD-Cineflex-V14.pdf.
- [142] G. R. Harik and F. G. Lobo, "A parameter-less genetic algorithm," in *GECCO'99. Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, Jul. 1999, pp. 258–265.
- [143] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart, "An empirical study on GAs "without parameters"," in *International Conference in Parallel Problem Solving from Nature*, 2000, pp. 315–324.