

3-21-2008

An Approach to Designing an Unmanned Helicopter Autopilot Using Genetic Algorithms and Simulated Annealing

Namir Aldawoodi
University of South Florida

Follow this and additional works at: <https://scholarcommons.usf.edu/etd>

 Part of the [American Studies Commons](#)

Scholar Commons Citation

Aldawoodi, Namir, "An Approach to Designing an Unmanned Helicopter Autopilot Using Genetic Algorithms and Simulated Annealing" (2008). *Graduate Theses and Dissertations*.
<https://scholarcommons.usf.edu/etd/114>

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.

An Approach to Designing an Unmanned Helicopter
Autopilot Using Genetic Algorithms and
Simulated Annealing

by

Namir Aldawoodi

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctorate of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Co-Major Professor: Rafael A. Perez, Ph.D.
Co-Major Professor: Kimon Valavanis, Ph.D.
Dewey Rundus, Ph.D.
Geoffrey Okogbaa, Ph.D.
Fernando Falquez, Ph.D.

Date of Approval:
March 21, 2008

Keywords: Function Generation, Formula Generation, VTOL Control, Automated
Helicopter Control, GPS Independent Pilot, Set-point Independent Pilot

© Copyright 2008, by Namir Aldawoodi

ACKNOWLEDGMENTS

I would like to thank Dr. Rafael Perez and Dr. Kimon Valavanis for their guidance, patience, and support as mentors and advisors. It was Dr. Perez's "Introduction to Artificial Intelligence" class that solidified my interest in Genetic Algorithms. I found the subject fascinating and, naturally, selected it as my field of research. Furthermore, I would like to thank Dr. Valavanis for his advice and help while I was in graduate school. It was in Dr. Valavanis' robotics course that I was introduced to many of the concepts and ideas that lead me in the direction of this study. I would also like to extend my thanks to the honorable members of the review committee for providing their time and insight. I would also like to thank Carlos Castillo (Graduate Student) for helping provide some of the materials and data used in this dissertation. Finally, I would like to thank the department of Computer Science and Engineering at the University of South Florida for their support and patience; especially the advising staff and others at the university who helped me complete all the paperwork on time.

TABLE OF CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vi
LIST OF ABBREVIATIONS AND ACRONYMS	xiii
ABSTRACT	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Problem Relevance	3
1.4 Applications of Unmanned Helicopters	3
1.5 Dealing with In-Flight Failures	5
1.6 Contributions of Proposed Pilot	7
1.6.1 Operation Format of Proposed Pilot	8
1.7 Summary of Contributions	8
1.8 Dissertation Outline	9
CHAPTER 2 LITERATURE REVIEW	11
2.1 Literature Review	11
2.2 Prevalence of UAV Applications	11
2.2.1 Examples of Civilian Applications of UAVs	13
2.2.1.1 The Altair	13
2.2.1.2 Altus I /Altus II	14
2.2.1.3 The Center for Interdisciplinary Remotely-Piloted Aircraft Studies (CIRPAS)	16
2.2.1.4 The Yamaha RMAX Unmanned Helicopter	16
2.3 Automated Helicopter Challenges	17
2.4 PID Controllers	24
2.4.1 Drawbacks of PID Controllers	26
2.5 Selecting a Search Tool	26
2.5.1 Classical Programming	26
2.5.2 Decision Trees	27
2.5.3 Statistical Regression Analysis	27
2.5.4 Soft Computing Methods	27
2.5.5 Artificial Neural Networks	28

2.5.6 Fuzzy Systems	30
2.5.7 Neuro Fuzzy Systems	31
2.5.8 Genetic Algorithms	32
2.5.9 Simulated Annealing	34
2.6 Selecting a Search Strategy	34
2.6.1 Why Use and Combine GA/SA	34
2.6.2 Similarities Between Genetic Algorithms and Simulated Annealing	35
CHAPTER 3 THE HELICOPTER SYSTEM	37
3.1 Helicopter Classification	37
3.2 Brief Helicopter History	37
3.3 Examples of Early Unmanned Helicopters	44
3.4 Examples of Most Recent Unmanned Helicopters	46
3.4.1 Yamaha RMAX	46
3.4.2 TAG Helicopters	46
3.4.3 The Autocopter	47
3.4.4 TGR Helicorp Alpine Wasp	49
3.4.5 Science Applications International Corporation (SAIC) UAV Helicopter	50
3.4.6 Lockheed Martin's Manned/Unmanned K-MAX Helicopter	51
3.4.7 Bell Eagle Eye	51
3.5 Examples of Experimental Helicopters	52
3.5.1 Unmanned Little Bird (ULB)	52
3.5.2 Boeing A160 Humming Bird	53
3.5.3 Northrop Grumman's Fire Scout	54
3.5.4 The Sky Tote	55
3.5.5 The Boeing Dragonfly (X-50)	56
3.5.6 Boeing Aerial Rotor Craft (UCAR)	57
3.6 Helicopter Tail Rotor Types	59
3.6.1 Traditional Tail Rotor	60
3.6.2 Fantail Rotor	61
3.6.3 No Tail Rotor (NOTAR)	62
3.7 Two Main Rotor Systems	64
3.7.1 Tandem Rotor Configuration	65
3.7.2 Coaxial Rotor Configuration	65
3.7.3 Intermeshing Rotor Configuration	66
3.7.4 Transverse Rotor Configuration	66
3.8 Tip-jet Configuration	67
3.9 The Rotor System	68
3.9.1 Rigid Rotor System	68
3.9.2 Semi rigid Rotor System	69
3.9.3 Fully Articulated Rotor System	69
3.9.4 Modern Rotor Configurations	70
3.10 Helicopter Flight Dynamics	70

3.10.1 Dynamic Control with Single Main Rotor	70
3.11 Helicopter Aerodynamic Considerations	73
3.11.1 The Autorotation Concept	76
3.11.2 Stability	78
3.11.3 Helicopter Limitations	78
3.12 Noise and Vibration	80
CHAPTER 4 THE PROPOSED AUTOPILOT	81
4.1 The Autopilot Module	81
4.2 The Proposed Autopilot	83
4.3 Collecting 'Observed' Data	85
4.3.1 Observed Data Source	85
4.3.2 The Control Equations	88
4.4 The Autopilot and Surrounding Architecture	89
4.4.1 Helicopter Control Architecture	89
4.4.2 Pilot Module Architecture	90
4.4.3 Pilot Module Behavior	91
CHAPTER 5 TECHNIQUE OF IMPLEMENTING A SOLUTION	93
5.1 Summary of Search Goals	93
5.2 The Flight Maneuvers	94
5.2.1 Simulating the Maneuvers	95
5.2.2 Approach Used in Duplicating the Maneuvers	95
5.2.3 Why Use a GA/SA Search Algorithm to Duplicate Maneuvers	96
5.2.4 Comprehensive Identification from FrEquency Responses (CIFER)	97
5.3 Genetic Algorithms and Simulated Annealing	98
5.3.1 Genetic Algorithms and the Search Space	99
5.3.2 Terminology	100
5.3.3 The Search Space	102
5.3.4 The Fitness Landscape	103
5.3.5 Possible Definition of a Genetic Algorithm	104
5.3.6 Genetic Algorithm Cycle	105
5.3.7 Why Genetic Search	107
5.3.8 Main Advantages of Genetic Search	108
5.3.8.1 Direct Manipulation of Encoded Variables	108
5.3.8.2 Search from a Population, Not a Single Point	109
5.3.8.3 Search Via Sampling, a Blind Search	109
5.4 Simulated Annealing	109
5.5 Summary Comparison of Genetic Algorithm and Simulated Annealing	114
5.6 Control Function Generation	115
5.6.1 Control Function Accuracy	115
5.6.2 Control Functions and Corresponding Functions	116
5.6.3 Function Testing Parameters	117
5.7 Data Collection	118

5.8 Mathematical Definition of a Function	120
5.9 Selecting Mathematical Primitives and Assembling Formulas	121
5.9.1 Assembling Formulas	121
5.9.2 Selecting Operators and Encoding Genes	122
5.9.3 Performance Criterion	122
5.10 The Solution Form	123
5.10.1 Assembling Building Blocks That Do Not Need Range Control	133
5.10.2 Constraints, Parameters and Assumptions	135
5.11 Running a Simulation	136
5.12 Testing Methodology	136
5.12.1 Test Goals	137
5.13 Implementation Platform	137
CHAPTER 6 EXPEREMENTAL RESULTS	139
6.1 Testing Summary	139
6.1.1 Summary of Static Test Results	139
6.2 Summary of Dynamic Test Results	144
6.2.1 Figure-8 Maneuver	145
6.2.2 Figure-8 Maneuver Testing Results	146
6.2.3 U-Turn Maneuver	151
6.2.4 U-Turn Maneuver Testing Results	152
6.2.5 Ascending Spiral Maneuver	157
6.2.6 Ascending Spiral Testing Results	158
6.2.7 Variable Height Figure-8 Maneuver	163
6.2.8 Variable Height Figure-8 Maneuver Testing Results	165
6.3 Sample Output of GA/SA Algorithm	169
6.4 Summary of Flight Path Error	171
6.4.1 Figure-8 Flight Path Error Summary	172
6.4.2 U-Turn Flight Path Error Summary	174
6.4.3 Spiral Up Flight Path Error Summary	178
6.4.4 Variable Figure-8 Flight Path Error Summary	181
6.5 Testing Conclusion	184
6.6 Conclusion of Research	185
6.7 Discussion and Future Work	186
REFERENCES	188
APPENDICES	197
Appendix A: Graphs of Control Signals	198
Appendix B: Brief VTOL History	202
B.1 List of Notable VTOL Development	202
Appendix C: Runtime Analysis of GA/SA Algorithm	207
ABOUT THE AUTHOR	End Page

LIST OF TABLES

Table 2.1:	UAV Regional Market Breakdown for Period 1997 to 2004	11
Table 5.1:	Set of Maneuvers to be Duplicated, Corresponding Flight Times and Flight Pattern Graphs	94
Table 5.2:	Comparing Genetic Algorithms with Simulated Annealing	114
Table 5.3:	A Sample Input Table that is used by the Search Algorithm to Derive Control Equations	119
Table 6.1:	Summary of Test Results	140
Table 6.2:	Summary of Figure-8 Path Errors in Relation to Set-points	174
Table 6.3:	Summary of U-Turn Path Errors in Relation to Set-points	177
Table 6.4:	Summary of Ascending Spiral Path Errors in Relation to Set-points.	180
Table 6.5:	Summary of Figure-8 Variable Height Path Errors in Relation to Set-points.	183
Table 6.6:	Summary of Path Errors in Relation to Set-points, All Figures are in Feet.	184
Table 6.7:	Summary of Euclidian Average Error of the GA/SA Controller with Relation to the Fuzzy Logic Controller.	185
Table C.1:	Computational Complexity Classes	208

LIST OF FIGURES

Figure 1.1:	Unmanned Agricultural Helicopter Use in Japan	5
Figure 2.1:	Classification of UAV Users	13
Figure 2.2:	NASA Altair UAV	14
Figure 2.3:	Altus II Flying over South California	15
Figure 2.4	Yamaha RMAX Helicopter	17
Figure 2.5	A Block Diagram of an Inertial Navigation System	19
Figure 2.6	An Example of a Bluetooth Based Location Determination System	22
Figure 2.7	A Block Diagram of a PID Controller	25
Figure 2.8	An Example of an Artificial Neuron where Weighted Inputs are Fed in and an Output is Provided in Response.	29
Figure 2.9	Fuzzy Logic Degree of Membership, Illustrates the Features of the Triangular Membership Function Which is Used in this Example because of its Mathematical Simplicity	32
Figure 2.10	Illustrates the Steps Involved in Creating a New Generation	33
Figure 3.1:	Paul Cornu's Helicopter (1907)	38
Figure 3.2:	Gyroplane-Laboratoire, 1933	39
Figure 3.3:	An Example of the Focke-Wulf_Fw_61 (1937)	40
Figure 3.4:	An Example of the Flettner Fl 282 Kolibri	41
Figure 3.5:	A Focke Achgelis Helicopter	41
Figure 3.6:	A VS-300 Piloted by Igor Sikorsky Towards the End of 1941	42

Figure 3.7:	Sikorsky R-4 Helicopter	43
Figure 3.8:	Bell 47 Helicopter	43
Figure 3.9:	Gyrodyne Helicopter	45
Figure 3.10:	A Drone Version of the HTK-1	45
Figure 3.11:	An Example of a TAG Helicopter	47
Figure 3.12:	AeroCopter Models	47
Figure 3.13:	AutoCopter Optional Arms Package	48
Figure 3.14:	The Alpine Wasp	49
Figure 3.15:	A Vigilante Helicopter	50
Figure 3.16:	The K-Max Unmanned Helicopter is Based on the K-Max Heavy Lift Helicopter as Shown	51
Figure 3.17:	Bell Eagle Eye Helicopter	52
Figure 3.18:	An Unmanned Little Bird Helicopter	53
Figure 3.19:	Boeing's A160T Hummingbird	54
Figure 3.20:	Northrop Grumman's Fire Scout	55
Figure 3.21:	Sky Tote	56
Figure 3.22:	An Example of an X-50	57
Figure 3.23:	Lockheed Martin Unmanned Combat Aerial Rotor Craft (UCAR)	58
Figure 3.24:	Northrop Grumman Unmanned Combat Aerial Rotor Craft (UCAR)	59
Figure 3.25:	Demonstrates the Torque Produced by the Main Rotor Along with the Anti-torque that Must be Generated by the Tail Rotor	60
Figure 3.26:	The Fenestron Tail System	62
Figure 3.27:	The NOTAR System	63

Figure 3.28:	MD NOTAR Helicopter	64
Figure 3.29:	The Mil Mi-12 Helicopter	67
Figure 3.30:	Stick and Pedal Controls	72
Figure 3.31:	Pitch and Roll Movements	73
Figure 3.32:	A Chart Showing the Progressive Loss of Lift as the Helicopter Approaches Critical Speeds	75
Figure 3.33:	The Height Velocity Diagram	77
Figure 4.1:	A Human Pilot Controlling a Helicopter Using a Joystick and Sensors	82
Figure 4.2:	A PID or Fuzzy or Human Pilot Controlling a Remote Helicopter	84
Figure 4.3:	Proposed Auto Pilot Module Controlling a Remote Helicopter	85
Figure 4.4:	GA/SA Derived Math Equation Controlling a Remote Helicopter	86
Figure 4.5:	Collecting Training Data for the GA/SA Search Algorithm	87
Figure 4.6:	Virtual Helicopter Input/Output Parameters	87
Figure 4.7:	Control Equations Flying the Virtual Helicopter	88
Figure 4.8:	Helicopter Control Architecture	91
Figure 4.9:	An Example of an Autopilot Module with Error Detection	92
Figure 5.1:	Typical Search Arrangement Using GA/SA Algorithms to Search in Parallel	96
Figure 5.2:	Example of Genetic Crossover	101
Figure 5.3:	Example of Genetic Mutation	102
Figure 5.4:	An Example of a Fitness Landscape	104
Figure 5.5:	The Simulated Annealing Algorithm	111

Figure 5.6:	Simulated Annealing Solutions Showing Local and Global Minima	112
Figure 5.7:	Helicopter Control Parameters	117
Figure 5.8:	Helicopter Sensor Readings	118
Figure 5.9:	Diagram Showing Control Equations Flying the Helicopter	119
Figure 5.10:	Two Functions that Make Up f_1	125
Figure 5.11:	Two Functions that Make Up f_1 with Range Control Implemented	126
Figure 5.12:	Diagram Showing $u(t)$	127
Figure 5.13:	Diagram Showing $v(t)$ and $w(t)$	127
Figure 5.14:	Diagram Showing $u(t)$, $v(t)$ and $w(t)$	128
Figure 5.15:	Diagram Showing the Resulting Function $y(t)$	129
Figure 5.16:	Diagram Showing how $y(t)$ is Generated at Point p_1	130
Figure 5.17:	Diagram Showing $u(t)$, $v(t)$ and $w(t)$ Generated at Multiple Points	131
Figure 5.18:	Diagram showing multiple points on $y(t)$	132
Figure 5.19:	Sample Longitudinal Control Signal	133
Figure 5.20:	A Longitudinal Signal being Mapped by 5 Building Block Functions	135
Figure 5.21:	The $u(x)$ Function being Derived from the Longitudinal (lon) Control Signal	138
Figure 6.1	Collective Signal Mapping Accuracy	141
Figure 6.2	Lateral Signal Mapping Accuracy	141
Figure 6.3	Longitudinal Signal Mapping Accuracy	142
Figure 6.4	Pedal Signal Mapping Accuracy	142

Figure 6.5:	Figure 8 (in flight loop) Closed Mode and Open Mode	146
Figure 6.6:	Figure 8, GA/SA Path and Fuzzy Logic Controller Path	147
Figure 6.7:	Figure 8, x vs Time Plot	148
Figure 6.8:	Figure 8, y vs Time Plot	149
Figure 6.9:	Figure 8, z vs Time Plot	150
Figure 6.10:	Figure 8, Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB	151
Figure 6.11:	U-Turn (in flight) Closed Mode and Open Mode	152
Figure 6.12:	U-Turn, GA/SA Path and Fuzzy Logic Controller Path	153
Figure 6.13:	U-Turn, x vs Time Plot	154
Figure 6.14:	U-Turn, y vs Time Plot	155
Figure 6.15:	U-Turn, z vs Time Plot	156
Figure 6.16:	U-Turn, Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB	157
Figure 6.17:	Ascending Spiral Maneuver Closed Mode and Open Mode	158
Figure 6.18:	Ascending Spiral, GA/SA Path and Fuzzy Logic Controller Path	159
Figure 6.19:	Ascending Spiral, x vs Time Plot	160
Figure 6.20:	Ascending Spiral, y vs Time Plot	161
Figure 6.21:	Ascending Spiral, z vs Time Plot	162
Figure 6.22:	Ascending Spiral, Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB	163
Figure 6.23:	Figure 8 Variable Height (in flight loop) Closed Mode and Open Mode	164

Figure 6.24:	Figure 8 Variable Height (in flight loop), GA/SA Path and Fuzzy Logic Controller Path	165
Figure 6.25:	Figure 8 Variable Height, x vs Time Plot	166
Figure 6.26:	Figure 8 Variable Height, y vs Time Plot	167
Figure 6.27:	Figure 8 Variable Height, z vs Time Plot	168
Figure 6.28:	Figure 8 Variable Height, Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB	169
Figure 6.29:	Ascending Spiral Sample Result Showing Fit	170
Figure 6.30:	Figure-8 Maneuver Showing the GA/SA Derived Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points	172
Figure 6.31:	Figure-8 Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path from That of the Fuzzy Logic Controller Path (Baseline).	173
Figure 6.32:	Figure-8 Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-points	173
Figure 6.33:	U-Turn Maneuver Showing The GA/SA Derived Equation Controller And Other Controllers Along With Their Respective Euclidean Distance From The Set-Points	175
Figure 6.34:	U-Turn Maneuver Showing the Euclidian Distance Of The GA/SA Derived Equation Controller Path from That of the Fuzzy Logic Controller Path (Baseline).	176
Figure 6.35:	U-Turn Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-points	177
Figure 6.36:	Spiral-Up Maneuver Showing the GA/SA Derived Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points	178

Figure 6.37:	Spiral-Up Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path From That Of The Fuzzy Logic Controller Path (Baseline).	179
Figure 6.38:	Spiral-Up Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-points	180
Figure 6.39:	Figure-8 Variable Height Maneuver Showing the GA/SA Derived Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points	181
Figure 6.40:	Figure-8 Variable Height Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path From That of the Fuzzy Logic Controller Path (Baseline).	182
Figure 6.41:	Figure-8 Variable Height Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-Points	183
Figure A.1:	Figure 8 (In Flight Loop) Showing All 4 Control Signals	198
Figure A.2:	U-Turn Showing All 4 Control Signals	199
Figure A.3:	Ascending Spiral Showing All 4 Control Signals	200
Figure A.4:	Variable Height Figure 8 Showing All 4 Control Signals	201
Figure B.1:	A German V/STOL VJ101 "Starfighter"	204
Figure B.2:	V-22 Osprey	205
Figure B.3:	X-35B Showing Lift Fan	206

LIST OF ABBREVIATIONS AND ACRONYMS

Genetic Algorithm	GA
Simulated Annealing	SA
Vertical Take Off and Landing vehicle	VTOL
Unmanned Aerial Vehicles	UAV
Department of Defense	DoD
Defense Advanced Research Projects Agency	DARPA
Proportional Integral Derivative	PID
Matrix Laboratory Mathworks	MATLAB
Artificial Neural Networks	ANNs
Artificial Intelligence	AI
Any local high point on a 2d or 3d graph	local maxima
Any local low point on a 2d or 3d graph	local minima
Anti-Lock Breaking System	ABS
Comprehensive Identification Frequency Responses	CIFER
Small unmanned Helicopter	SH
Fuzzy logic Controller	FC

**AN APPROACH TO DESIGNING AN UNMANNED HELICOPTER
AUTOPILOT USING GENETIC ALGORITHMS AND
SIMULATED ANNEALING**

Namir Aldawoodi

ABSTRACT

This dissertation investigates the application of Genetic Algorithms (GA) and Simulated Annealing (SA) based search techniques to the problem of deriving an auto-pilot that can emulate a human operator or other controller flying a Small unmanned Helicopter (SH). A Helicopter is a type of Vertical Take Off and Landing Vehicle (VTOL). The maneuvers are none aggressive, mild maneuvers, that include u-turns, ascending spirals and other none extreme flight paths.

The pilot of the helicopter is a Fuzzy logic Controller (FC) pilot; it is assumed that the pilot executes the maneuvers with skill and precision. The FC pilot is given set-points (points in space) that represent a path/flight maneuver and is expected to follow them as closely as possible. Input/Output data is then collected from the FC pilot executing maneuvers in real time. The collected data include control signals from the FC pilot to the SH and the resulting output signals from the SH that include time, x, y, z coordinates and yaw (the angle of the SH relative to the x, y axis). The Genetic Algorithm/Simulated Annealing based search algorithm attempts to generate a set of mathematical formulas that best map the collected data. The search algorithm presented in this dissertation was implemented in Java and has a JSP (Java Server Pages) graphical user interface.

The results obtained show that the search technique developed; termed Genetic Algorithm / Simulated Annealing controller or (GA/SA) controller allows for the derivation of accurate SH control equations. The results include performance quantification of the algorithm in the derivation phase and the testing phase. Graphs are included; they demonstrate the accuracy and path data of the GA/SA controller as compared to the FC pilot and other controllers. The final results showing the formulas found are also included.

A technique was also developed during this dissertation to encode the genetic strings that represent the candidate formulas during the search. This technique allowed the combination of strings to yield new formulas that are valid. The results can be used by other investigators to expand the complexity of the formulas generated during the search. The technique has advantages such as the ability to operate in open-loop conditions and is able to fly the SH without the need for set-point data and without the need for GPS or some other location determination technology. The technique may be used as a backup controller that can take over control of a helicopter in case the main controller is unable to function due to a GPS malfunction or another situation where accurate positioning data cannot be obtained.

CHAPTER 1

INTRODUCTION

1.1 Motivation

An unmanned aerial vehicle (UAV) is a pilotless aircraft controlled remotely or autonomously [1]. Vertical Take-Off and Landing (VTOL) vehicles are a subclass of UAVs that can take off and land vertically [2]. Small unmanned helicopters (SH) are a type of VTOL with important civil and military applications. A small unmanned helicopter can be flown by a ground control unit or by an auto pilot normally placed onboard [1]. Autopilots designs vary, however, as they need positioning data to achieve the task of controlling the helicopter [3]. Positioning information can be obtained from gyros, compasses, and other inertial navigation systems that determine location by dead reckoning. This is a navigation method that determines its current position by calculating assumed distance and direction moved since the last known location [4]. More modern methods include Global Positioning Systems (GPS), which use a system of satellites, computers, and receivers to determine location. It accomplishes this by comparing the time it takes for signals from different satellites to reach the receiver. There are also hybrid systems that combine dead reckoning with GPS to create a more reliable navigation system.

In addition to position information, an autopilot will require destination data or way points that define a desired path to a destination [6]. However, current autopilots have limitation, as they may encounter situations where 3-D positioning information is lost or unavailable. Another limitation is the need for way points to determine the next path to follow to a desired destination; should the way points become difficult to locate,

then the autopilot may not be able to reach its goal. In addition, some autopilot systems require intervention or assistance from a ground unit; hence, if the ground link is lost, the autopilot may not be able to effectively control the helicopter [7]. Currently, there is a lack of a viable position and way point independent backup autopilot. The main motivation of this research is to develop an independent backup autopilot that does not require location or path data to reach a predetermined destination. This new autopilot would serve as a backup to the main autopilot. The proposed system is independent of the type of autopilot onboard adding another level of redundancy and safety.

1.2 Problem Statement

It is the objective of this dissertation to investigate and quantify the ability of Genetic Algorithms (GA) and Simulated Annealing (SA) based search techniques to solve the problem of deriving an auto-pilot not requiring location or path data while directing a small unmanned helicopter to a pre-established destination. The maneuvers executed by this autopilot are non-aggressive mild maneuvers that include u-turns, spirals, and other non-extreme flight paths. As no unique answer exists, soft computing techniques will be used to generate mathematical equations that emulate the pilot's commands; the GA/SA derived controller (GSC) is expected to provide autonomous control of the small unmanned helicopter. This GSC, or automated pilot, can be onboard or on the ground—the location of the latter and the way it interfaces with the helicopter is not relevant to this research. The automated pilot is responsible for following a flight path generated by another control module such as the Fuzzy logic Controller (FC) pilot. The performance of this GSC autopilot module will be compared to other automated pilot systems such as a Proportional-Integral-Derivative (PID) controller, the FC controller, and other automated controllers.

The data is collected from an FC pilot executing a set of pre-selected maneuvers in real time. It is assumed that the pilot executes the maneuvers with skill and precision. The collected data include control signals from the pilot to the SH and the resulting

sensor output signals from the SH that include flight time, x, y, z coordinates and yaw. The collected data is used to derive a pilot as well as test the performance of the new pilot. The GA/SA based search algorithm attempts to generate a mathematical formula that best emulates the FC controller outputs using flight time as the sole input.

1.3 Problem Relevance

This problem is relevant as there currently is no technology that can pilot a SH without positioning and path data as presented in this dissertation. The proposed pilot can function without the need for GPS or other positing information. Furthermore, the pilot needs no path data to reach a pre-determined destination. The proposed pilot requires only flight time as input. As such, the proposed pilot will add a layer of redundancy to unmanned small helicopters. This redundancy makes this type of UAV more likely to recover from a failure of positioning or path data system.

The generated formulas model the system and enhance autonomous vehicle and unmanned vehicle navigation. It accomplishes this by providing a methodology to approximate the actions of human or automated SH operators, and captures that knowledge in a formula that can be used to control a SH. The generated formula allows for mathematical modeling of the system. The mathematical model allows for further study and analysis of the helicopter system. There may also be future applications related to missile avoidance systems that require a pseudo open loop control of a jet or a UAV.

1.4 Applications of Unmanned Helicopters

Several applications exist where unmanned helicopters are used; these include tasks considered Dull, Dirty and Dangerous [8]. Dull operations include missions that last a long time, for example, long intelligence gathering and surveillance flights. In these types of missions, the advantage of a UAV includes its alertness at the end of the

mission similar to the start of the mission; also, it does not suffer from human fatigue. Dirty missions involve operations in areas that may be hazardous to humans such as biologically or chemically contaminated locations. Dangerous missions include missions that pose a danger to humans such as combat missions.

Small unmanned helicopters have military applications that include reconnaissance, surveillance, target acquisition, communications relay, and re-supply [10]. For example, the A160 Hummingbird UAV differs from other helicopters on the market as it can reach higher altitudes and hover for an extended period of time. Additionally, it can travel further and operate at a much higher ceiling than current helicopters (30,000 feet vs. 20,000 feet) [11]. It also operates much more quietly and features a unique speed rotor that adjusts the speed of the main rotor at different altitudes and air speeds [11]. The A160 was developed by Frontier Systems Inc., of Irvine, California; the company was later purchased by Boeing in May 2004. The A160's unique characteristics are designed to meet U.S. Armed Forces' current and emerging requirements for military autonomous helicopters.

There are also civilian applications where small autonomous helicopters are used; some of the applications include reconnaissance and support in natural disaster areas, police observation, firefighting, agricultural chemical spraying, broadcasting and other applications [12]. One example of a civilian use helicopter is the Yamaha RMAX used for agriculture, as well as research. Figure 1.1 from Bernard Microsystems Limited (a UAV production firm) shows the increased use of agricultural helicopters in Japan. There exist significant reasons for this as crop-dusting can be hazardous to humans [13].

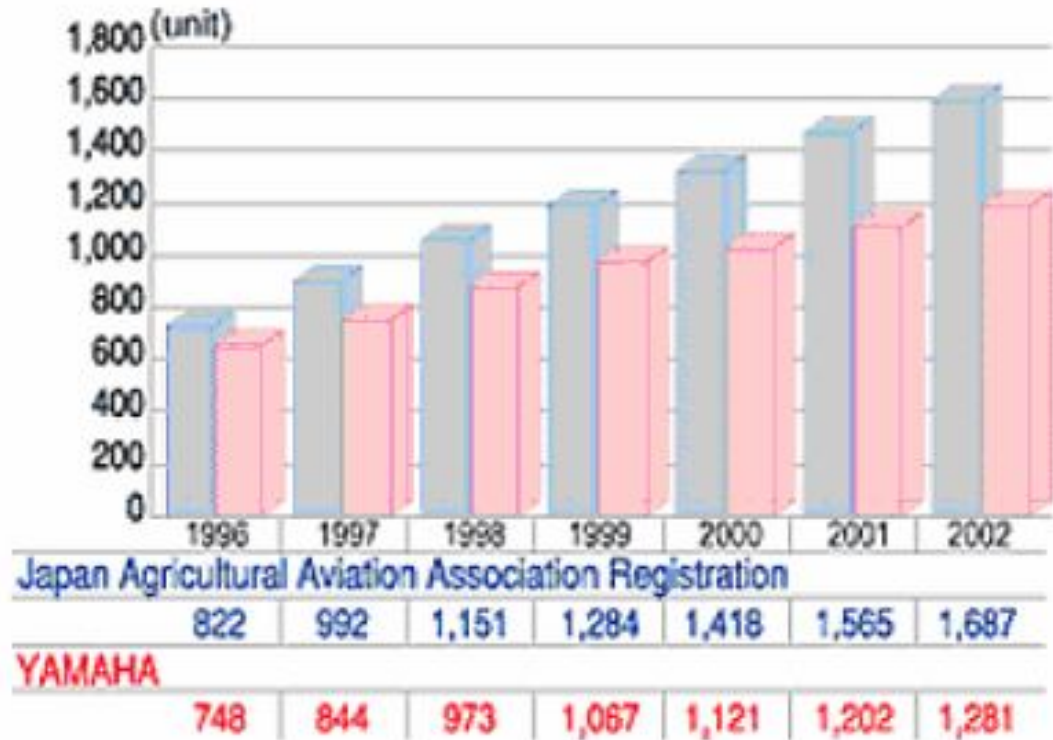


Figure 1.1: Unmanned Agricultural Helicopter Use in Japan. (source: http://www.barnardmicrosystems.com/L4E_uav_market.htm)

1.5 Dealing with In-Flight Failures

As with any complex system, the risk of mechanical, electrical or other failure always exists, one that can compromise the mission and result in the UAV becoming lost. Therefore, in-flight failures and crashes increase the cost of unmanned air vehicles, and limit their availability for missions [14]. The *Predator*, a U.S. made military UAV, had a crash rate of 32.8 per 100,000 flight hours in 2002 [14]. In 2003, the *Predator* rate increased to 49.6 per 100,000 flight hours; comparably, another US military UAV, the *Global Hawk*, had an accident rate of 167.7 per 100,000 flight hours [14]. In contrast, the *F-16* had a 2003 crash rate of 3.5 per 100,000 flight hours [14]. A Pentagon report recommends a UAV failure rate of 25 per 100,000 flight hours or less by the year 2009, and 15 per 100,000 flight hours by the year 2015 [14]. However, improving UAV reliability can be costly and offset the cost advantage of UAVs over

manned aircraft; one such way to boost reliability is to add redundancy [14]. The NASA *Global Hawk* has built-in redundancy in the form of four onboard GPS navigation systems.

Navigation applications for UAVs require a higher level of robustness. A navigation system must degrade gracefully regardless of sensor failures [15]. High-grade, non-UAV navigation systems utilize high-performance sensors able to use inertial navigation alone and without feedback from other sources such as GPS [15]. However, low-cost navigation systems for UAVs require feedback to constantly correct for errors in inertial sensors. Currently, unmanned aircraft may deal with a location determination failure (such as losing a GPS signal) by switching to an alternative location determination method. This includes ground based GPS or using radar systems such as laser radar to navigate. Communication failures may also occur, where the UAV is unable to contact home base. In this situation, the UAV may establish alternate forms of communication using alternative satellite or UHF based systems. The UAV may also switch to a holding pattern until communication is re-established or another course of action is determined. Returning to base or making an emergency landing at the nearest location is another option, provided the UAV is still able to navigate.

Recent advances to deal with in-flight failure include flying UAVs in formation such that if one UAV is compromised, then the rest of the formation can determine the best way to proceed and issue commands to the impaired UAV. This approach is known as the self-flying, self-planning, pack-hunting method. Currently, the U.S. is developing UAVs that implement pack-hunting and planning behaviors, which are expected to be operational by the year 2008. In addition, there are new onboard location determination systems such as the Shipboard Relative GPS. These systems are designed to provide an exact location within eight inches. This level of precision will enable automated aircraft to land on a carrier deck. Additionally, new UAVs will be designed to imitate swarming behavior to ensure that human/autopilot mix remains possible. The US military's plan for UAV improvements in the near future include navigation,

control, and path planning enhancements that will allow machines to have cooperative situation awareness. Also, the addition of mission-oriented robotic path planning to UAV's incorporates tactical information to deal with unforeseen circumstances. This includes the use of remote sensor information for the purpose of path planning, which is expected to improve mission survivability without the need for human intervention [16]. These newer techniques are expected to lower in-flight failure by providing redundancy, situational awareness, and robustness to deal with emergencies.

1.6 Contributions of Proposed Pilot

The proposed GA/SA derived pilot would add an additional layer of redundancy to an existing onboard pilot(s). It learns flight control signals as a function of flight time for a previously determined flight path. It can take over control if the main pilot is unable to fly the aircraft due to failure(s) involving positioning or path data. The GA/SA derived pilot is designed to be independent of 3-D positioning information requiring only flight time as an input parameter. The GA/SA derived pilot can be an advantage in unexpected emergencies, and may limit the loss of UAVs in situations where positioning or path errors occur.

The proposed pilot can also act as an additional component of an error detecting system; this is achieved by comparing the output (control) signals of the main pilot to that of the proposed pilot during flight. A large discrepancy between the two pilots may indicate an anomaly in the system, and early error detection may make a difference between a recovered UAV and one that is lost. The latter is part of Failure Determination Identification (FDI), a significant part of UAV design [17].

Non-adaptive control systems manage failures by providing fault-tolerance without the need to reconfigure the control structure [17]. However, in adaptive systems; one goal of FDI is fault isolation, which may include restructuring to prevent fault propagation to healthy components [17]. The GA/SA derived pilot can be useful in

detecting a failure and can be used as a resource in restructuring the control hierarchy of a UAV. Restructuring control describes a topic outside the scope of this research; hence, the proposed GA/SA pilot will not carry out restructuring on its own. Nevertheless, it may be used as a component in such a system to backup the main pilot in both adaptive and non-adaptive UAV control systems.

Another advantage to the proposed pilot is that it enables mathematical modeling of the system. Since a formula is generated, it clearly relates the input parameter to the output signals and can be used to further analyze the system's response as a function of time. As stated earlier, there are possible military applications where a form of open loop control may be useful in missile avoidance and defense countermeasures. One such proposal involves the use of an open loop missile evasion algorithm for fighters; the open loop control is recommended if the location of aircraft or the missiles targeting the aircraft is unknown [18].

1.6.1 Operation Format of Proposed Pilot

The proposed pilot duplicates a closed loop system by learning SH control commands issued during closed loop system operation with respect to flight time. It is assumed that the closed loop system may use any pilot, human or computer controlled. Hence, the proposed pilot will operate in a pseudo open-loop format by generating flight control signals based on flight time alone; this allows it to run independent of path data and positioning information.

1.7 Summary of Contributions

The significant contributions of this dissertation are summarized as follows:

- A technique for deriving a GA/SA based autopilot was developed.

- Proposed autopilot functions independently of positioning and path data when flying a predetermined path using flight time as sole input.
- A technique was developed to derive formulas from observed data.
- A technique was developed to encode genetic strings that represent candidate formulas during the search; designed so that any combination, crossover, mutation or perturbation of the encoded solution will continually yield a valid new combination.
- The generated formula(s) follow standard mathematical rules and are flexible in structure; hence, other investigators will be able to substitute alternate formulas or expand the complexity of the formulas in future work.
- Genetic Algorithm and Simulated Annealing hybrid search was shown to produce valid, usable results in deriving SH control equations.

1.8 Dissertation Outline

The dissertation is organized as follows:

- Chapter 1 includes motivation, problem definition and summary of the contribution.
- A Literature review of current UAV technology, soft computing methods including Fuzzy systems, Genetic Algorithms and Simulated Annealing are presented in Chapter 2.
- Chapter 3 discusses the helicopter system. The chapter briefly discusses the origins of helicopter and helicopter flight principles.

- Chapter 4 presents the proposed autopilot.
- Chapter 5 explores the technique of implementing a solution along with a discussion that defines the control signals, data collection and formatting. A brief overview of Genetic Search and Simulated Annealing along with crossover techniques and annealing schedules is presented. Later in the chapter, functions are discussed and the performance criterion is defined along with the fitness function. The solution form is presented with details on mathematical operators. Testing methodologies are also discussed along with test goals.
- Chapter 6 presents the testing results with graphs showing signal mapping. The final graphs demonstrating flight paths are presented with additional graphs that compare the GA/SA algorithm to other algorithms, including PID and fuzzy logic controllers. The chapter wraps up with concluding remarks about the GA/SA controller and discusses future research goals.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature Review

The following sections review UAV applications and research, PID controllers as well as search techniques. These were considered as possible methods to solve the problem of deriving a formula based pilot.

2.2 Prevalence of UAV Applications

In 2002, there were an estimated 2400 UAVs in both military and civilian applications [22], and in Japan there were 1565 unmanned helicopters used for agriculture [22]. UAVs account for about \$100 million annually in commercial sales alone [22]. The U.S. UAV spending reached \$3 billion in the 1990s, and it is expected that this budget will be tripled in this decade [24]. Table 2.1 shows the regional market use for UAVs from 1997 to 2004 [25].

Table 2.1: UAV Regional Market Breakdown for Period 1997 to 2004 (source: http://med.ee.nd.edu/MED9/Papers/Aerial_vehicles/med01-164.pdf)

REGION	MARKET PERCENTAGE (%)
Europe	25-30
North America	35-40
Pacific Rim	15-25
Middle East	10
Other	11-14

According to a National Aeronautics and Space Administration (NASA) report, the U.S. Air Force, Army, Marine Corps, and Navy all possess and operate UAVs of some type, the latter used for reconnaissance, combat support, and other related applications [30]. Military UAV application is starting to mature and find genuine applications in mission planning and support roles. Civilian applications for UAVs include those used by NASA, the National Oceanic and Atmospheric Association (NOAA), and the Department of Homeland Security (DHS) [30]. Figure 2.1 illustrates a NASA diagram that demonstrates the current private and public sector organizations considering or currently using UAVs. Note that several UAV-related technologies developed for military use have similar or identical applications in civil UAVs [30]. However, significant economic and philosophical differences exist between civil and military applications for UAVs. The DOD UAV has a specific combat role it is designed to fulfill. Thus, the vehicle design is combat-oriented with equipment such as armor shielding, sensor suites, and munitions that allow it to survive in a hostile environment. In a DOD application, the completion of the mission without harm to personnel takes precedence over cost considerations. Alternately, in civil applications such as NASA, the basic objective is to further develop the core technologies to reduce the economic cost while increasing flight safety.

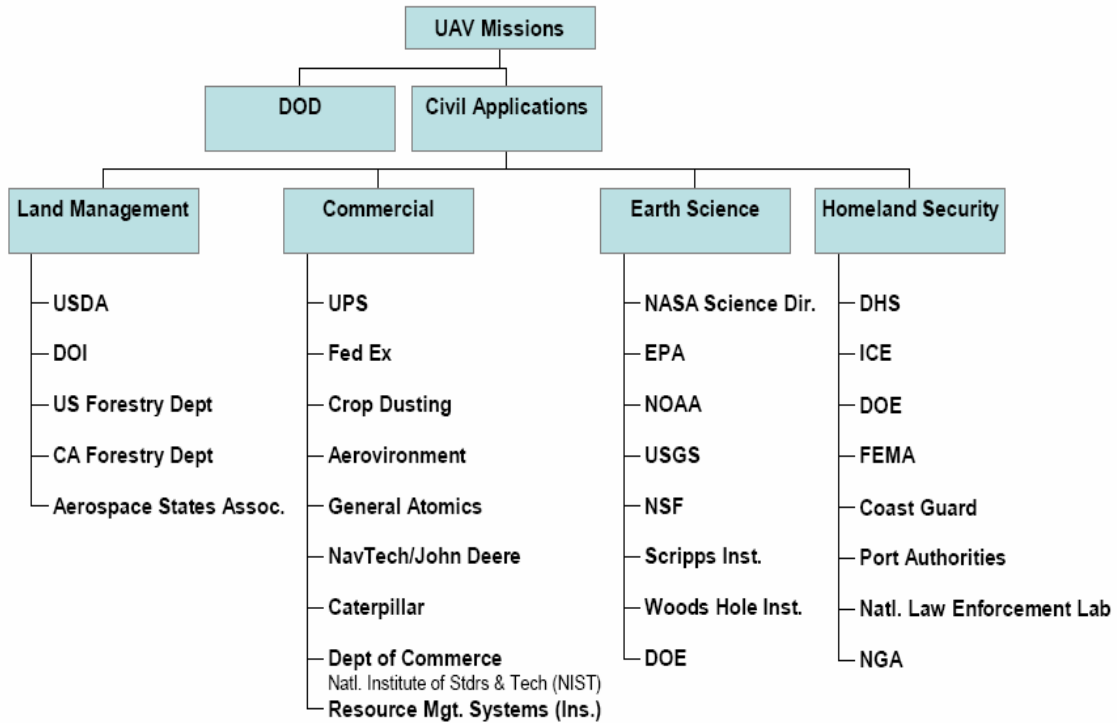


Figure 2.1: Classification of UAV Users (source: http://www.nasa.gov/centers/dryden/pdf/111761main_UAV_Capabilities_Assessment.pdf)

2.2.1 Examples of Civilian Applications of UAVs

The following sections will discuss operational UAVs currently used for commercial or research applications. Some of the UAVs are adapted from military versions while others were initially designed as civil UAVs.

2.2.1.1 The Altair

The Altair is a UAV built by General Atomics Aeronautical Systems Incorporated, and demonstrates a high altitude version of the military Predator B UAV. The Altair is designed for increased reliability, as it features a fault-tolerant flight control system and advanced avionics. In the current configuration, it can carry up to 660 lbs. internally, and an additional 3000 lbs. on its wings. This UAV operates with a 52,000

foot ceiling and can fly without refueling for 30 hours. It is currently used by the NASA Dryden Flight Research Center [30]. Figure 2.2 shows an Altair in flight [30].



Figure 2.2: NASA Altair UAV (source: http://www.nasa.gov/centers/dryden/pdf/111761main_UAV_Capabilities_Assessment.pdf)

2.2.1.2 Altus I / Altus II

The Altus aircraft was developed by General Atomics Aeronautical Systems Incorporated, located in San Diego, CA; this UAV illustrates a civilian version of the U.S. Air Force Predator with a similar appearance. However, it features a slightly longer wingspan geared more towards carrying atmospheric sampling and other scientific research instruments. This version can carry 330 lbs. of sensors and other equipment in a nose-mounted compartment. The location is designed to allow for sampling of fresh air

unaffected by heat or pollutants from the engine. Altus II features a ceiling of 65,000 feet and can fly without refueling for about 24 hours. The difference between Altus I and II is that the former includes a single-stage turbocharger, and the latter a two-stage turbocharger. The Altus I is currently used in the Naval Postgraduate School while the Altus II is located at the NASA Dryden Flight Research Center. Figure 2.3 shows an Altus II flying over south California [30].



Figure 2.3: Altus II Flying over South California (source: http://www.nasa.gov/centers/dryden/pdf/111761main_UAV_Capabilities_Assessment.pdf)

2.2.1.3 The Center for Interdisciplinary Remotely-Piloted Aircraft Studies (CIRPAS)

The Center for Interdisciplinary Remotely-Piloted Aircraft Studies (CIRPAS), a research center located at the US Naval Postgraduate School, was established in 1996 by the Office of Naval Research. The scientific community benefits from the data that CIRPAS collects from atmospheric as well as ground-based meteorological, aerosol, and cloud particle sensors. The data are collected and compiled on location and then provided in various formats to several user groups. CIRPAS operates assorted manned aircraft as well as unmanned aerial vehicles that include the UV-18A ‘Twin Otter’, the Pelican, the Altus ST UAV, the Predator UAV, and the GNAT-750 UAV. The National Oceanographic Laboratory also uses CIRPAS as a national research facility.

2.2.1.4 The Yamaha RMAX Unmanned Helicopter

The Yamaha RMAX helicopter was first introduced in 1983, and includes several applications including surveillance, crop dusting, and other agricultural uses. It can carry 65 lbs. and fly for 90 minutes [30]. For example, it can film a volcanic eruption from a close range, an extremely risky undertaking for a manned aircraft. It also includes applications for film and can take aerial shots. The Yamaha RMAX helicopter remains one of the most advanced, commercially available, UAVs currently on the market [31]. An outstanding feature of the RMAX includes its Yamaha-exclusive flight altitude control system, or YACS. This system allows RMAX to hover in a stationary position. The Yamaha YACS system’s design allows it to hover in place pending further instruction, without pilot input [31]. This added stability made it easier to train pilots to use the helicopter as well as lowered the loss rate. RMAX is also extremely refined with little vibration. Additionally, high sophistications afforded versatility that was previously impossible.

For example, if RMAX is ordered with a GPS system, it can be used to take extremely high definition photos from the same location over regular time intervals. This

feature allows farmers, for example, to accurately monitor and measure crop growth [31]. In the second quarter of 2003, Yamaha released an update to the RMAX, named the RMAX Type II G. The G implied that this UAV came with a Global Positioning System (GPS) standard. The RMAX base model featuring a single GPS module costs about \$86,000. The Aerial Photography version can fly about 500 feet above the ground and costs about \$150,000 for the base model, and as much as \$230,000 when fully loaded [31]. Yamaha also offers a flight research model designed for universities that features a manual-only flight mode, and retails for about \$120,000 [31]. Figure 2.4 illustrates the Yamaha RMAX helicopter.



Figure 2.4: Yamaha RMAX Helicopter (source: <http://www.yamaha-motor.co.jp/global/news/2002/02/06/sky.html>)

2.3 Automated Helicopter Challenges

Helicopters are multiple input, multiple output (MIMO) nonlinear machines. As a platform, helicopters represent unstable and highly coupled systems [2]. This inherent instability makes controlling a helicopter a challenging task. Helicopters require constant adjustment, otherwise they may crash or run off course. Furthermore, adverse weather

conditions such as rain or snow pose significant challenges, leading to possible loss of aircraft. In addition, there are also communication challenges to overcome. VTOLs that fly long range missions rely mainly on GPS, if the GPS signal is lost then the aircraft will not know its location and the mission may be compromised.

Bandwidth factors must also be considered. For example, the Pentagon is still weighing how many UAVs it can deploy to army units, as these UAVs may strain the limits of the military's satellite communications. Consequently, army officials may cancel two of the four planned UAV rollouts because of bandwidth concerns [26]. The UAVs are being developed under the Pentagon's Future Combat Systems (FCS) program. The army has considered adding more advanced sensors and weapon capabilities with the ability to transfer high-speed, full-motion video. All these new features will significantly increase the necessary bandwidth. Thus, it could overload existing satellite capability because it will require 45 megabits per second for each plane [26]. For example, a Predator or Globalhawk UAV in its current iteration needs a full transponder on a satellite to transfer data [26]. A larger number of UAVs (1000 or more) would exceed the current available bandwidth. Even 100 planes can present a significant problem, as UAVs require Ku-band capacity which is unavailable or scarce in the Middle East. The U.S. military uses some commercial satellites to overcome the lack of bandwidth; however, this is not always the best option [26].

Currently, the U.S. military is working on a Transformational Communications Satellite (T-Sat) program expected to launch in 2013 [26]. Once completed, this network will provide laser crosslinks and IP-router technology to help solve the bandwidth needs [26]. However, the T-SAT program suffered setbacks from budget cuts: in 2006, \$200 million was cut; and for 2007, the Senate Armed Services Committee is considering cutting back an additional \$70 million from the program [26]. There are also applications where UAVs are required to operate indoors, where GPS does not work well if at all. These missions usually involve indoor reconnaissance and rescue missions where the GPS signal remains quite weak. In such circumstances, developers of UAVs must

consider alternative location determination technologies to GPS [28]. The following technologies offer possible alternatives GPS:

- Inertial Navigation Systems
- Cellular phone networks (2G, 2.5G, 3G, and 4G)
- 802.11 based networks
- Bluetooth
- RFID - Radio-frequency identification
- Ultrasound
- UWB - Ultra-wideband
- IrDA - Infrared Data Association
- TV radio signals

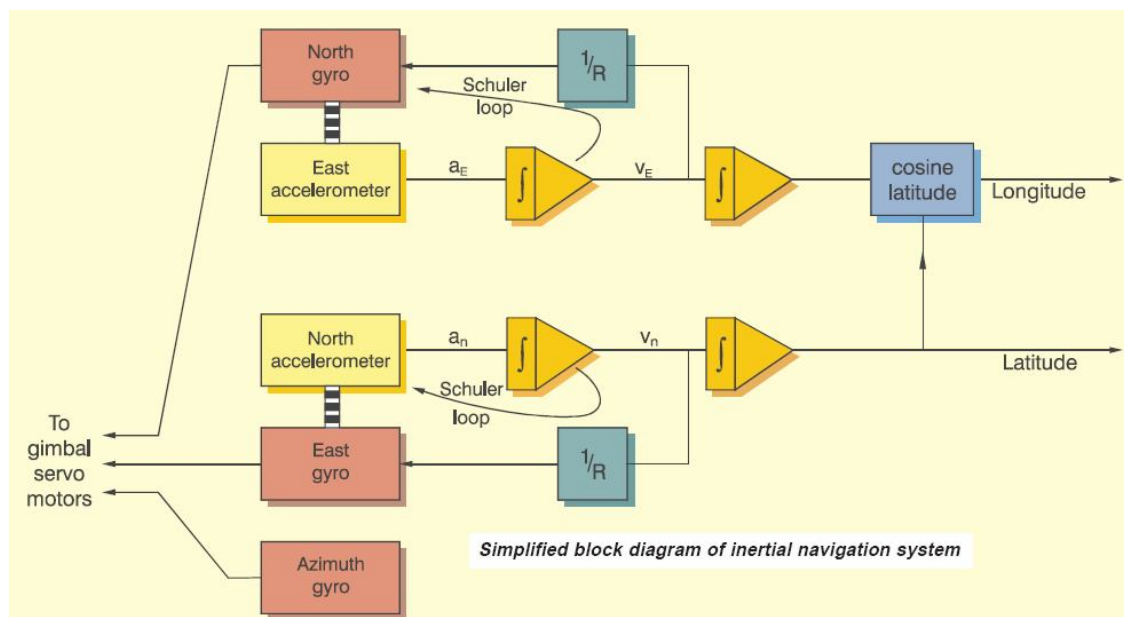


Figure 2.5: A Block Diagram of an Inertial Navigation System (http://www.imar-navigation.de/download/inertial_navigation_introduction.pdf)

Inertial Navigation Systems (INS) use dead reckoning to calculate position; this is achieved with accelerometers, gyroscopes and compasses. However, the likelihood of error increases with time. As such, INS tend to use GPS (when available) to synchronize

its position. Figure 2.5 shows a block diagram of a typical INS system. In the absence of GPS alternative technologies are used to locate position. Hence, a pseudo open loop navigation system such as the one proposed in this paper would present a viable backup system in situations where GPS or location data is not available. Search of current literature found no specific examples that used equations to control a small unmanned helicopter in pseudo open loop mode. Cellular phone networks can determine geographical positioning using some form of radiolocation that uses cellular base stations [34]. The most common method involves triangulation, using radio towers [34]. The location can be determined using one of several methods:

- Angle of Arrival (AOA): Requires at least two towers and locates a position using the point where the lines from each tower intersect.
- Time Difference of Arrival (TDOA): Similar to GPS as it uses multilateration (also known as hyperbolic positioning) and time difference in signal arrival to determine location. This system requires at least three towers to determine location.
- Location Signature: Uses "fingerprinting" to store and then recall patterns in signals. One example is multipath signature, which refers to radio signal propagation phenomenon resulting in signals following multiple paths to reach a receiving antenna. This results from atmospheric reflection or from bouncing off a terrestrial object such as a building or a mountain. [34].

The AOA and TDOA systems depend on a line of sight, which can be a challenge in mountainous terrain or around extremely large objects such as skyscrapers. The alternative is location signature, which tends to work more effectively in large cities or mountainous regions [33, 34]. In 2005, a startup company called Skyhook Wireless launched a Wi-Fi based positioning system as an alternative to GPS technology [32]. The WPS is a client software package that includes a reference database of about 1.5 million private and public Wi-Fi access points. The database is constantly updated with the locations of current and new Wi-Fi access points. Skyhook claims their technology has an accuracy of 70 to 140 feet. This system will be available in the 25 most populated cities

in the US [32]. The advantages of cellular location determination are that assets already exist. However, the disadvantages include the multi-path effect. Also, the TDOA localization method requires clock synchronization while the AOA method needs an antenna array. The system is inaccurate indoors and can be costly to maintain [36].

In Bluetooth based positioning technology, one way of determining location is through the use of a central server. In this situation, a mobile user first connects to a server, the server then uses the onboard Bluetooth interface to determine the closest Bluetooth provider. Subsequently, it retrieves the current location from the provider [35]. This model assumes that Bluetooth technology will become ubiquitous enough that a home or office environment would provide enough devices that can act as location providers [35]. Since Bluetooth transmitters have alternate power classes, the system will have different ranges of operation. The lowest power class will yield a communication range of about 35 feet [35]. This is enough to provide location information at room level granularity [35]. Figure 2.6 shows an example of a Bluetooth based location determination system [35].

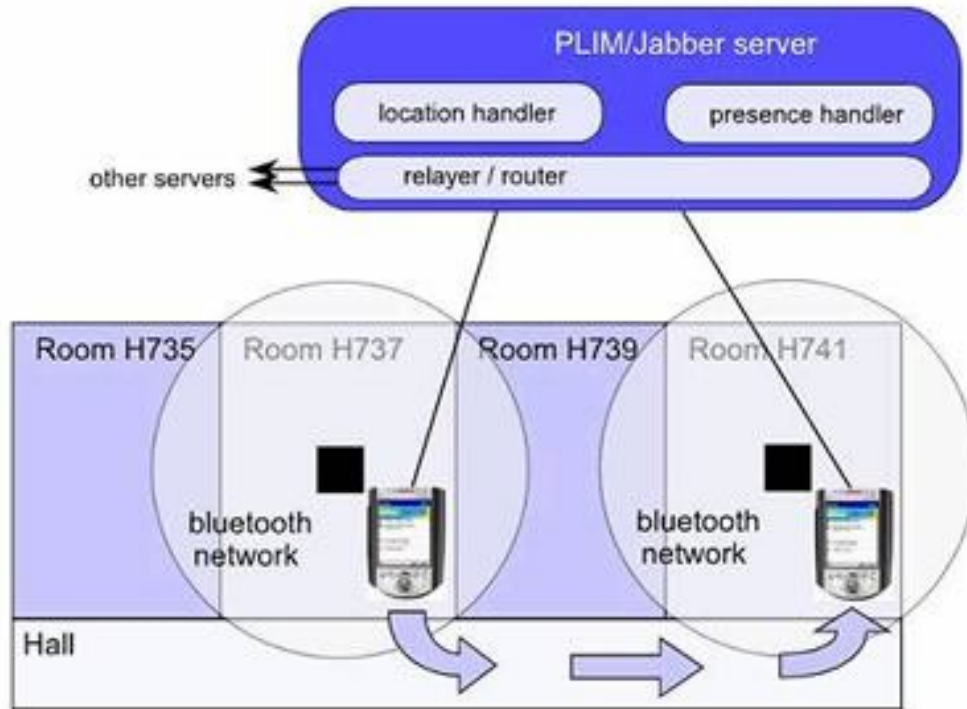


Figure 2.6: An Example of a Bluetooth Based Location Determination System (source: <http://www.telin.nl/index.cfm?language=en&context=660&id=659>)

An RFID system utilizes RFID tags to help determine location. For example, AXCESS' patented ActiveTag system uses low cost battery powered tags that transmit a wireless message with a range of 30 to 100 feet [36]. These messages are received by hidden receivers no larger than a human palm [36]. The receivers are connected using a standard network to enterprise system software. The system provides real time location displays as well as automatic inventory counts [36]. This is an example of a location determination system that can track inventory.

Ultrasound is also used for locating an object. One such system, the Bat (by AT&T) uses the principle of tri-lateration, which refers to finding a position by measuring distances [37]. The system works by a controller sending an RF request to the object, then a transmitter or Bat responds by sending a short pulse of ultrasound. The ceiling-mounted receivers determine the pulse's flight time to an accuracy of about three inches [35]. The advantages of this system include accuracy that makes it suitable for

indoors applications. The disadvantages include a large fixed-sensor infrastructure that must be built into the ceiling. In addition, the system is very sensitive to the placement of sensors and is not easy to deploy. The system can also be costly to install and does not scale well [35].

There are also magnetic tracking based position determination systems such as the Tracking Motion Star [35]. This system generates a magnetic pulse along an axial DC field. The position of an object is computed by measuring the response in three orthogonal axes [35]. The measurements are combined with those of the earth's fixed magnetic field. The system has an accuracy of 1 mm, illustrating its main advantage of high accuracy. Disadvantages include large implementation costs, the need to tether the object to a control unit as well as the requirement for sensors to remain within three to 12 feet of the transmitter. In addition, the system is adversely affected by the presence of metallic objects, which can cause shifts in accuracy [35].

Other location determination systems are infrared (IR) based. The IR Data Association's Active Badge (AT&T) works by using a badge located on an object [35]. The badge emits an IR signal unique to that object. Locally based IR receivers identify the signal and relay it to processing software that determines location. However, the IR signal can be blocked by walls and other objects. The system is able to locate an object with room-level granularity. IR based systems do not scale well due to the limited IR signal range [35]. High costs are also associated with the installation and maintenance of these systems. In addition, sunlight and fluorescent light can interfere with the IR signals.

TV and radio signals can also be used to determine location. A typical system assumes that a mobile receiver unit is in an area of at least three fixed commercial television stations. The system also assumes that each television station is broadcasting standard television signals. The TV broadcasts must include the vertical chrominance burst as well as the horizontal synchronizing signals. The standard television signals are received and processed by the mobile receiver. Next, the arrival times of the

synchronizing signals are measured and compared to a fixed reference station that receives TV broadcasts and produces reference signals. A processor then computes the location of the mobile receiver unit, using the mobile receiver unit and the reference receiver signals respectively [39, 40].

2.4 PID Controllers

Proportional, Integral, Derivative (PID) controllers are a generic control-loop feedback mechanism that has a wide application in industrial control systems. A PID controller compares two variables: one is a measured process variable and the other a desired set-point. The PID controller then measures the error between these two variables and responds with a corrective action that attempts to match the process variable with the desired variable [26]. The controller is designed to eliminate the need for constant operator interaction and to achieve automatic control of the plant. PID controllers have applications in cruise control systems for cars, industrial control, and several other applications where a need exists to constantly adjust input to maintain an optimal/desired point [26]. The goal of a PID controller is to hold the process variable at the desired value (set-point). Figure 2.7 shows a diagram of a PID controller [26].

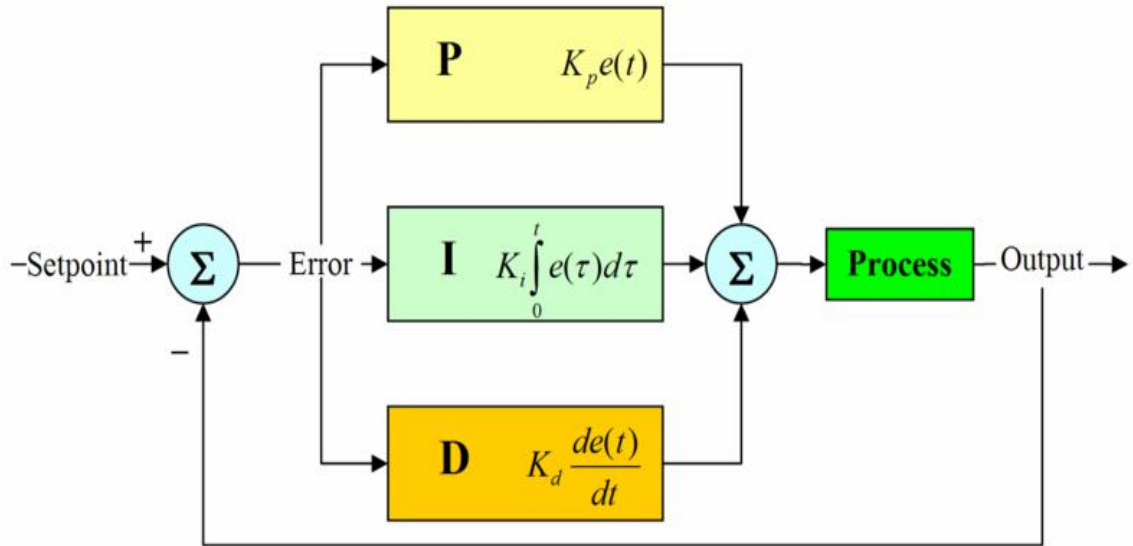


Figure 2.7: A block diagram of a PID controller (source: <http://upload.wikimedia.org/wikipedia/commons/4/40/Pid-feedback-nct-int-correct.png>)

PID controllers calculate the error between the output and set-point and respond to minimize that error. Additionally, changes in the set-point also lead to a change in the PID controller response.

PID controllers have three modes:

1. The P or Proportional Band controller: this output is proportional to the measured error.
2. I or Integral controller output: this output is proportional to the time the error has been present.
3. D or Derivative controller: this output is proportional to the how fast (rate) the change of error occurs with respect to time.
4. “Tuning” the PID controller involves choosing the correct values for P, I, and D.

2.4.1 Drawbacks of PID Controllers

PID controllers are prone to “hunting,” meaning they oscillate while seeking the right output to minimize error. In addition, if gains of proportional, integral and derivative are not optimally set; then the controlled process input can become unstable and oscillate. Since PID controllers are linear, their performance in non-linear systems may or may not be optimal. To make PID controllers perform effectively they should be enhanced through methods such as gain scheduling or fuzzy logic. Also, the differential term can be susceptible to small amounts of measurement (process noise), resulting in large errors or instability. In some applications, the differential band is turned off with almost no loss of control; the latter is equivalent to using the PID controller as a PI controller.

2.5 Selecting a Search Tool

In this dissertation, a search strategy is needed to construct the formulas that will control the SH. This strategy is needed in order to combine the mathematical building-blocks into a formula that is able to duplicate the control signals to the helicopter. The following methodologies were considered:

2.5.1 Classical Programming

Classical programming techniques were evaluated and found to be inefficient and unable to derive a set of equations from observed data in this application. This is because no assumptions should be made as to the shape of the sample data. If assumptions are made then the algorithm could be limited to considering solutions programmed ahead of time. Furthermore, constructs would have to be rather complex to search the domain.

2.5.2 Decision Trees

Decision trees were considered because of their relative speed (faster than version spaces) for a large concept space and where disjunction is easier to carry out. However, they were ruled out since they were not considered flexible enough to produce the function formats required. They are too complex to adapt for use in building formulas that must fit a set of data. Also, a decision tree may not always explain its classification clearly.

2.5.3 Statistical Regression Analysis

Statistical (regression based approaches) were found to be ineffective when the final form of the function is not known or if there is limited flexibility as to the mathematical components or primitives that can be used. The algorithm of choice must also be robust so that it does not get stuck in local maxima, and it must also be resilient to noise and discontinuous data. The algorithm must also be flexible enough to search for values for a pre-selected mathematical building block. There are, of course, statistical based methods available that can search for patterns in data. Mainly, regression analysis (RA), which can be used to analyze both linear and nonlinear data, but RA has its limitations since there are some problem classes that do not lend themselves well to this type of approach. These include problems in which there is little or no information about the function that generated data. In these cases, researchers typically use neural networks to learn more about the domain space of the function and then use this information to apply regression techniques. Hence, regression analysis would not be very useful since the domain space is not well defined and little is known about what the generated function should look like.

2.5.4 Soft Computing Methods

Soft computing methods have no universally accepted definition; however, they can be summarized as a non-traditional computation approach, where traditional

approaches are known as hard computing methods. Soft computing methods attempt to emulate the techniques and approaches used by humans and animals to solve problems. Soft computing techniques essentially copy biological learning or problem solving behaviors. These techniques include:

- Artificial Neural Networks
- Fuzzy Systems
- Genetic Algorithms and Evolution-based Strategies
- Simulated Annealing

Note that these approaches are not independent, but rather strongly related. They have a similar solution-strategy approach whose goal is to resolve complex problems that cannot be solved efficiently using traditional, hard computing, methods. Often, one or more methods are combined to produce hybrid approaches that may offer advantages to solving more difficult problems. The hybrid approach is usually utilized to minimize a bias or shortfall inherent in either of the combined methods. Neuro-Fuzzy approaches are but one example of a hybrid approach that offers advantages to using Neural Networks or Fuzzy Systems on their own.

2.5.5 Artificial Neural Networks

Artificial Neural Networks (ANNs) represents are a data driven approach that is essentially an information processing system. ANNs attempt to duplicate the behavior of the human brain by emulating biological neurons [27]. The artificial neurons are able to reason like a human at some level and are able to learn from data directly. They learn by example, so they do not need to know much about the system they are trying to duplicate. This makes them quite useful for applications where the process that generated the data need not be understood. Artificial Neural Networks usually include a series of neurons connected together. Each neuron is connected to another by weights that are adjusted. The learning process occurs when the weights on the neurons are adjusted so that the

network can fit a series of inputs to a known output series. Small adjustments based on gradient descent are made to the weight of a neuron. This is based on the difference between the current output and the desired output; a neuron is said to be ‘learning’ when the weight associated with one of its inputs is updated. Once training is complete, verification is fast, and as the connecting weights have no relation to the physical characteristics of the environment, it is considered a black-box model (Fig 2.8).

The quality of data directly affects the performance of an ANN; which depends upon the source, range, quantity and quality of data used in training. It is also crucial that the training data set is similar in features to the test data for ANN to perform well. If the training and test data have dissimilar features, the performance may vary. The latter is true of almost all soft computing methods that ‘learn’ from their environment. The main drawback to using a neural network to generate SH control equations is due to the difficulty in deriving mathematical equations that represent the knowledge learned by network. This is because neural networks do not provide a practical way to extract knowledge encoded in its weights and neurons. A mathematical formula cannot be easily derived with mathematical building blocks of the programmer’s choosing. Another difficulty lies in determining exactly what features of the data the neural network has encoded; hence, it can limit further analysis.

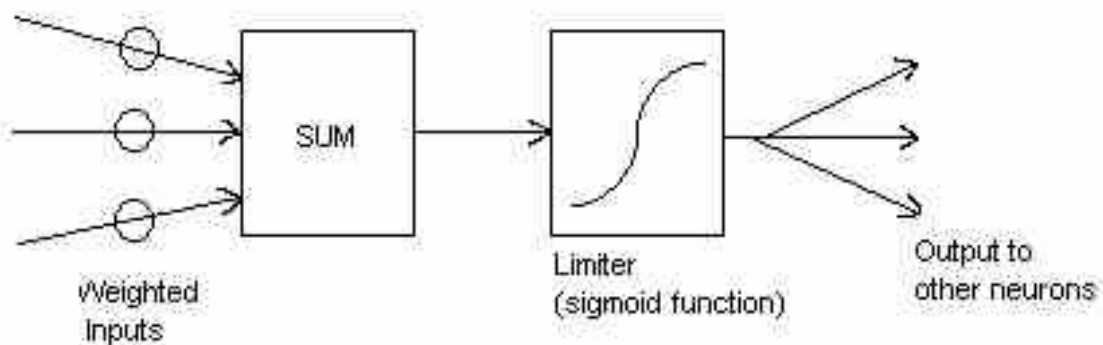


Figure 2.8: An Example of an Artificial Neuron where Weighted Inputs are Fed in and an Output is Provided in Response. During Training the Weights on the Inputs are Adjusted Until the Neuron Performs Satisfactorily. (source: <http://www.seattlerobotics.org/encoder/nov98/neural.html>)

2.5.6 Fuzzy Systems (Fuzzy Logic & Neuro-Fuzzy Systems)

Fuzzy Logic was developed in 1965 when it was introduced by Dr. Lotfi Zadeh in a paper in which he described the mathematics of Fuzzy Set theory. Dr. Zadeh formalized the theory of Fuzzy Logic in 1973. Since that time, Fuzzy Logic has been used in a wide range of applications, specifically those in industrial systems control. Fuzzy Logic systems are essentially structured numerical estimators. This approach extends Boolean logic to handle the concept of partial truth; the latter means that the truth takes a value not completely true and not completely false, but rather a value in between. To implement fuzzy logic, the idea of Fuzzy Sets is developed; a collection of objects that may belong partially to the set or belong to a degree. These take values between 0 and 1 rather than absolute one or zero. Figure 2.9 illustrates the concept of degree of membership.

Disadvantages of fuzzy logic controllers include the following:

- Requires a lot of data
- Fuzzy logic (sometimes) provides a crude sizing
- Not useful for new program types
- Not useful for programs much larger or smaller than the historical data

Fuzzy logic is a great tool to deal with problems of uncertainties and imprecise information. Data uncertainty can be due to randomness or lack of precision. Imprecision is usually due to lack of sharp boundaries in the information, whereas randomness has to do more with the nature of the event. Fuzzy logic can be used to construct formulas; however, it proved to be less attractive than GAs and SAs which lend themselves more easily to formula generation.

2.5.7 Neuro Fuzzy Systems

The Neuro-Fuzzy approach is a hybrid one for it uses the components of a conventional fuzzy logic system, but the computations at each stage are performed by a hidden layer of neurons. The neural network's learning capacity is used here to enhance the system knowledge. There exist various architectures of Neuro Fuzzy systems. The hybrid combination includes the benefits of a combined ANN and a Fuzzy Logic system. The resulting system also eliminates some disadvantages associated with Fuzzy Logic and Neural Networks by using their common features as a way to enhance them. The common features are distributed representation of knowledge, a model-free estimation, and the ability to handle data with uncertainty and imprecision. Respectively, Fuzzy Logic is tolerant of imprecise data while Neural Networks have a good tolerance for noise in the data.

A neural network's learning capability is used to automatically generate additional fuzzy rules and membership functions. This combination reduces the time and cost in design and implementation of a solution. The fuzzy logic approach contributes to this combination, enhancing the generalization capability of the neural network. It achieves this by providing a more reliable output reading when encountering data beyond the limits of the training data. This method is a viable method and could have been used in this study. However, it was ruled out because it is not straight forward to generate mathematical formulas from the resulting solution.

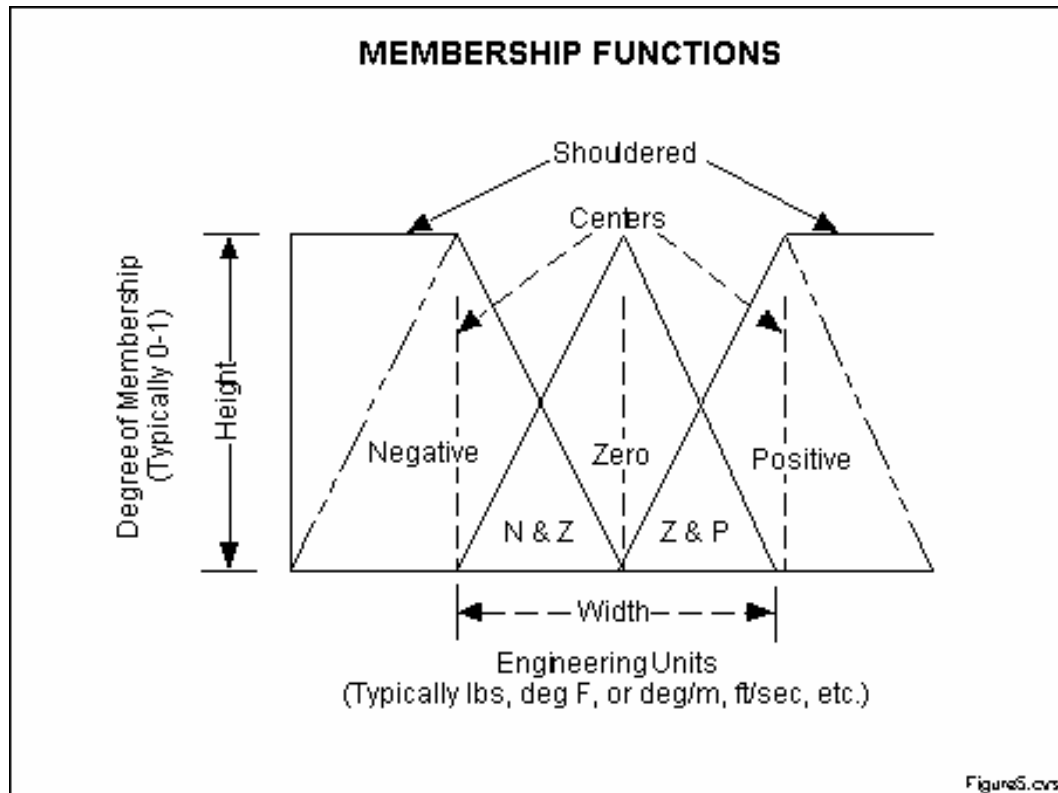


Figure 2.9: Fuzzy Logic Degree of Membership, Illustrates the Features of the Triangular Membership Function Which is Used in this Example because of its Mathematical Simplicity. Other Shapes can be used but the Triangular Shape Lends itself to this Illustration. (source: http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part4.html#INTRODUCTION)

2.5.8 Genetic Algorithms (GAs)

A genetic algorithm (GA) is a search technique used in computing that can find exact or approximate solutions to search and optimize problems. Genetic algorithms belong to the category of global search heuristics, and are considered a sub-class of evolutionary algorithms, or evolutionary computation. The latter is a class of techniques inspired by evolutionary biology. Hence, biological terms such as inheritance, mutation, crossover, and selection have been adapted in evolutionary computing. Genetic Algorithms are iterative in nature, so a population or collection of solutions is first randomly generated and then tested to evaluate their goodness or fitness. Finally, the

fittest solutions are selected and ‘bred’ to create the next generation. Mutation, which refers to random changes in genetic code, may also be used to cover more of the search space. The previous processes yield a new generation, and the process is repeated. Figure 2.10 shows a general lifecycle of genetic search techniques. The advantages of GAs include the ability to avoid getting stuck in local maxima. This is because GA’s search in parallel from several varied solutions and are not hindered by discontinuities in solutions expressed by mathematical formulas. This is because they directly manipulate a string representing those formulas [41, 42]. This is especially useful when searching large, complex domains, the type encountered when searching for unknown functions.

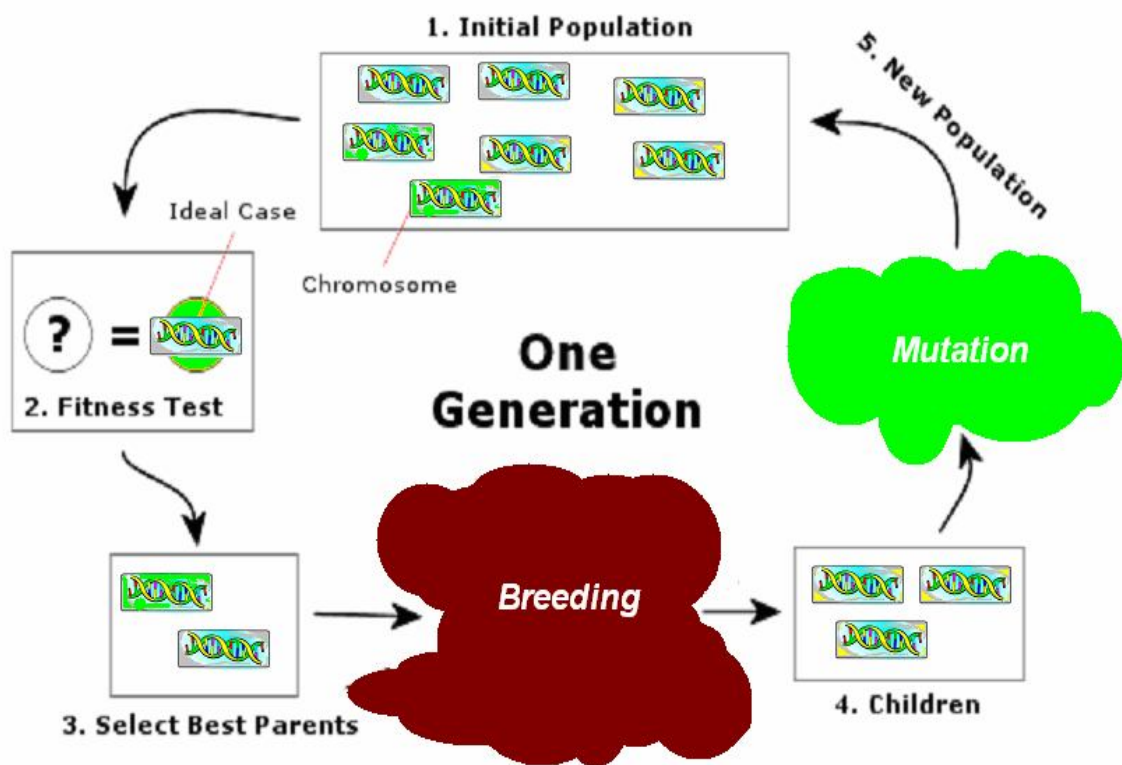


Figure 2.10: Illustrates the Steps Involved in Creating a New Generation. The Steps are Repeated a Finite Number of Times Till a Solution is Found or the Maximum Number of Generations has been Reached.

2.5.9 Simulated Annealing (SA)

Simulated annealing (SA) is a generic probabilistic meta-algorithm designed to solve the global optimization problem by finding a close approximation to the global optimum of some function in a large search space. This algorithm was invented by S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi in 1983; and independently, by V. Cerny in 1985. The name ‘Simulated Annealing’ refers to the inspiration for this algorithm, the annealing method in metallurgy. The latter represents an approach that involves heating and then cooling a metal in a controlled way for the purpose of achieving larger, less defective crystals within the material. The heat frees the atoms from their initial positions, a local minimum of the internal energy of the metal. Once heated, the atoms wander at random through higher energy states. Next, the slow rate of cooling allows the atoms a better chance of finding configurations with lower internal energy than the initial one.

2.6 Selecting a Search Strategy

The method selected to solve this problem is a soft-computation method that uses Genetic Algorithms and Simulated Annealing to search in parallel. This is a hybrid model that searches for an optimal solution within a time constraint. The search strategy will be referred to as GA/SA search.

2.6.1 Why Use and Combine GA/SA

Genetic and Simulated Annealing search techniques were selected because GA/SA search is not mathematically based. Hence, no direct calculations on the data would be performed. Therefore, data discontinuities, noise and data inconsistencies would not impact the search strategy. Second, GA/SA algorithms search for a solution independent of what the data looks like. This means that any search bias is reduced in relation to patterns present in the data, and the algorithm is free to seek any pattern hidden within

the data. Also, due to the random nature of GA/SA search, the results will vary from simulation to simulation. This increases the possibility of finding a better result. In addition, GA/SA search is resistant to getting stuck in local maxima [43], an important consideration when searching for patterns within a data set. Due to these advantages, GA/SA search is a good fit for deriving flight control functions from data, provided a mathematical building block is chosen correctly.

Although Genetic Algorithms and Simulated Annealing search share traits in their approach, a hybrid approach that uses both in parallel to search independently of each other was chosen for specific reasons. It was assumed that if there were any bias built into the data, having two searches in parallel would increase the chances of finding a solution. The 'solution population' is managed differently in each method. A GA will keep a whole group of solutions around (a population) while an SA will keep only one solution, which will be continually perturbed.

The GA/SA hybrid approach presented in this dissertation searches for an optimal solution in parallel and then waits for both searches to finish. The GA and SA search threads are independent. This means that the search parameters will be chosen so each algorithm receives approximately the same search time. The best performing result is chosen at the end of the simulation.

2.6.2 Similarities Between Genetic Algorithms and Simulated Annealing

Genetic Algorithms and Simulated Annealing have similarities. First, they both search in a finite number of cycles and 'home in' on a solution. Also, GA cycles are based on the number of iterations or generations, and SA cycles are measured by a temperature variable. The 'working set' of both strategies is somewhat similar as well. In a GA, a population of solutions remains and is 'bred' at each generation by combining bits and pieces from the current generation to produce the next. However, an SA includes a single solution perturbed multiple times during each temperature stage. This results in a

pseudo-population comprised of individual solutions, all derivatives of one solution. However, in SA, only one solution can survive to the next stage. This is in contrast to GA which uses a selected subset of the current population to 'breed' solutions to for the next generation.

Generally speaking, a GA always seeks the next lower energy state (less error) while a SA-based search may sometimes accept a higher error state to make the search less localized. This strategy is used to avoid local maxima in the search space. SAs maintain one solution and must to find a way to avoid localizing the search too much. Conversely, a GA can achieve a broader search by keeping less desirable 'individuals' in the current population. These 'individuals' may contain genes that will be useful later. Simulated Annealing guarantees an optimal solution if the temperature is annealed infinitely slowly, similar to how a GA will find an optimal solution if it is allowed an infinite number of generations.

CHAPTER 3

THE HELICOPTER SYSTEM

3.1 Helicopter Classification

Helicopters are a class of aircraft that generate lift and propulsion using one or more horizontal rotors [56]. Each rotor may have two or more rotor blades. Helicopters are categorized as rotary-wing aircraft because they generate lift from the rotor blades spinning around a mast. The term “helicopter” is derived from the French word hélicoptère [56]. The origins for the French word hélicoptère may be traced back to the two Greek words: ‘heliko’ (meaning spiral) and ‘pteron’ (meaning wing) [92]. Since lift can be generated without the need to move the aircraft forward, the main advantage of helicopters lies in their ability to take off and land vertically. As a result, helicopters are used in isolated, congested, or remote areas that are not accessible to fixed wing planes. Helicopters are also capable of hovering in one location for prolonged periods and with more efficiency than alternative vertical take-off and landing (VTOL) methods [56].

3.2 Brief Helicopter History

One of the earliest helicopter models was flown in 1901 by Ján Bahýľ (Slovak inventor) who developed a helicopter that used an internal combustion engine as a power source and managed to reach height of about three feet. In a later attempt on May 5, 1905, Bahýľ’s helicopter reached a height of 14 feet and flew for about 5300 feet [56]. In 1906, the French brothers Jacques and Louis Breguet experimented with airfoils leading to the development of a plane they called “Gyroplane No.1” which managed to carry one man about two feet in the air for about a minute. This was considered the first manned helicopter flight. However, since the plane was extremely unstable and required two men on the ground to hold it at either end it, the flight was considered a tethered flight [56]. At

about the same time, the French inventor Paul Cornu developed a helicopter that featured two 20 foot counter-rotating rotors powered by a 24 hp engine. This plane was able to achieve a height of one foot and stay aloft for 20 seconds without the need for tethering to keep it stable; thus, this is regarded as the true piloted free flight. However, instability issues caused Cornu to abandon the project. Figure 3.1 shows Paul Cornu's helicopter [56].



Figure 3.1: Paul Cornu's Helicopter (1907) (source: <http://en.wikipedia.org/wiki/Helicopter>)

The Gyroplane Laboratoire, shown in figure 3.2, is probably one of the earliest practical applications of helicopter design. The plane was built by the french designer Louis Breguet. The design featured an open steel tube framework that housed the engine and the fuel tank. It was powered by a 240 HP radial engine that turned two coaxial rotors [56].

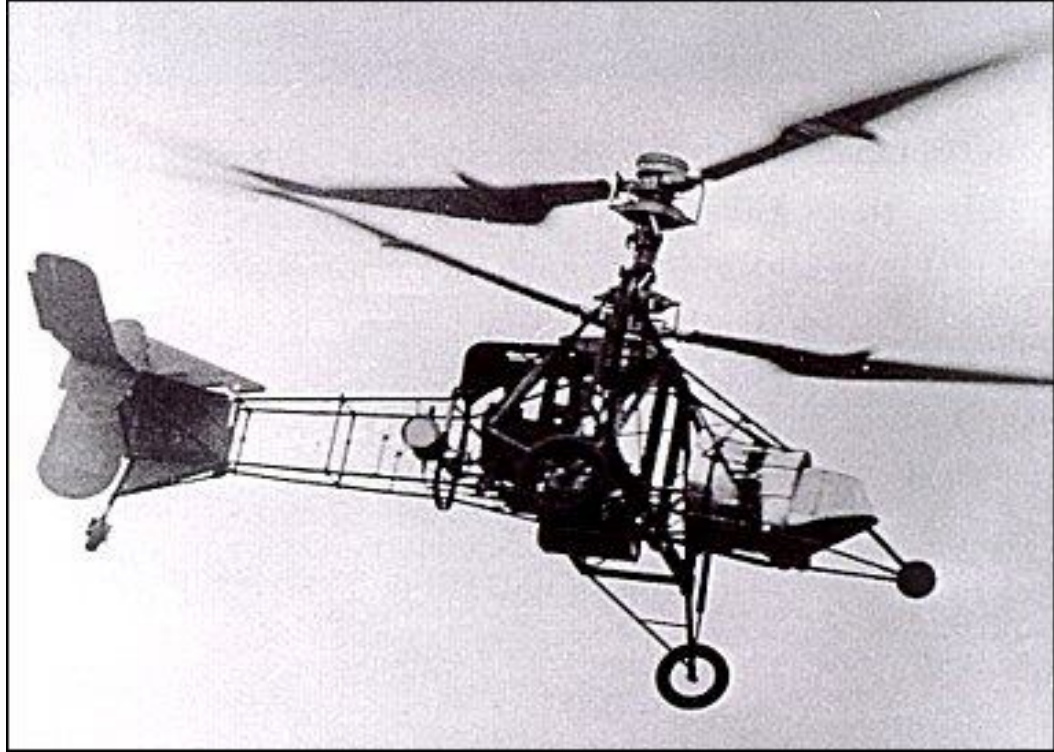


Figure 3.2 Gyroplane-Laboratoire, 1933, (source: http://en.wikipedia.org/wiki/Gyroplane_Laboratoire)

In 1936, Germany designed what is considered to be the first useful helicopter. The German Focke-Wulf Fw 61 broke many helicopter world records at the time it was introduced in 1937. Its design was like that of a fixed-wing aircraft that included a tail; a front propeller was added to provide forward thrust. The main rotors were located where the wings would normally go on a fixed-wing airplane. Two counter-rotating rotors were mounted on long sideway struts. Figure 3.3 shows an example of the Fw 61 plane [57].



Figure 3.3: An Example of the Focke-Wulf_Fw_61 (1937) (source: <http://en.wikipedia.org/wiki/Helicopter>)

The Germans also designed the Flettner F1 282 Kolibri synchropter with a modern looking hull design but with no tail rotor. The helicopter featured two counter-rotating/intermeshing rotors located on the cabin top and situated close together. Each rotor tilted outwards to avoid hitting the other rotor's shaft. These models were used in WWII and saw action in the Mediterranean Sea. Figure 3.4 shows an example of the F1 282 helicopter [58]. The first German helicopter to reach production is the Focke-Achgelis Fa 223 Drache [56]. The name "Drache" means "Dragon", about 20 palnes were made during WWII. The Fa 223 used radial engines that developed 1,000 horsepower. Figure 3.5 shows an example of the Focke-Achgelis Fa 223 Drache.



Figure 3.4: An Example of the Flettner Fl 282 Kolibri (source: <http://en.wikipedia.org/wiki/Helicopter>)



Figure 3.5: A Focke Achgelis Helicopter (source: http://en.wikipedia.org/wiki/Focke_Achgelis_Fa_223)

Igor Sikorsky was an American helicopter builder of Russian ancestry. The first helicopter he built was the Vought-Sikorsky 300; it flew in 1939 tethered and then flew un-tethered the following year. Figure 3.6 Shows a VS-300; the design featured one

main rotor and a tail rotor [59]. Sikorsky is credited with building the first full-scale production helicopter in 1942, with a total of 400 copies made. The unique thing about the Sikorsky R-4 helicopter was the addition of the anti-torque tail rotor to the design; this configuration would become the most common helicopter design worldwide. The R-4 entered service with the United States Army and was used in Burma for rescue missions in WW II. The British Royal Air Force became the first British military unit to be equipped with helicopters which led to the formation of the Helicopter Training School in January 1945. The school was located at RAF Andover, and nine Sikorsky R-4B Hoverfly I helicopters were used to train British pilots [59]. Figure 3.7 shows a Sikorsky R-4 helicopter [60].



Figure 3.6 A VS-300 Piloted by Igor Sikorsky Towards the End of 1941 (source: <http://en.wikipedia.org/wiki/Helicopter>)

In March 1946, the Bell 47 designed by Arthur Young became the first helicopter to be licensed for certified civilian use in the US. In 1967, the Bell 206 became the most successful commercial helicopter ever built, with more flight hours and more industry records than any other aircraft in the world. Figure 3.8 shows a Bell 47 helicopter.



Figure 3.7: Sikorsky R-4 Helicopter (source: <http://en.wikipedia.org/wiki/Helicopter>)



Figure 3.8: Bell 47 Helicopter (source: http://en.wikipedia.org/wiki/Bell_47)

The development of turbo-shaft engines after WWII allowed designers to build larger and faster helicopters. One of the early helicopters with turbo-shaft engines was the Kaman K-225 synchropter which was introduced in 1951 [56]. Turbo-shaft power plants are the usual power source used in most helicopters with the exception of small or inexpensive helicopters.

3.3 Examples of Early Unmanned Helicopters

Enrico Forlanini is credited with building the first unmanned helicopter back in 1877; his design did not have active stabilization or steering [56]. The technology was in its infancy in the first half of the 20th century; however, technological advances made after WWII made it feasible to build and control unmanned helicopters. One of the earliest companies is Gyrodyne of America who produced various unmanned helicopter models including the models for the U.S. Navy DASH program [82]. The program started in the 1950's when the U.S. Navy was looking for a way to control the Russian submarine threat. At the time, submarine torpedoes had limited range and complex control systems in addition to expense. Gyrodyne had viable coaxial helicopter designs that can be used by the Navy as a cost effective method to deliver conventional and nuclear anti-submarine weapons [82]. The helicopters would be stationed on regular Navy ships and carriers. Figure 3.9 shows an example of a Gyrodyne coaxial unmanned helicopter.

The Kaman Aircraft Company was founded in 1945. The company specialized in the design and manufacturer of manned helicopters including the HTK-1 (1954) which was a piloted helicopter. Kaman also manufactured the world's first electrical drone (this term was used for early UAVs because they were very basic) in 1953. The Kaman QH-43 unmanned helicopter was the first helicopter in the world to be controlled by remote control (1957). Figure 3.10 shows an example of a Kaman HTK-1 drone (QH-43).



Figure 3.9: Gyrodyne Helicopter (source: <http://www.gyrodynehelicopters.com>)



Figure 3.10: A Drone Version of the HTK-1 (source: http://www.gyrodynehelicopters.com/mitscher_class.htm)

3.4 Examples of Most Recent Unmanned Helicopters

3.4.1 Yamaha RMAX

The Yamaha RMAX ranges in price with base models starting at \$86,000 and can go as high as \$1,000,000 for a fully loaded aircraft [84]. The Yamaha has wide applications from agriculture to aerial photography to surveillance. It features single or multiple GPS systems onboard as well as cameras. It can fly in manual or fully automatic modes. Depending on packaging and options, a fully automated RMAX carries four cameras and sends back pictures in real time while it flies along a predetermined flight path. If the operator observes something that they want a closer look at, they are able to override the pre-programmed flight path and fly manually. Once the observation task is done, the RMAX can continue following the original flight path.

3.4.2 TAG Helicopters

The Tactical Aerospace Group (TAG) manufactures different models such as the TAG-M65, TAG-M80 and TAG-M100. The company makes four product lines from a military model that is designed to be rugged and operate in hostile environments to a civilian version that is designed for commercial applications [85]. The company also makes target drones for military application as well as a low-cost trainer model. The none-drone models can be built with digital and video cameras as well as a film camera. The helicopters also come with sensors, instrumentation, IR, and thermal cameras. These small VTOLs have applications such as surveillance pipeline inspection, power line inspection, and can serve as an ad-hoc communication relay. Figure 3.11 shows an example of a TAG helicopter.



Figure 3.11: An Example of a TAG Helicopter. Note the guns mounted below (source: <http://www.tacticalaerospacegroup.com>)

3.4.3 The Autocopter

The AutoCopter is a self-stabilizing helicopter that uses neural-networks for control. The proprietary neural-network used is able to learn; as a result, the system can compensate for wind and weight variation as well as other factors. As a result, the AutoCopter is stable in hover and in flight. Figure 3.12 shows some AeroCopter Models.



Figure 3.12: AeroCopter Models (source: <http://www.neural-robotics.com>)

The neural-network controlled AutoCopter is easy to fly (little training is needed); this is because the system will not allow the pilot to enter commands that will make the helicopter unstable or cause it to crash [86]. There is a semi-autonomous flight mode that allows the operator to control the VTOL. In addition, there is a fully autonomous flight mode that uses a GPS system to navigate. All the operator has to do is to upload set-points (flight plan) using a laptop computer. Once the flight plan is loaded, the AutoCopter is able to execute the flight plan. At any time during the flight, the ground-based pilot can override the autonomous mode. Should the helicopter lose contact with ground control, it will turn back until it is within radio range, and if it runs out of fuel, it will deploy a parachute automatically [87].

Options include video and IR sensors as well as still cameras and spectrometers. The helicopter comes with a laptop PC, a GPS receiver, a heading gyro, and other accessories [87]. There are also arms packages that consist of a 12-gauge Auto Assault-12 Full-Auto Shotgun as shown in figure 3.13. The design is “smooth and robust” in flight allowing the AutoCopter enough stability to be used in aerial photography. The AutoCopter is made by Neural Robotics Incorporated (NRI) and retails just shy of \$100,000.



Figure 3.13: AutoCopter Optional Arms Package (source: <http://www.defensereview.cm/article846.html>)

3.4.4 TGR Helicorp Alpine Wasp

The Alpine Wasp will be a derivative of TGR's SNARK 2, which is a military UAV [89] that carries high-resolution IR Cameras mounted in a way that allows a 360 degree coverage. It can also take off and land automatically, and it's satellite controlled. The fuselage is of a carbon fiber and Kevlar construction that allows it to be strong and stealthy. The military version is armed and has two sub versions: the Land based Snark and Sea Snark for Naval applications. Many of the helicopter specifications such as speed, endurance, and armament are classified. It does feature a diesel engine that allows to it be very efficient. The civilian rescue version is intended to operate on Mt. Everest as a medical emergency service based in Nepal. The Wasp will be capable of being operated autonomously and can reach heights of 30,000 feet which is impressive for helicopters because they normally operate at a maximum ceiling of about 14,000 feet [89]. Mt. Everest is 29,055 feet high, so the Wasp is well suited to operate at these altitudes. Figure 3.14 shows the Alpine Wasp.



Figure 3.14: The Alpine Wasp (source: <http://robotgossip.blogspot.com/2007/02/unmanned-helicopter-for-everest.html>)

3.4.5 Science Applications International Corporation (SAIC) UAV Helicopter

The Vigilante VTOL is made by SAIC which markets a range of small and inexpensive unmanned helicopters. The Vigilante 502 model is an unmanned helicopter that is able to achieve a maximum speed of 117 mph. The Vigilante UAV features air and ground support. This VTOL and many of its variants weigh about 1,100 lbs. and are 26 feet long [90]. The main rotor has a diameter of 23 feet and a height of 8 feet at the rotor. It can also reach altitudes around 12,000 feet and has a payload capacity of about 150 lbs. It has a good range since it features a 36-gallon fuel tank. The helicopter control system allows for an autonomous flight. This helicopter is currently in use by the U.S. Army and Navy. Figure 3.15 shows an example of a Vigilante.



Figure 3.15: A Vigilante Helicopter (source: <http://www.saic.com/products/aviation/vigilante/vig.html>)

3.4.6 Lockheed Martin's Manned/Unmanned K-MAX Helicopter

The unmanned K-Max helicopter was constructed as a result of collaboration between Lockheed Martin and Kaman helicopter [91]. Although it is somewhat slower than the competing Northrop Grumman's Fire Scout, its lifting capability is far superior as it is able to lift 6,000 lbs and can reach a maximum altitude of 29,000 feet. The piloted version (K-Max) is a large, single-seat helicopter employing a dual-meshed main rotor system that eliminates the need for a tail rotor. The FAA certified this model in 1994. This type is used mainly for the transport of heavy external loads needed for the logging and construction industries. Lockheed Martin is developing an autonomous control system that will feature specialized mission avionics. The resulting automated K-Max sustained flying for more than 12 hours without a human pilot on board. This system may compete with the Boeing Little Bird helicopter program. Figure 3.16 shows a K-M-Max helicopter.



Figure 3.16: The K-Max Unmanned Helicopter is Based on the K-Max Heavy Lift Helicopter as Shown. (source: <http://www.kamanaero.com/helicopters/uav.html>)

3.4.7 Bell Eagle Eye

Bell Helicopter is working on the TR918, also known as the Eagle Eye tilt rotor, a UAV. It is notable that the FAA has issued an “experimental flight certificate” for the UAV which is first time that a vertical-lift UAV has been issued such a certification [94]. This model features fly-by-wire flight controls with the ability to carry up to 200 lbs. and an

endurance of about five hours. There is also a very similar Navy version. Potential customers include the U.S. Marine Corps, the U.S. Army as well as NATO. Figure 3.17 shows the Bell Eagle Eye helicopter.



Figure 3.17: Bell Eagle Eye Helicopter (source: <http://www.vtol.org/news/issues1205.html>)

3.5 Examples of Experimental Helicopters

3.5.1 Unmanned Little Bird (ULB)

This aircraft is designed by Boeing and is based on an MD 530F helicopter [94]. It can fly with or without a safety pilot. It is being tested at the U.S. Army's Yuma Proving Grounds which is located in Mesa, Arizona. The ULB is able to take off, hover, and carry out unmanned missions on its own. It is able to land itself on a helipad with six inch accuracy. The first flight of the ULB was in October 2004. It is able to carry about 740 pounds of payload; newer models are expected to carry an additional 800 pounds. Funding for the ULB is provided by Boeing itself as a proof-of-concept for the Level 5 UAV sensor control. The system is adaptable and can be used on other manned and unmanned aircraft and can function as a full autopilot. Figure 3.18: An Unmanned Little Bird helicopter.



Figure 3.18: An Unmanned Little Bird Helicopter (source: <http://www.vtol.org/news/issues706.html>)

3.5.2 Boeing A160 Humming Bird

The A160 is a demonstrator aircraft built for the U.S. army; it is also known as the “Hummingbird.” The first flight was on January 29, 2002. It weighs approximately 4,000 lbs. and features rotor blades that are 17 feet long [94]. In addition, the landing gear is retractable [88]. The initial engine used was a 300 HP automobile engine, but that will change if this model enters production. The VTOL is able to carry about 300 pounds and can fly for as long as 24 hours without the need to refuel. Figure 3.19 shows Boeing's A160T Hummingbird.



Figure 3.19: Boeing's A160T Hummingbird (source: http://www.news.com/8300-10784_3-7-0.html?keyword=helicopters)

3.5.3 Northrop Grumman's Fire Scout

The Northrop Grumman's Fire Scout, also known as the MQ-8B, is a close air support VTOL and is designed for the U.S. Navy as an anti-submarine craft [94].

The Fire Scout is in the later stages of development; currently, there is interest from the U.S. Army as well as other countries. This unmanned helicopter is able to fly at about 125 knots and is able to reach an altitude of 20,000 feet. It can also carry up to 600 lbs of payload and has an underbelly harpoon that can be used to anchor to a flight deck.

However, it does not have built-in radar which is needed for combat roles. This UAV does not introduce any new or significant features over what is already available, so its future is uncertain. Figure 3.20 shows an example of the Fire Scout VTOL.



Figure 3.20: Northrop Grumman's Fire Scout (source: <http://www.weeklystandard.com/weblogs/TWSFP/2007/06/>)

3.5.4 The SkyTote

The SkyTote, an experimental VTOL, is made by Aerovironment INC located in Monrovia, California [94]. This company also makes endurance UAVs (solar powered). The SkyTote is being developed for the U.S. Air Force. This VTOL is intended as a study for a cargo pickup and delivery vehicle. The SkyTote has a wide number of applications. It is interesting because it sits on its tail end. A Wankel rotary engine is the source of power, allowing for a compact size. Figure 3.21 shows a Sky Tote VTOL.



Figure 3.21: Sky Tote (source: http://www.vectorsite.net/twuav_09.html)

3.5.5 The Boeing Dragonfly (X-50)

The Boeing Dragonfly uses a "canard-rotor wing" (CRW) combined with a narrow fuselage. It features a twin-fin canard wing as well as canard fins on the front part of the VTOL. This plane is able to function like a VTOL and can also fly like a conventional plane. During vertical takeoff, the wing spins much like a helicopter rotor, powered by jet-exhaust tips on the wings. However, in forward flight, it acts like a traditional wing. Boeing is building this plane for possible use by the U.S. Navy and Marine Corps. The first model flew on December 3, 2003 [94]. This VTOL weighs about 1,460 lbs and is approximately 17' 8" inches long.

The X-50 uses a Williams F-112 turbofan with exhaust ducting to allow air to be diverted to the wingtips (during takeoff and landing), towards the rear in forward flight, and to power thrusters that help stabilize the aircraft. The rotor design is of a gimbaled hub type and is able to function as a regular wing when locked in position during forward flight.

There is no need for a tail rotor because the jet-exhaust on the wingtips does not produce torque on the fuselage. Hence, the drive system is simpler; however, it is not very efficient in hover mode. The VTOL is rated at 430 MPH. There are two X-50 models currently in existence [94]. There are also plans for a piloted version. Should the X-50 make it to production, it will set the record as the first ever rotary-wing VTOL (with tip-jets) to enter service. Figure 3.22 shows an example of an X-50.



Figure 3.22: An Example of an X-50.(source: http://www.vectorsite.net/twuav_09.html)

3.5.6 Boeing Aerial Rotor Craft (UCAR)

Boeing is also working on another experimental program for the U.S. government which has the title: "Uninhabited Combat Armed Rotorcraft" (UCAR) [94]. This program was initially called "Robotic Rotary Wingman," and work started in the spring of 2002. The program called for a robotic rotorcraft that can be fully armed and has the ability to

attack targets that are not accessible to the regular (aircraft-based) UAVs such as the Predator. The UCAR production versions were expected to retail between \$4 million to \$8 million. Both Northrop Grumman and Lockheed Martin were able to make it to the final stages of selection and were tasked with making a prototype. The Northrop Grumman UCAR featured dual rotors similar to the one that Kaman helicopters use. This was because Kaman worked on the design with Northrop Grumman.

On the other hand, Lockheed Martin presented a UCAR with a single four-blade rotor that used a NOTAR system in place of a tail rotor. The entries were designed to be "stealthy" and featured internal bays to store weapon systems. The designs featured no tail rotor in order to eliminate as much noise as possible. Both designs can fly at about 160 knots and have a ceiling of 20,000 feet. Figure 3.23 and figure 3.24 show examples of the Unmanned Combat Aerial Rotor Craft (UCAR).



Figure 3.23: Lockheed Martin Unmanned Combat Aerial Rotor Craft (UCAR)
(source: http://www.vectorsite.net/twuav_09.html)



Figure 3.24: Northrop Grumman Unmanned Combat Aerial Rotor Craft (UCAR)
(source: http://www.vectorsite.net/twuav_09.html)

3.6 Helicopter Tail Rotor Types

Most helicopter designs utilize a single main rotor along with a separate rotor to control torque. The anti-torque or tail rotor uses a small variable pitch rotor to control spin. There are variations on these designs; for example, German, British and American designs rotate counter-clockwise (when viewed from the top), while the rest of the designs rotate in the other direction [56]. Helicopters that use the single main rotor design experience torque as a result of the engine spinning the main rotor; this causes the body of the plane to turn in the opposite direction to counter the torque. Figure 3.25 demonstrates the torque produced by the main rotor along with the anti-torque that must be generated by the tail rotor. In order to control spin, an anti-torque mechanism is used to control the body movement of the helicopter and also provide yaw control. There are three common anti-torque mechanisms used currently:

- Traditional tail rotor configuration.
- A Fantail configuration (prevalent in Europe).
- No Tail Rotor (NOTAR) system.

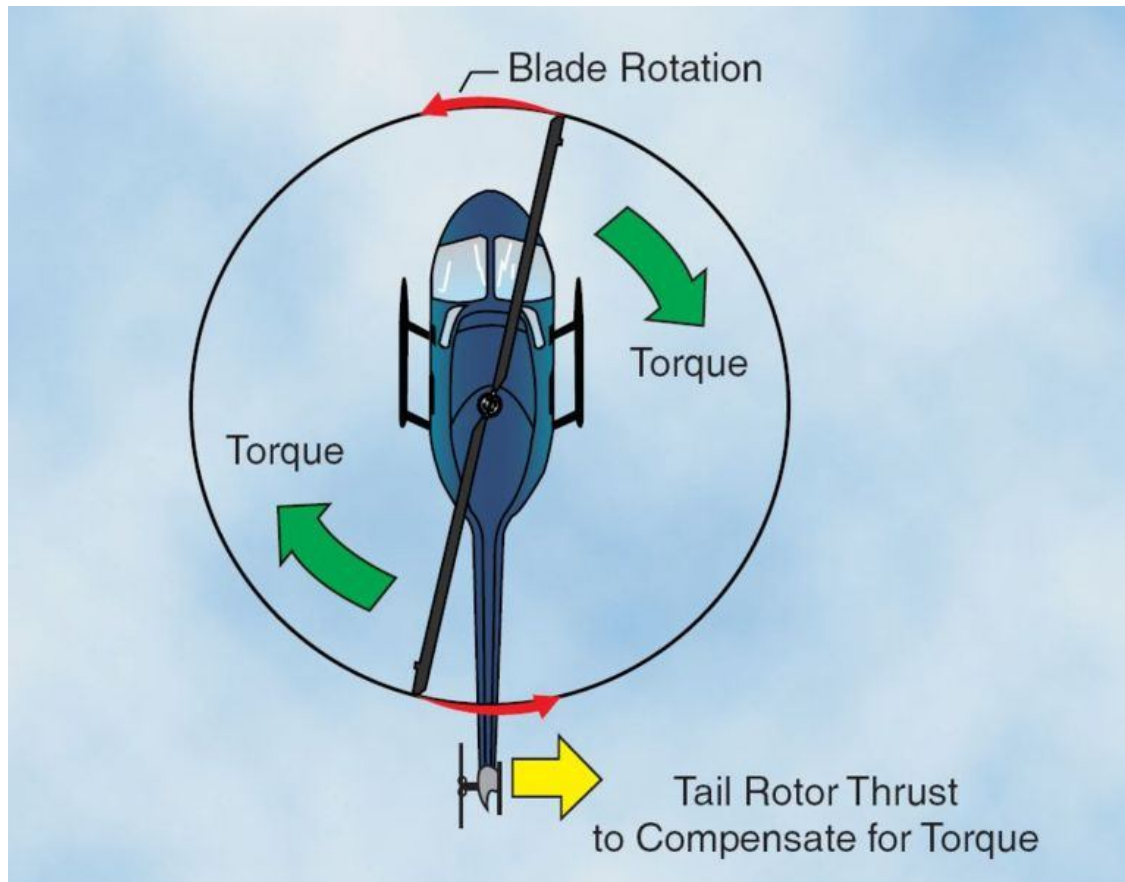


Figure 3.25: Demonstrates the Torque Produced by the Main Rotor Along with the Anti-torque that Must be Generated by the Tail Rotor (source: <http://en.wikipedia.org/wiki/Helicopter>)

3.6.1 Traditional Tail Rotor

Traditional approach to controlling torque uses a small rotor (compared to the main rotor) mounted on the tail in a vertical or semi-vertical fashion (assuming the helicopter is a single main rotor design). The tail rotor counters the affects of torque by either pushing or pulling against the tail of the helicopter. The tail rotor is usually driven by a drive shaft connected to the main transmission and then to gear box located at the end of the tail boom. The drive shaft design may be a single long shaft or multiple shafts connected to each other to allow for more flexibility so it can adjust as the tail boom flexes. The gearbox is used to transfer the power to the tail rotor and, in some configurations, includes gearing to optimize the RPMs of the tail rotor. On large

helicopters, a vertical stabilizer or airfoil may be integrated in the tail design to provide additional torque control. These vertical stabilizers provide some measure of protection should the tail rotor fail.

3.6.2 Fantail Rotor

This configuration is also known as a Fenestron (or Fantail) which consists of a ducted fan that replaces the tail rotor. The fan's housing is built into the tail skin. Another difference lies in the number of blades; the fan uses between 8 to 18 blades while a conventional tail rotor design uses at most five blades. The blades are also arranged in varying distance, so the noise produced will be distributed over a range of frequencies thereby giving the impression of quieter operation than a conventional tail rotor. The housing allows the fan to rotate faster resulting in smaller blades. The Fenestron tail rotor was introduced in the 1960s in Europe, and was used on the SA 340 and later on the Aérospatiale SA 341 Gazelle [56]. Currently the Eurocopter uses this design. The U.S. Army also used a ducted fan on the, now defunct, RAH-66 Comanche helicopter project [56]. Figure 3.26 shows a Fenestron on an EC 120B helicopter.



Figure 3.26: The Fenestron Tail System (source: <http://en.wikipedia.org/wiki/Helicopter>)

3.6.3 No Tail Rotor (NOTAR)

The acronym NOTAR, or “NO Tail Rotor,” is a somewhat new anti-torque control approach. This system was initially developed by Hughes Helicopters. Presently, MD Helicopters uses this system in their planes (no tail rotor is needed with this system) [56]. The concept uses what is called a Coandă effect; this effect refers to the tendency of a fluid stream to attach itself to a convex surface rather than follow its original direction [79]. This effect can be demonstrated by holding a spoon under a stream of flowing water from a tap. Assuming the water is falling straight down, it will change direction to match the convex curvature of the spoon. The NOTAR concept functions in a similar way to an airplane wing developing lift. Except that lift-like force is used to provide anti-torque to the helicopter. The system uses a variable pitch fan which is enclosed in the back of the helicopter fuselage. This fan gets power from the same transmission that powers the main rotor. Figure 3.27 shows a diagram of the NOTAR system.

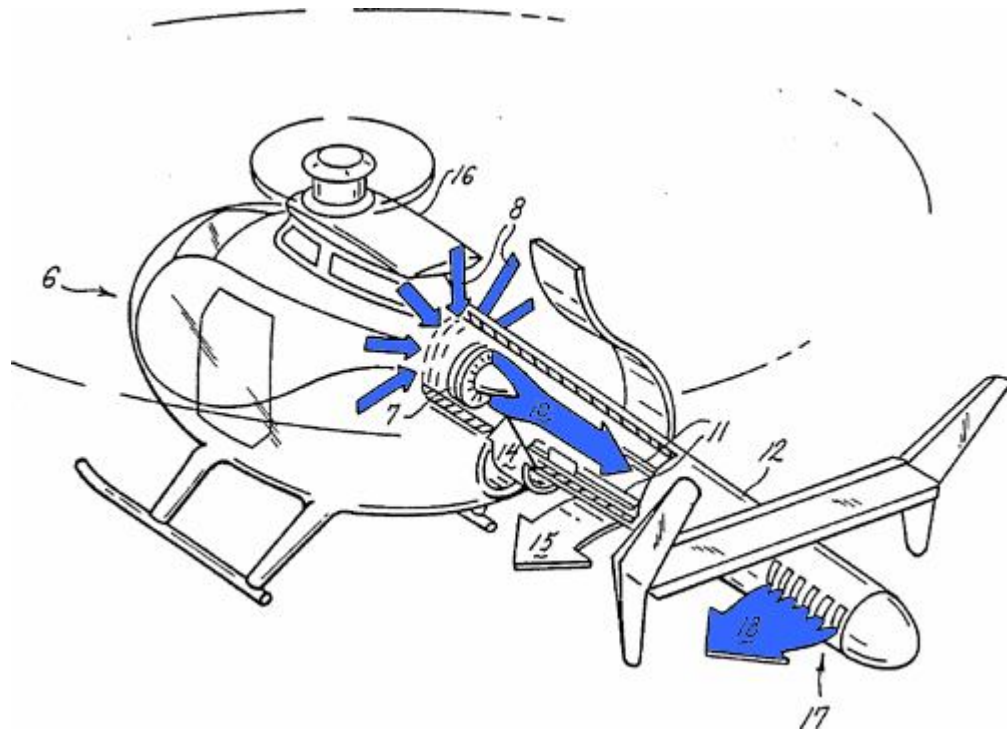


Figure 3.27: The NOTAR System.
 (source: <http://en.wikipedia.org/wiki/Helicopter>)

The fan forces low pressure air through the left side of the tail. The low pressure air is pushed through slots (two or more) on the tail boom. This causes the downwash from the main rotor to follow the surface of the tail boom. The downwash in turn creates lift (in a sideways direction) acting as an anti-torque force. The force is proportional to the amount of airflow. This is reinforced by a directed jet thrust and vertical stabilizers. Work on NOTAR started in 1975 and was pioneered at Hughes Helicopters [56]. The first prototype flew in December 1981. Hues fitted an OH6A helicopter model with NOTAR. Currently there are about three NOTAR helicopters; they are all made by MD Helicopters. Figure 3.28 shows an MD NOTAR helicopter.



*Figure 3.28: MD NOTAR Helicopter.
(source: <http://en.wikipedia.org/wiki/Helicopter>)*

3.7 Two Main Rotor Systems

These helicopter design configurations use two main rotors. The main advantage of these configurations is increased lift. Generally, the need for the tail rotor is eliminated since the main rotors can be configured to counteract each other [56]. The counter-rotating rotor configuration spins the rotors in opposite directions to counteract the torque generated by each rotor. There are three prevalent configurations:

- Tandem rotor configuration: Utilizes two rotors mounted at the front and back of the fuselage.
- Coaxial rotor configuration: Utilizes two rotors mounted on the same vertical axis, one on top of the other.

- Intermeshing rotor configuration: Utilizes two rotors, mounted side to side, at a slight angle to allow the rotors to intermesh.

There is also a side to side configuration that is referred to as transverse rotor configuration. This is usually found on early helicopter designs. Recently, the tilt rotor (Osprey) helicopter is being designed as a transverse rotor. The rotors in this configuration are located at the wing tips or outriggers.

3.7.1 Tandem Rotor Configuration

Tandem rotor configurations use two horizontal main rotors mounted at the front and aft of the fuselage. There is no need for a tail boom, and the rear rotor is usually mounted a little higher than the front. Accelerating and decelerating the helicopter is achieved through “differential collective pitch.” In accelerating mode, the aircraft is pitched forward by increasing the collective pitch on the tail rotor and decreasing the collective pitch on the front rotor at the same time. The result is the tail raising and the front dipping slightly. Deceleration, or moving backwards, is achieved by pitching up the front rotors’ collective pitch. The latter is increased while the back rotor collective pitch is decreased. In order to control the direction of the helicopter (Yaw), opposing cyclic pitch is used. For example to rotate right, the front rotor cyclic pitch is tilted to the right while the rear rotor cyclic pitch is tilted left [56].

3.7.2 Coaxial Rotor Configuration

The coaxial rotor configuration uses a pair of rotors that are designed to turn in opposing directions [56]. However, both rotors are mounted on the same mast and have the same axis of rotation. The coaxial rotors are mounted one on top of the other. The main advantage to this type of configuration is that asymmetrical lift is eliminated [56]. Asymmetrical lift is generated when the advancing half of the rotor generates more lift than the retreating half. This is not a concern while hovering but becomes significant as forward

speed increases. The coaxial configuration eliminates this because each advancing half of the rotors compensates for the lack of lift of the other half. There are also disadvantages to this configuration. The main drawback is the mechanical complexity associated with mounting two rotors on the same mast.

3.7.3 Intermeshing Rotor Configuration

The intermeshing rotor configuration places two rotors side by side on top of the fuselage. The rotors are mounted with a slight angle to the vertical so that the blades can intermesh. The rotors turn in opposite directions. A common name for this configuration is the “synchropter.” The main advantage of this method is the high stability and increased lifting power [56]. One of the early successful synchropters was used by Germany during WWII in anti submarine activities. The helicopter was called the Flettner Fl 282 Kolibri [56]. Kaman, an American company, produced the HH-43 Huskie during the cold War. This helicopter was used by the USAF for rescue missions. Currently, Kaman makes a K-Max model that is designed to be used as a sky crane.

3.7.4 Transverse Rotor Configuration

Transverse rotor design is usually used on helicopter fuselages that look more like regular winged planes. The rotors are located at the end of the wings. In the absence of wings, the rotors are located on structures called outriggers. The transverse rotor configuration depends on differential collective pitch to produce roll. This configuration was used on early helicopter designs, mainly the Focke-Wulf Fw 61 and the Focke-Achgelis Fa 223 [56]. The world's largest helicopter uses this configuration. Figure 3.29 shows the Mil Mi-12 (the biggest helicopter in the world). Recently, this configuration has been used on a new class of helicopter, the tilt rotor; examples of these helicopters include the Bell XV-15 and the V-22 Osprey.



Figure 3.29: The Mil Mi-12 Helicopter.
(source: <http://en.wikipedia.org/wiki/Helicopter>)

3.8 Tip-jet Configuration

This configuration is unorthodox and unusual. This helicopter design needs only a single main rotor powered by nozzles located on the tip of the rotor blade. The nozzles are pressurized or small engine (turbines) can be located on the wingtips. This design has been proven to work, does not generate torque; however, it is not as fuel efficient as regular helicopter designs. It can also be very noisy. Examples of helicopters that used tip-jet configuration include the Percival P.74, Hiller YH-32 Hornet; both were not very functional. However, the Fairey Jet Gyrodyne and the Fairey Rotodyne (40 passengers) were both good flyers [56]. Another concept was called the Rotary Rocket Roton ATV which had rockets on the tips of the wings. There are no Tip-jet configurations in production, and all of the jet-tip designs never made it past the prototype stage.

3.9 The Rotor System

The main rotor is the part of the helicopter that is responsible for generating lift. It consists of a hub, a mast, and two or more rotor blades mounted on the mast. As the blades rotate, they generate lift. The main rotor is mounted horizontally, while the tail rotor is mounted vertically or semi-vertically. The main function of the tail rotor is to control the torque produced by the main rotor. Helicopters with two main rotors do not need a tail rotor as anti-torque control is achieved by having the two main rotors spin in opposite directions. The mast is essentially a shaft (made of metal) that has a cylindrical shape. The mast connects the rotor to the transmission. At the top of the mast, the rotor blades are attached to an anchor point called the hub [56]. In main rotors, the methods by which the blades are connected to the mast are classified as follows:

- Semi rigid.
- Rigid.
- Fully articulated.

3.9.1 Rigid Rotor System

In this configuration, the hub and mast are rigid, and no movement is allowed between them. This rotor system has the advantage of mechanical simplicity over fully articulated rotor systems [56]. This system does not have vertical or horizontal hinges; hence, the blades do not have the ability flap or drag. However, the blades can be feathered [56]. During helicopter operation, the loads generated by flapping and lead/lag have to be absorbed by the bending of the blades since there are no hinges. This system requires blades that can flex, thereby absorbing forces which would normally require hinges. The system exhibits less lag in control response due to its rigidity. There is also an increased safety margin in that the rigid rotor system makes the chance of mast bumping minimal. Mast bumping is a concern in semi-rigid rotors [56].

3.9.2 Semi rigid Rotor System

The semi rigid rotor system is designed to allow flapping and feathering. The majority of semi rigid rotor systems use two blades that are rigidly attached to the rotor hub [56]. The hub is attached to the mast by a trunnion bearing (teetering hinge). This attachment method allows the hub to tilt freely in relation to the main rotor shaft. The result is blades that “see-saw,” so when one blade flaps up, the other blade flaps down. Feathering is achieved by the use of a feathering hinge; this hinge works by changing the blade pitch angle. Due to the absence of a vertical drag hinge, the lead-lag forces must be absorbed by blade bending. Helicopters that use the semi-rigid rotor configuration are susceptible to “mast bumping.” This can cause the stops of the rotor flap to shear the mast resulting in an unsafe situation. This condition is most likely to occur during low-G maneuvers; hence, operations manuals for these helicopters warn about low-G conditions.

3.9.3 Fully Articulated Rotor System

This type of rotor system attaches each rotor blade to the rotor hub using a series of hinges. These hinges allow each blade independent movement. These systems normally use three or more rotor blades. Because the blades are fully independent, they can flap, feather, lead, or lag individually. The system uses a flapping hinge (horizontal hinge) which enables up and down movement of the blade. The flapping movement is designed as a mechanism to compensate for asymmetrical lift [56]. The system uses a lead-lag (drag) hinge or vertical hinge to enable the rotor blade to move back and forth and is usually attached to dampers to limit the amount of movement. The drag hinge allows the rotor system to counteract the Coriolis Effect which is caused by acceleration and deceleration. The Coriolis Effect is defined as “the apparent deflection of moving objects from a straight path when they are viewed from a rotating frame of reference” [95]. The system also allows for feathering or changing the pitch angle of the blades to control the thrust of the main rotor [56].

3.9.4 Modern Rotor Configurations

Modern rotor systems may combine the above rotor systems to achieve the desired design characteristics of a new helicopter. Some types of rotor hubs may use a flexible hub that allows the blades to bend without requiring hinges. These flexing configurations are called flextures and use composite materials in blade construction. In addition, there is a new type of bearing called an elastomeric bearing. These bearings are being used to replace the standard roller bearings in newer helicopter designs [56]. Since elastomeric bearings are made from a rubber compound, they offer limited movement that has excellent suitability for this type of application. The use of flextures and elastomeric bearings means that these parts require no lubrication; the latter means less maintenance. These designs have the added advantage of having vibration absorption characteristics which translates into less stress and extends the service life of helicopter components.

3.10 Helicopter Flight Dynamics

In order to achieve controllable flight, an aircraft must have three dimensional controls. Fixed-wing aircraft achieve this by using small movable surfaces to change the flow dynamic of the air passing around it. This process essentially changes the plane's shape achieving directional control [56]. Helicopters, on the other hand, have used a different method. In order to change pitch (forward or backwards) and roll (tilting the helicopter side to side), the helicopter must change the angle of attack on the main rotor blades. The process of changing the angle of attack on the main rotor is called cycling [56]. The change in pitch allows for changing the amount of lift at different points in the rotor cycle.

3.10.1 Dynamic Control with Single Main Rotor

In order to achieve control on the vertical axis, the tail rotor is used. This is referred to as yaw control. Yaw control is achieved by varying the pitch of the tail rotor which in

turn changes the amount of sideways thrust. On most helicopters, yaw is controlled with foot pedals. This corresponds to a fixed-wing plane's rudder control pedals.

The collective pitch, or collective for short, is the control that adjusts the angle of attack on the rotor blades [56]. A larger angle of attack means more lift. A smaller angle of attack will provide less lift. This control consists of a lever that is usually located on the left side of the pilot. Helicopters take off when the collective is increased and the throttle is given power. The throttle control varies the power output of the helicopter engine. The engine is connected to the main rotor shaft through a transmission assembly. The throttle control is located on the top of the collective control lever. It is much like a motorcycle power controller; it is a grip that can be twisted to add power. This throttle allows the pilot to control the rotational speed of the engine (RPM). This is important because helicopters are designed to run at certain RPMs. There is usually a chart for each helicopter model that shows the optimal RPM for a given load (weight) and speed. Too low of an RPM can cause a stall condition much like a low airspeed can cause a stall in fixed wing aircraft [56]. High RPMs can cause damage to the main rotor hub as well as the drive train. Helicopters require that the RPM be maintained within a narrow range for optimal and safe operation. Figure 3.30 shows a summary of stick and pedal controls on a helicopter.

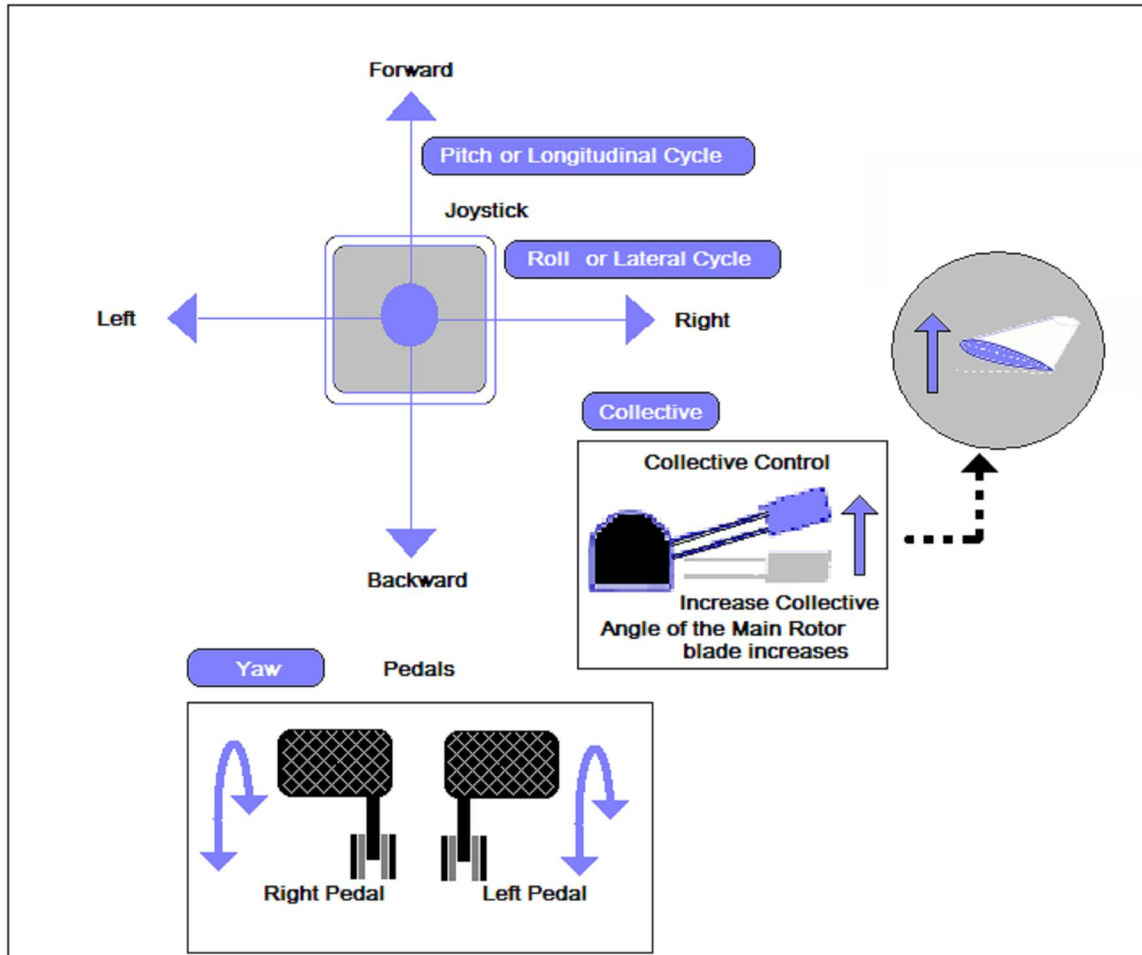


Figure 3.30: Stick and Pedal Controls.

The cyclic pitch control is used to adjust the rotor blade pitch in a cyclic fashion. This control, also called the cyclic, allows lift to be varied over the rotating blades so that the blades produce more lift on one side than on the other [80]. This variation in lift over the rotor disk plane of rotation allows the rotor disc to tilt resulting in range of motions. It also allows the pilot to change altitude when flying in a forward direction. This control is much like a joystick and is placed directly in front of the pilot. The cyclic allows the pilot to control the angle of the rotating section of the swashplate [56]. When the swashplate has no tilt, then all the helicopter blades are set to the collective angle. However, when the swashplate is tilted, there is a pitch-up at a given azimuth angle with an opposing pitch-down of the rotor directly across. This creates a sinusoidal variation in the angle of attack on the rotor blades which makes the helicopter tilt in same direction as the cyclic. For

example, if the cyclic is pushed forward, then the rotor disk is tilted in that same direction making the helicopter move forward. The pitch and roll movements are shown in figure 3.31.

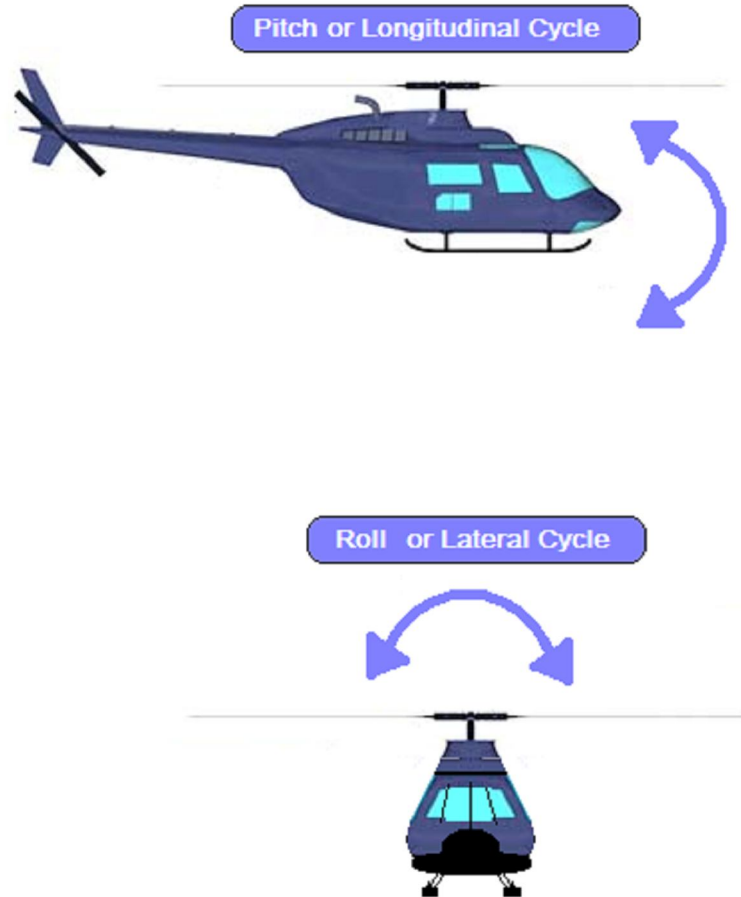


Figure 3.31: Pitch and Roll Movements.

3.11 Helicopter Aerodynamic Considerations

When a helicopter moves through the air, the rotor speed relative to the forward velocity becomes an important operational consideration [80]. The forward motion of the helicopter causes one side of the rotor blade to move at a speed equal to the rotor tip speed plus aircraft speed. This results in higher lift on the advancing blades [56]. However, on the

other side of the helicopter, the retreating rotor blades move at rotor tip speed minus the aircraft speed. This results in a loss of lift on the retreating side of the rotor. The asymmetry of lift becomes more pronounced as the speed increases. To compensate for this condition, lift is decreased on the advancing blades and increased on the retreating blades by adjusting the angle of attack. Figure 3.32 illustrates the progressive loss of lift as the helicopter reaches its maximum allowable safe airspeed.

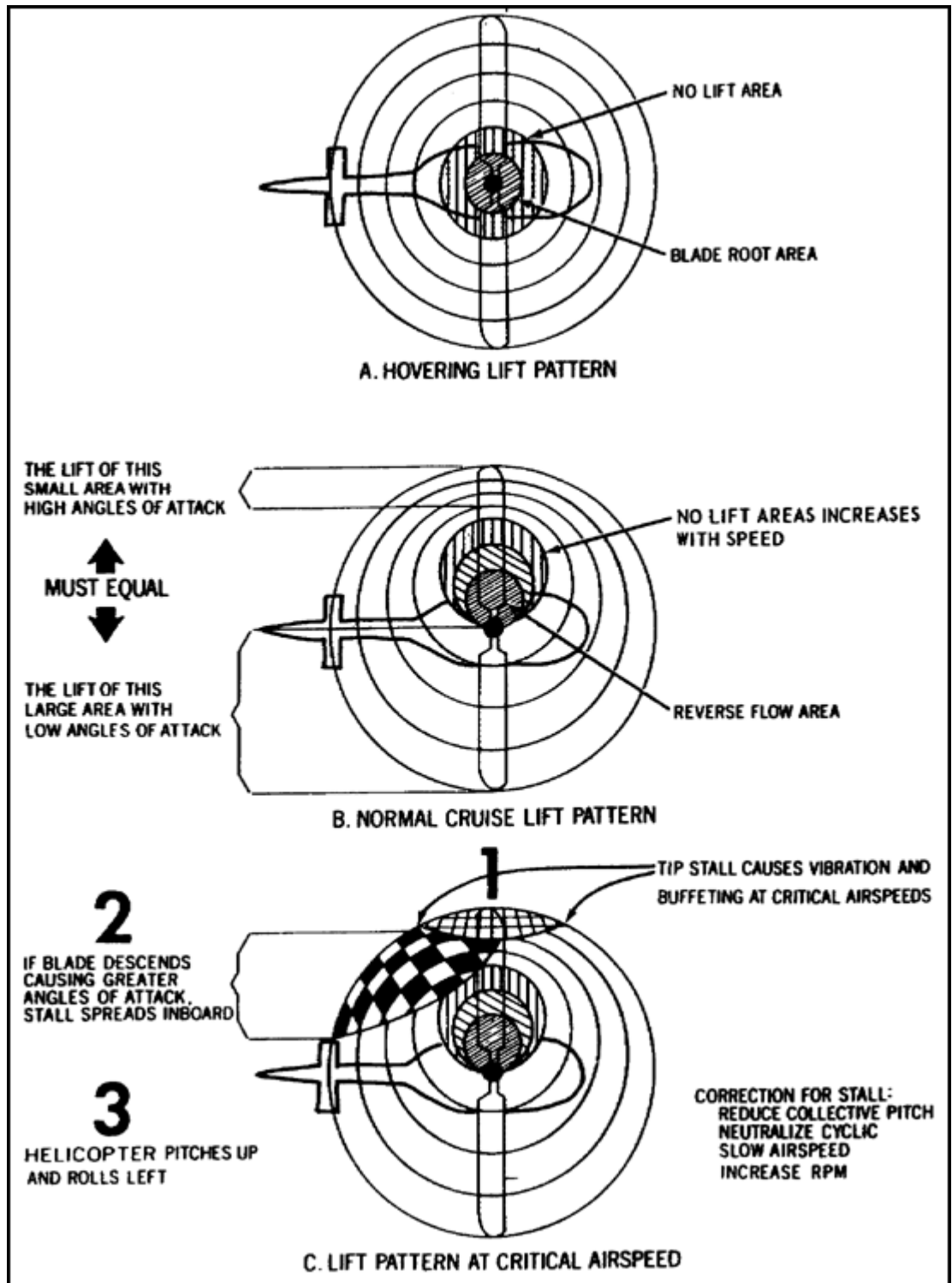


Figure 3.32: A Chart Showing the Progressive Loss of Lift as the Helicopter Approaches Critical Speeds. (source: <http://www.globalsecurity.org/military/library/policy/army/accp/at0966/le3.htm>)

However, there is a limit for lift adjustment; should the angle of attack of any airfoil or wing become too high, then the airflow over it may separate and cause a sudden loss of lift. The latter results in what is called aerodynamic stall. There are four possible scenarios in which this can happen:

- Should the helicopter reach higher speeds, the airflow around the advancing blades can reach speeds close to that of the speed of sound. This generates shock waves on the rotor that interrupts the airflow flowing over them resulting in loss of lift.
- Higher speeds also cause the retreating blade to lose lift. This occurs because of the lower relative airspeed of the blades (since it is moving opposite to the direction of travel). Helicopters correct this lower lift condition by increasing the angle of attack on the blades on the retreating side. However, too high an angle can result in aerodynamic stall or “retreating blade stall” in this case.
- If the main rotor RPM becomes too low along with an increase in collective pitch, this will result in aerodynamic stall.
- The “vortex ring state” is a condition that is unique to helicopters; this condition is also called “settling with power.” This can occur when a helicopter is descending or hovering and happens to contact with its own down-wash. This can cause massive amounts of turbulence resulting in loss of lift.

3.11.1 The Autorotation Concept

Many powered aircraft can glide safely to the ground without power. Fixed wing aircraft are able to glide using lift generated from their wings in a controlled fall. Helicopters are also powered aircraft; however, they have limitations in gliding. This is achieved by using the inertia in the rotors; this has to be combined with a downward glide forcing the air to flow through them [56]. In this operation, the main rotor works like a windmill, turning as the air passes through it. This approach is autorotation; it makes it possible to survive instances where there is a sudden loss of power (provided that the drive train is intact). Since the tail rotor is connected to the main rotor by the transmission,

spinning the main rotor by autorotation provides all flight controls even though there is no engine power. However, autorotation can only be performed within certain height and speed ranges; outside of these ranges this procedure is not possible. Figure 3.33 shows a height-velocity diagram.

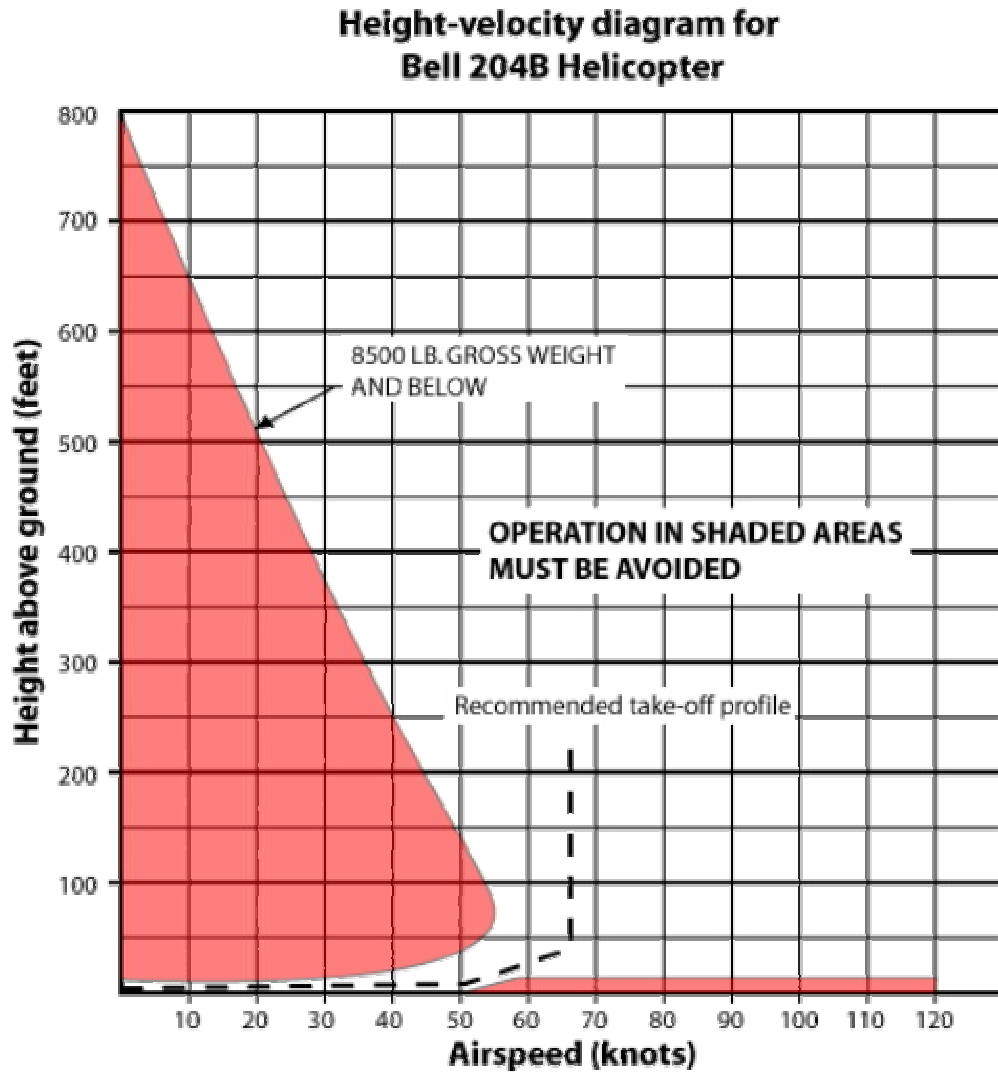


Figure 3.33: The Height Velocity Diagram. (source: http://en.wikipedia.org/wiki/Height-velocity_diagram)

3.11.2 Stability

Fixed wing aircraft designs are usually very stable. If a fixed-wing aircraft experiences a gust of wind or if the pilot changes the controls to cause a sudden yaw, roll, or other motion, then the design properties of the plane will counteract the movement and return the aircraft to its original heading [56]. Also fixed wing aircraft designs have enough built-in stability that constant pilot input is not needed. The plane tends to fly itself for the most part. However, this is not the case with helicopters which are, inherently, not stable. In order to just hover, the pilot must provide continuous input and make adjustments to the flight surfaces to keep the helicopter from drifting. Because of the myriad of forces that act on a helicopter, a net force (such as a wind gust) will cause the helicopter to move (and stay moving) in that direction [81]. Furthermore, the cyclic must be adjusted to continue hovering in the same location.

Helicopter flight controls are affected by one another; for example, if the cyclic is changed to pitch forward (so that the helicopter can move forward), this will result in some loss of lift which must be compensated for by increasing the collective to generate more lift. However, increasing the collective results in slower rotor speed; thus, in order to maintain the correct rotor RPM, the throttle must be increased. Changes in the collective setting will also result in power changes (changing main rotor torque); this means that the pilot must compensate by adjusting the tail rotor using the foot pedals. Small helicopters are inherently very unstable, making them require constant cyclic adjustment in flight [56].

3.11.3 Helicopter Limitations

The main limitation of helicopters is their relatively slow speed. The fastest production helicopter in the world, the Westland (modified) G-Lynx, can only fly about 250 MPH [81]. Most production helicopters rate between 150 to 200 MPH. The theoretical top speed for a helicopter is 250 MPH [81]. The factors that contribute to the helicopter's limited speed are as follows:

- If the helicopter is hovering, then the speed of the rotor blade's outer tips is the result of two variables: the RPM and length of the rotor blade. However, when the helicopter is moving through the air, a third variable comes into play when determining the rotor blade speed. That variable is the forward velocity of the helicopter. The difference between the speed of the helicopter and the rotational velocity of the blades becomes significant at higher forward velocities. This means that advancing rotor blade speed is the sum of the speed of the rotor blade at the tip and the forward velocity of the helicopter. Under certain conditions, the rotor blade's tip may exceed the speed of sound resulting in shock waves. These shock waves disrupt the airflow over the blades to the extent that they cause stalling, increase drag, and may cause excessive vibration. One possible solution to this condition is to use spiraling rotors [56]. However, current technology and metallurgy does not provide the materials needed to construct spiraling rotors.
- Rotor blades are designed to be flexible; this is done so that they flex with the direction of travel of the rotor. That means they should provide more lift at slower speeds. This comes into play because of the asymmetrical lift generated as the helicopter moves forward. When the rotor blade moves in the same direction as the helicopter, it is moving faster through the air. However, when the blade spins away from the direction of travel, then its speed is slower in relation to the air around it. The blade is designed to twist and lift with speed such that an advancing blade has a smaller angle of attack while the retreating blade develops more lift because it flaps down producing a higher angle of attack. At high speeds, excessive flap may develop while the retreating blade may stall due to a very high angle of attack. Helicopter designers rate helicopters on their "maximum safe forward speed" with a "Velocity, Never Exceed" or VNE rating.
- The design of the rotor head is another limiting factor on flight dynamics. Flight maneuvers that generate negative-G conditions are an issue in a semi-rigid rotor system. This will cause the blades to flap down and strike the airframe or tail boom resulting in a crash.

- Another concern with helicopters is the vortex ring effect [56]. This condition is sometimes caused by the downward wind from the rotor blade creating a circular vortex around the rotor blades. Terrain, rain, wind, and other factors can affect how this ring is formed and may cause the helicopter to lose some lift. If enough lift is lost, a condition called “setting with power” ensues which will cause the helicopter to hit the ground.

3.12 Noise and Vibration

Helicopters can be noisy, coupled with the fact that they fly relatively close to the ground and over cities which makes noise control an important concern. New designs to limit helicopter noise were due to heliports being closed and flight constraints over some national parks.

Helicopters produce vibrations as they run; an unadjusted helicopter can produce massive amounts of vibrations that result in mechanical failure. In order to deal with vibration, helicopter rotors can be adjusted for height and pitch [56]. There are also dampers that control vibration for height and pitch. There are also mechanical systems that depend on feedback to control vibration. However, vibration measurement is difficult. A common way to adjust rotors is to use a strobe flash to determine if the painted markings on rotors are aligned.

CHAPTER 4

THE PROPOSED AUTOPILOT

4.1 The Autopilot Model

The autopilot derived in this dissertation is designed to be a component of a larger system. The pilot may or may not be physically located on the helicopter. Figure 4.1 shows an example of a human pilot issuing control commands (mechanically) using a joystick and receiving feedback from the helicopter (gauges). The feedback conveys the current state of the helicopter allowing the pilot to issue new commands based on the current state of the machine.

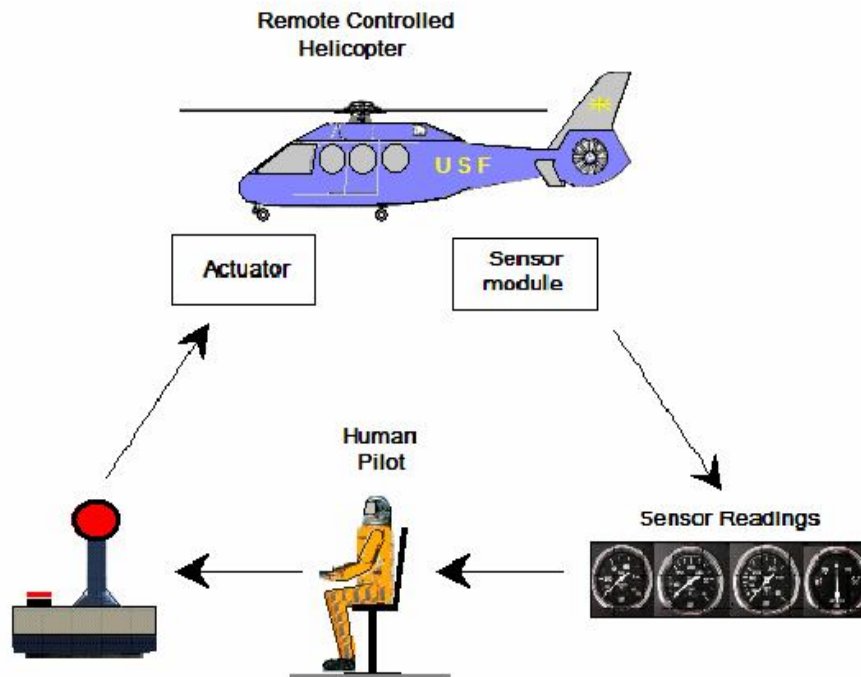


Figure 4.1: A Human Pilot Controlling a Helicopter Using a Joystick and Sensors.

However, the helicopter can also be controlled with an autopilot. Figure 4.2 shows a Proportional, Integral, Derivative (PID) controller, a fuzzy logic controller (Fuzzy), or a human pilot using an interface (electronic, mechanical or other) to send commands to a helicopter and receives feedback. Feedback can be in an electronic, mechanical, or other form; however, it is assumed that, regardless of signal format, control and feedback signals can be measured and recorded. The pilot is also given set-point data that convey the desired destination(s). The set-point data denote points in space that specify the desired flight path. The pilot (PID or otherwise) would then use the set-point data and the feedback from the helicopter to determine the next set of commands to issue. The loop continues until the helicopter has executed the desired flight path. The process of deriving a control module starts with the “observation” of a pilot be it a PID controller, Fuzzy controller, human, or other control technology executing a maneuver on a small unmanned helicopter as shown in figure 4.2.

4.2 The Proposed Autopilot

Figure 4.3 demonstrates the proposed pilot interfacing with the helicopter. The auto pilot proposed in this dissertation would interface directly with sensors on the helicopter and relay commands directly to the helicopter controls. Note the lack of set-point data input as the proposed pilot does not need this information. The proposed pilot will be an Artificial Intelligence (AI) derived set of control equations. The “observed” data is collected from a Fuzzy pilot flying a pre-selected set of maneuvers. Once data collection is complete, the information is then fed to a GA/SA based algorithm that generates control functions. These functions consist of specifically selected mathematical building blocks that are designed to recreate the “observed” output (control values to the helicopter) when given flight time (t). Hence, at $t=0$, the derived functions should return the correct control signal. Likewise, when the functions are given $t=n$, they must return the control signal value initially observed at time n . Figure 4.4 shows a table of “observed” values used by a GA program to derive a set of functions that are, in turn, used to control a helicopter. Note that the actual algorithm used is a GA/SA hybrid.

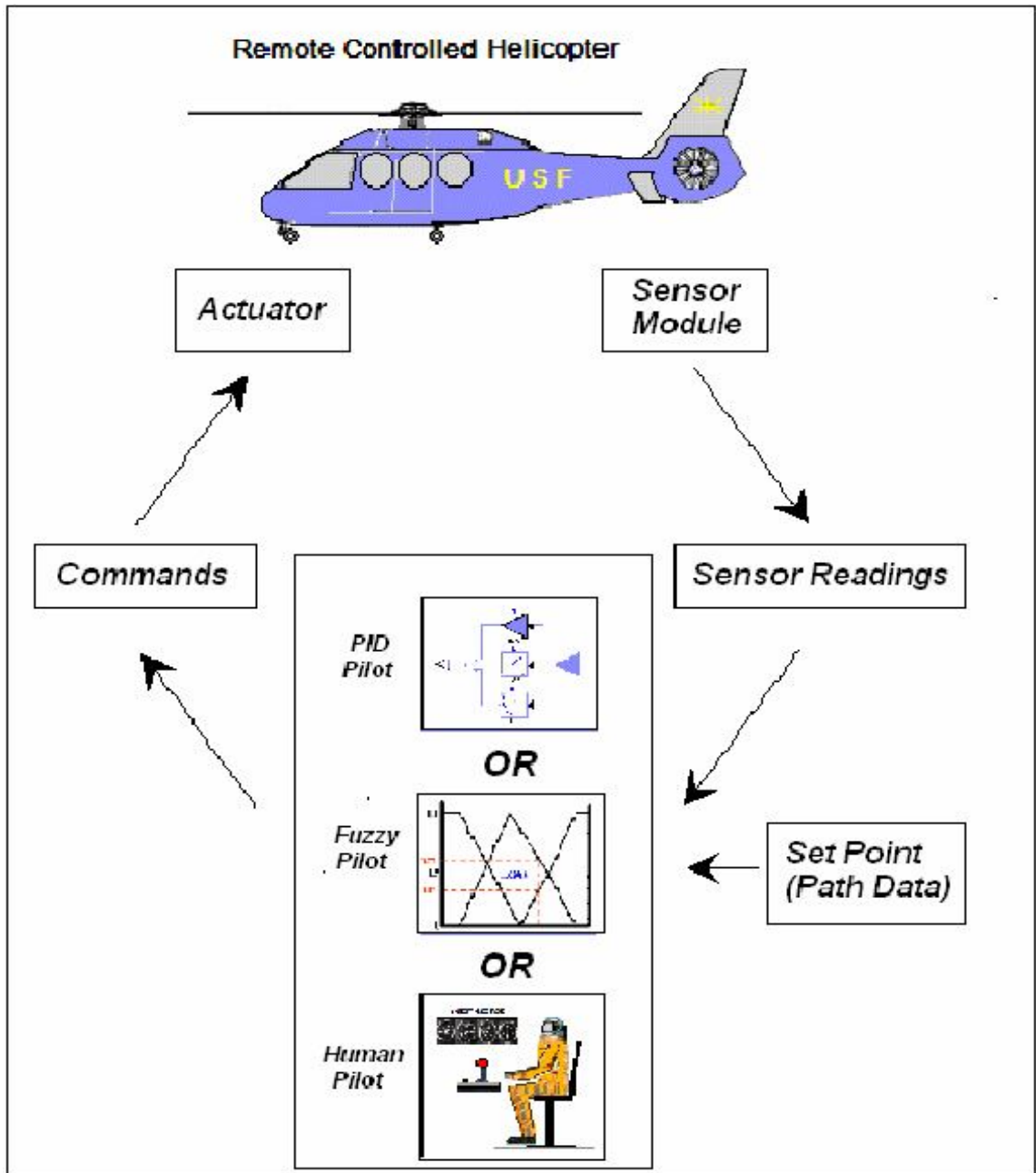


Figure 4.2: A PID or Fuzzy or Human Pilot Controlling a Remote Helicopter.

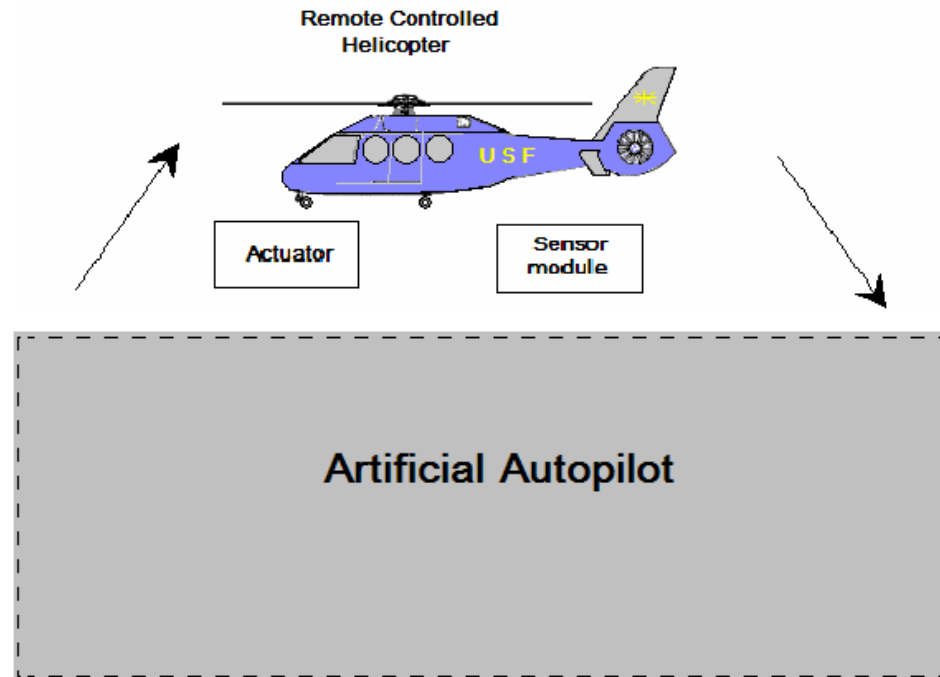


Figure 4.3: Proposed Auto Pilot Module Controlling a Remote Helicopter.

4.3 Collecting ‘Observed’ Data

Figure 4.5 demonstrates the method used to collect data. A recording module is placed on the input and output side of the helicopter. Next, the Fuzzy controller executes a pre-determined maneuver. The maneuver is timed from the start to completion. This process can be repeated to collect data for additional maneuvers.

4.3.1 Observed Data Source

The source for the “observed” data is a virtual helicopter created in MATLAB. The virtual helicopter takes four control (input) parameters and generates five outputs as shown in figure 4.6. The input parameters are the collective, pedal, longitudinal, and lateral. The outputs are the x-axis location, the y-axis location, the z-axis location, the yaw reading, and the elapsed time. This data is used in the derivation and the testing of the GA/SA pilot. The virtual helicopter was coded by a group of USF students to test

automated pilots including a Fuzzy controller. The details on creating the helicopter model are outside the scope of this dissertation.

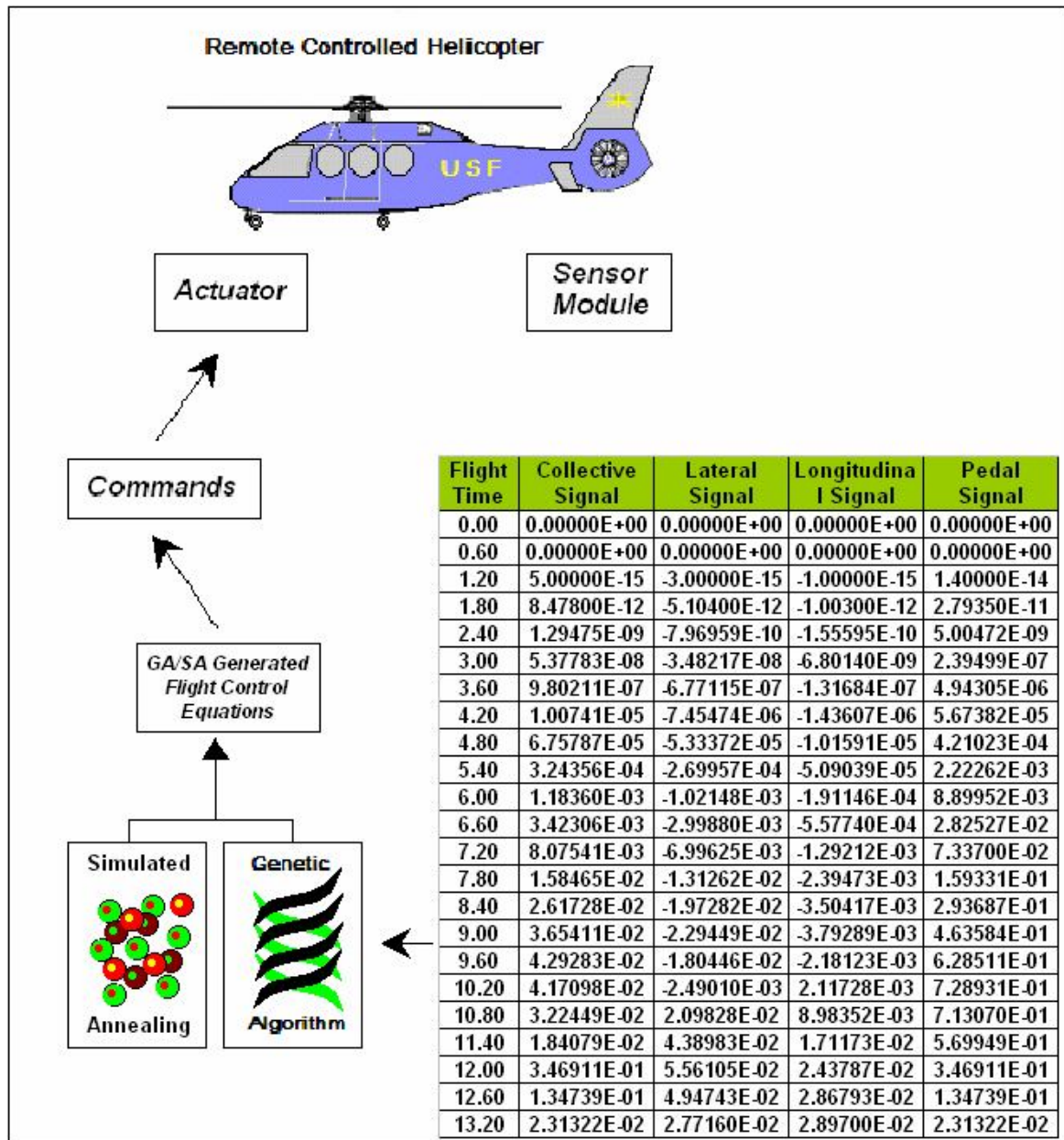


Figure 4.4: GA/SA Derived Math Equation Controlling a Remote Helicopter.

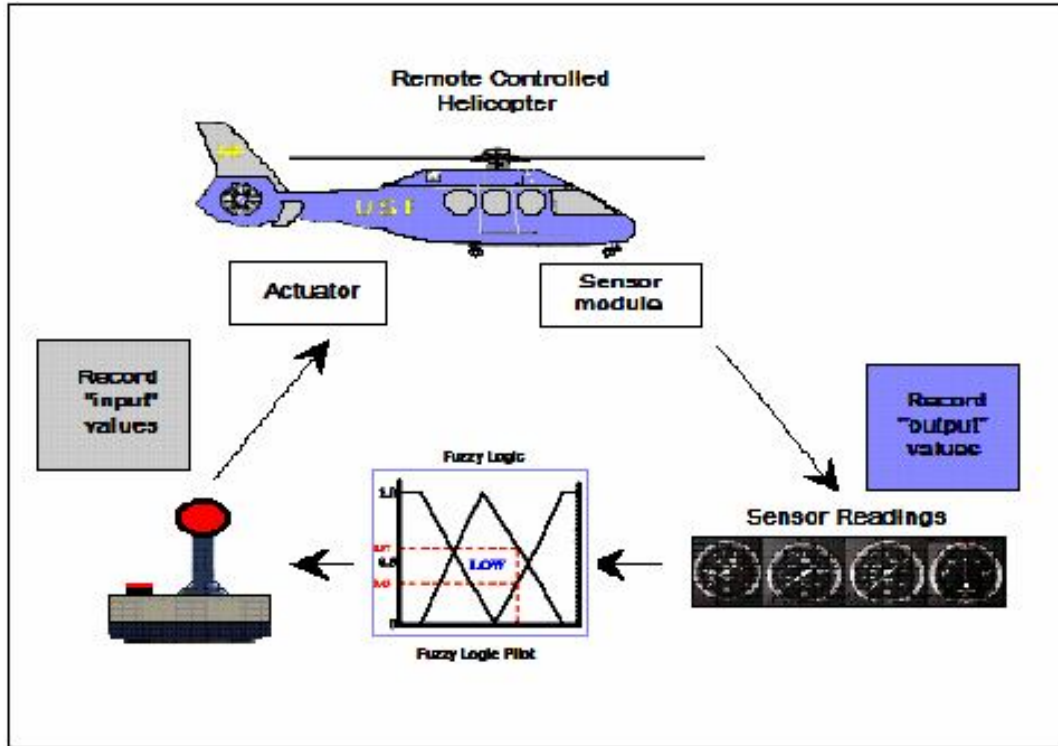


Figure 4.5: Collecting Training Data for the GA/SA Search Algorithm.

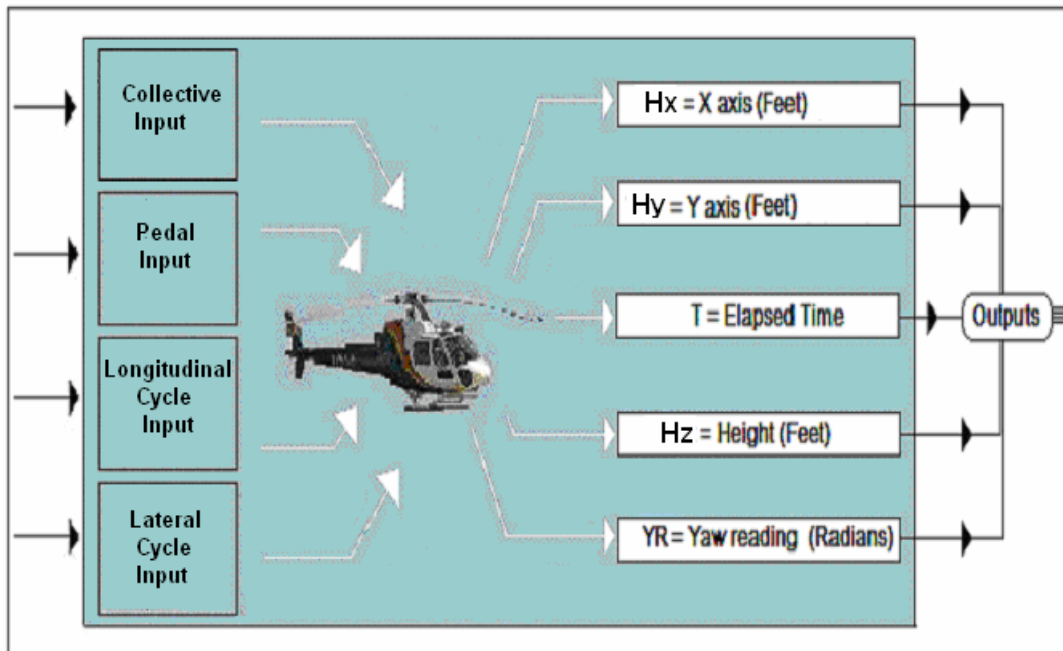


Figure 4.6: Virtual Helicopter Input/Output Parameters.

4.3.2 The Control Equations

First, observed input data and flight time are provided to the GA/SA search algorithm as shown in figure 4.4. The GA/SA search algorithm generates control equations, one per control signal. The four equations comprise the proposed auto-pilot and are then used to control the helicopter on their own given flight time as the only input. Figure 4.7 shows how the four control equations fly the helicopter.

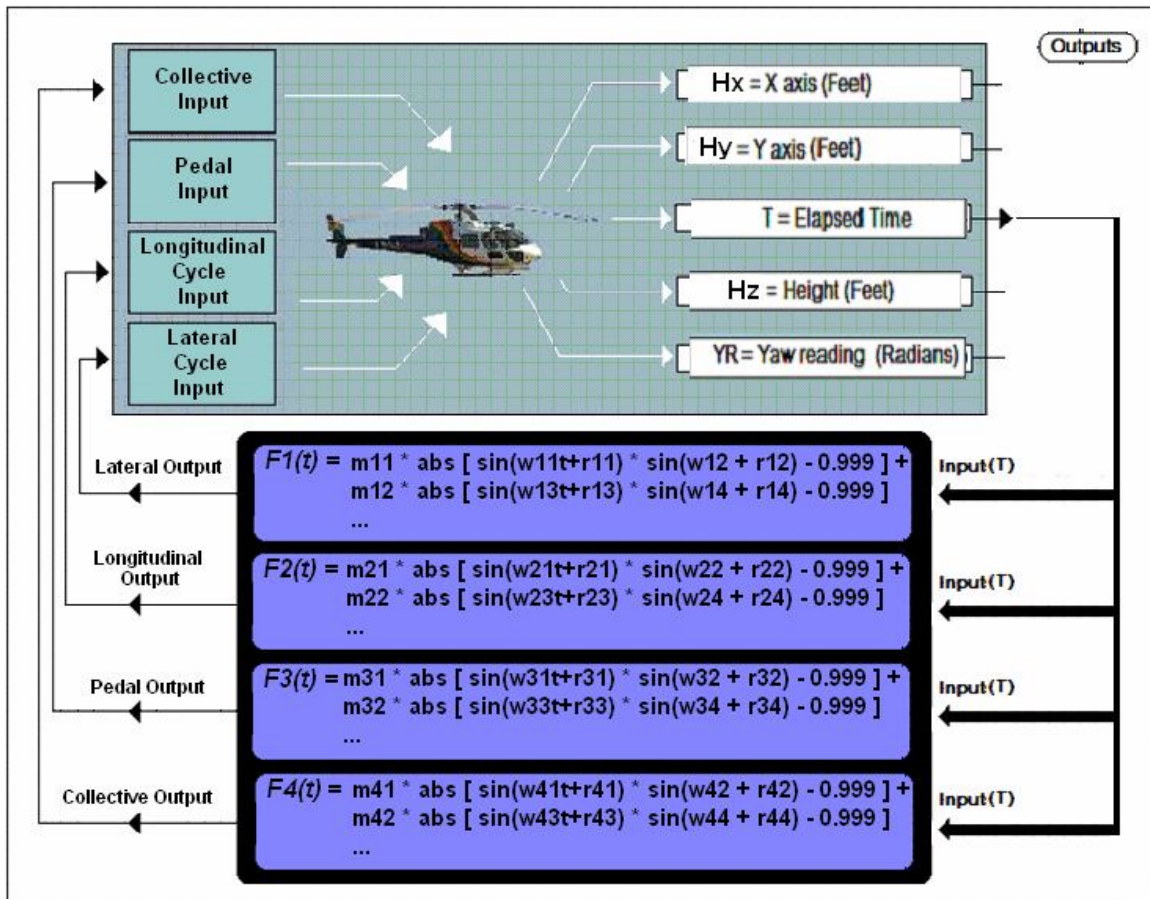


Figure 4.7: Control Equations flying the virtual helicopter.

4.4 The Autopilot and Surrounding Architecture

It is assumed that the auto-pilot module will be embedded in control architecture. Figure 4.7 shows a typical AI control architecture based on subsumption. This architecture is based on breaking down complex intelligent behaviors into a series of behavioral units each tasked with implementing a specific goal. The layers are generally laid out from reactive (basic response) to more deliberative layers that require complex processing [96]. For example, a robot's most primitive subsumption based layer is to back out if it hits an object. The more complex layers deal with long range goals such as avoiding the same obstacle to higher layers that deal with completing the mission. Generally, the layers get more abstract as they move from reactive to more deliberative layers [96]. Therefore, each layer is designed to subsume the behavior of the layers below it. As such, "reach a goal" layer must be placed above a more primitive layer that keeps control of the plane. The latter layer is itself placed over a more primitive layer which avoids any situations that can lead to a crash. The main advantage of this paradigm is that separate layers can over rule other layers. As such, the most reactive layers work much like reflexes while the more deliberative layers are pre-occupied with reaching the goal. The proposed pilot can be located parallel to the main pilot. Should the main pilot or GPS experience a failure, the proposed pilot can take over and complete the current maneuver or execute another maneuver such as "return to base."

4.4.1 Helicopter Control Architecture

Figure 4.7 demonstrates the hierarchy of control modules (white boxes) from most deliberative to most reactive; the most deliberative module is that of the mission planner. It is responsible for understanding the mission goal and communicating the mission parameters in a format that the navigator module can use. The lower level navigator then generates a set of flight segments that, together, complete the mission; the navigator must ensure that each segment is executed properly by the pilot module. Hence, the navigator sends the pilot module a set of specific actions. The low level auto-pilot module is reactive, concerned mainly with executing a given flight maneuver as

accurately as possible. The lowest, and strictly reactive, module in the control hierarchy is the Safety System; this module is tasked with insuring that commands from the auto-pilot module do not exceed the physical limitations of the helicopter or result in an unsafe helicopter flight state. If the pilot module issues control signals that exceed the safe flight envelope, then the Safety System would override or trim these signals so that the resulting signals are “safe” and do not lead to loss of control.

4.4.2 Pilot Module Architecture

A detailed block diagram of the auto-pilot module is shown in Figure 4.8. The pilot module is interconnected to the Navigator module, the GPS module, the Monitoring System, the Safety System, and most importantly there are interfaces that provide it sensor readings and outputs to the helicopter flight controls. The Safety System module is essentially transparent to the pilot module. The navigator sends commands to the pilot module; these commands instruct the pilot to execute a single maneuver. The monitoring system in conjunction with GPS help the navigation module determines if a flight maneuver was executed correctly (meaning that the helicopter is now where the navigator expected it to be after completing the flight maneuver). In case of a discrepancy, the system reacts by signaling with an error code. The system uses GPS readings to determine the error level. The error signals as depicted in Figure 4.8 demonstrate how errors are communicated. The external modules presented here are “assumed,” and their exact function is outside the scope of this research which is concerned with learning the actions of a pilot executing a specific maneuver.

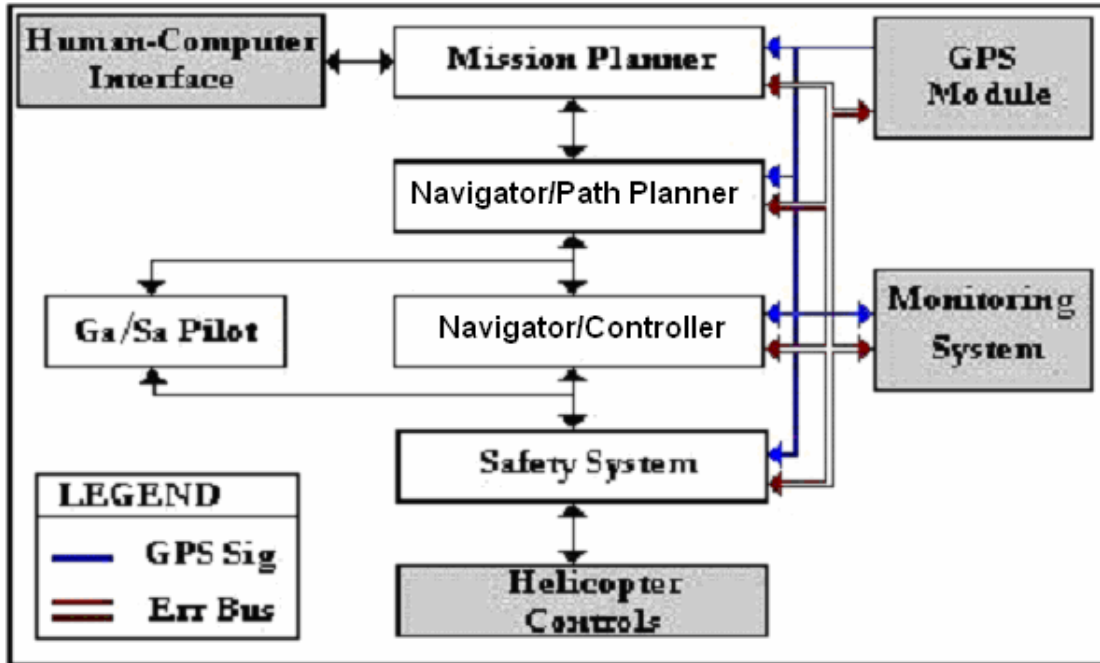


Figure 4.8: Helicopter Control Architecture.

4.4.3 Pilot Module Behavior

The pilot module is assumed to have some general structure that supports error detection; a possible example of a pilot module is shown in figure 4.9. Note that the main pilot is able to detect an error and try to recover. However, if recovery is not possible due to a GPS or sensor fault, the pilot should be able to pass control to a backup pilot. In this case, it will be the GA/SA pilot that functions independently of location data (GPS) and set-point data. The figure shows the main pilot executing a loop that checks if each command is executed successfully. For example, the right turn command is issued by the Navigator layer and is then executed by the main pilot. If the maneuver runs correctly, then the main pilot is ready for the next command. If there is an error, then an error recovery is attempted. If the pilot is unable to recover from the error after a (predetermined) number of attempts, then it has the option of passing the control to the backup pilot. It is assumed that the safety system has built in stabilization such as an autogyro or a similar system. This built in stabilization automatically corrects for wind gusts, wind direction, and other external weather conditions that can change a helicopter's heading. The stabilization system is important since the GA/SA pilot flies in

open-loop. The latter will insure that the control formulas are useful regardless of wind-direction.

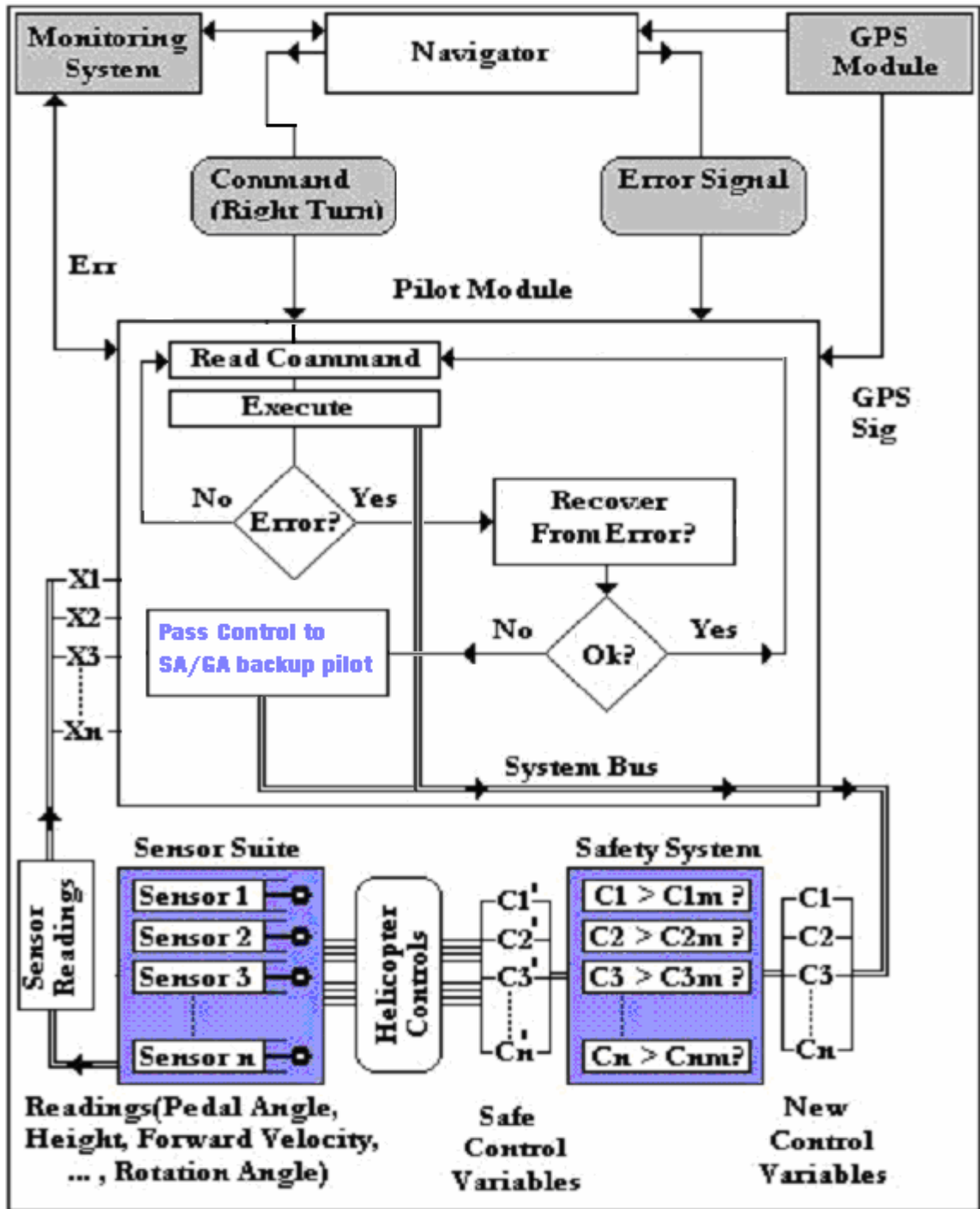


Figure 4.9: An Example of an Autopilot Module with Error Detection.

CHAPTER 5

TECHNIQUE OF IMPLEMENTING A SOLUTION

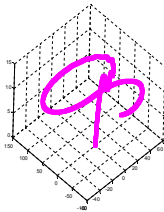
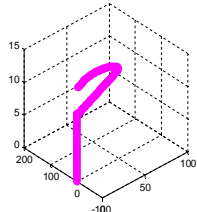
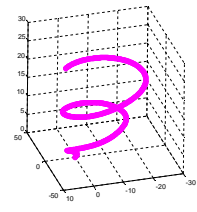
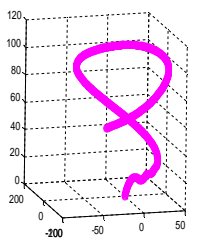
5.1 Summary of Search Goals

The goal of this dissertation is to generate a set of control formulas that will act as an autopilot and fly a pre-selected maneuver. A GA/SA search algorithm will be used to create the control formulas using helicopter control signals and flight time. If the proper set of mathematical functions are selected and used as building blocks, then the GA/SA should be able to generate control formulas. A set of four pre-selected maneuvers are used as test cases. These maneuvers are initially flown by a Fuzzy Controller pilot, following set-points or “points in space” that specify the desired path. It is assumed that the Fuzzy pilot is well-tuned and is able to follow the set-points. Inputs and outputs were recorded in real-time as each maneuver was performed inside the MATLAB environment. The Fuzzy Controller pilot acts as an observed-data “source” (the data source can be a human pilot, a Fuzzy Controller pilot, a PID controller, or any technology that is able to execute these maneuvers correctly). The collected data includes inputs to the pilot as well as output from the pilot to the helicopter. The virtual helicopter was built using MATLAB version 7.1.0.246. Table 5.1 shows each of the four selected maneuvers along with their respective flight time and a small graph showing the path as executed by the Fuzzy Controller pilot.

5.2 The Flight Maneuvers

The four pre-selected flight maneuvers are non-aggressive flight maneuvers that are designed to provide a complex flight pattern. The maneuvers selected also show the flexibility of the auto-pilot in that it is able to duplicate different maneuvers.

Table 5.1: Set of Maneuvers to be Duplicated, Corresponding Flight Times and Flight Pattern Graphs.

	Flight Pattern	Flight Time (seconds)	Flight Pattern Graph
1	In flight loops	120	
2	Takeoff and U-Turn	120	
3	Ascending Spiral	30	
4	In flight loops with changing height	150	

5.2.1 Simulating the Maneuvers

As the fuzzy logic controller (source) executes each maneuver on the MATLAB helicopter, the source's actions and feedback from the virtual helicopter are recorded for the entire duration of the flight. The collected data is then converted to a tabular format that can be used by the GA/SA search algorithm. The data is also used later for testing.

5.2.2 Approach Used in Duplicating the Maneuvers

There are a number of approaches that can be used to copy flight maneuvers. Perhaps the simplest approach is to use a lookup table that takes in an input value and gives immediate feedback. The lookup table is 100% accurate with the training set and it needs no training time. However, no new knowledge is derived, and the system is not modeled. Soft-computing techniques are a good alternative that offer potential. However, not all soft techniques provide output that can be mathematically analyzed. The goal of the GA/SA search algorithm is to derive a solution in terms of an equation with mathematical components selected by the tester. Mathematical formulas are straight forward to implement and provide insight as to how flight time is used in calculating control signals. Formulas show exactly how the input (Flight time) and outputs are related and allow the researcher to model the system. The results can be analyzed and studied further to reach new conclusions about the system.

Since the proposed method uses genetic search and simulated annealing to assemble a formula out of a set of mathematical building blocks, any inherent bias may be minimized by using the two search algorithms. Both search algorithms are run in parallel or they can be run serially. Hence, they are independent of each other and that makes it more likely for one to generate an optimal solution. Naturally, the likelihood of a unique formula depends on the number of maxima points in the data. The algorithm proposed provides a versatile and powerful search tool that can be used in investigating, verifying, and analyzing problems where the relationship between the independent and dependent

variables is not well understood or known. The search itself is guided by a fitness function selected for the purpose. The function can be changed or adjusted as needed. A typical search arrangement is shown in figure 5.1.

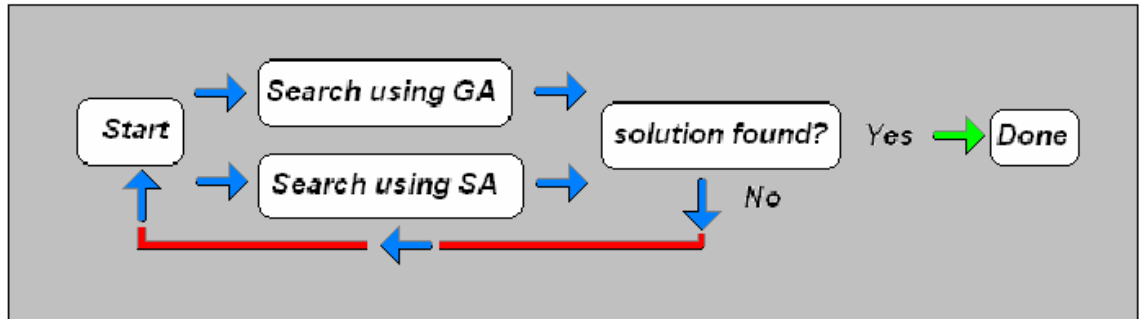


Figure 5.1: Typical Search Arrangement Using GA/SA Algorithms to Search in Parallel.

5.2.3 Why Use a GA/SA Search Algorithm to Duplicate Maneuvers

Genetic Algorithms and Simulated Annealing techniques were selected as the preferred search method after reviewing the literature for a robust and effective search technique for this type of problem. These guided hill-climbing techniques are among one of the most effective and robust search techniques [103]. However, there is debate as to which is more effective, Genetic Algorithms or Simulated Annealing. There are many studies on the subject [104]. Theoretically Genetic Algorithms are very similar to Simulated Annealing in the way they search with the main difference being multiple populations vs. a single solution [103].

Both methods assume that ‘good solutions’ are found ‘near’ current good solutions [103]. However, a GA assumes that combinations of two parents will usually lead to a ‘near’ solution [103]. Hence, for some applications, evaluating solutions near an existing solution may be more efficient, giving the advantage to Simulated Annealing. Still, not all comparisons between the two are accurate because many papers do not consider execution time [103]. There is no consensus among the scientific community other than for some problems one may be more optimal than the other [105]. In general, it is believed that a GA takes longer but finds slightly better solutions [103]. An SA is believed to converge faster.

As such, one may do better than the other depending on the search space. For this type of exploratory work it is better to have flexibility by using both. In addition, if there are multiple processors on a system, then there is an advantage to search in parallel as shown in figure 5.1.

5.2.4 Comprehensive Identification from FrEQUENCY Responses (CIFER)

It is worth noting that a system NASA uses, called Comprehensive Identification from FrEQUENCY Responses (CIFER), works by measuring frequency response. This system has some similarities to the proposed method. CIFER is summarized on NASA's website as: "System identification is a procedure by which a mathematical description of vehicle or component dynamic behavior is extracted from test data. It can be thought of as an inverse of simulation." [97]. Both CIFER and the proposed method have system modeling capability. CIFER is used to derive a frequency-based dynamic description of a vehicle. The proposed method generates mathematical equations that can control a vehicle or describe a vehicle. However, CIFER uses a set of proprietary, non-parametric, frequency responses that describe the coupled characteristics of a given system. CIFER depends on "advanced Chirp-Z transform" and "composite optimal window techniques" that are designed to have better performance than Fast Fourier Transforms [99]. The "Chirp z-transform" is based on the z-transform with one of its functions having a linearly changing frequency [98]. In essence, CIFER uses "sophisticated nonlinear search algorithms" to model a system [99]. This is in contrast to the proposed method which derives formulas based on functions of the tester's choice. Also, the proposed method uses genetic and simulated annealing search techniques instead of Fourier Transforms. A search of the literature indicated that there are no studies that investigate the ability of Genetic Algorithms and Simulated Annealing to find specific mathematical formulas from a set of data the way that they are applied in this dissertation. The work done in this dissertation would further research in the field of Artificial Intelligence by introducing a new search tool that can be applied to derive new and unique formulas that may be used to control automated systems.

5.3 Genetic Algorithms and Simulated Annealing

The science of computer-based genetic algorithms is relatively young; the earliest research started in the late 1950s. Researchers first stumbled upon genetic-like algorithms as a result of writing algorithms for search in artificial systems. A number of biologists used computers to perform simulations of genetic systems. Some of the earlier studies are listed below:

- Simulations of computer-based evolution began around 1954 with the work of Nils Aall Barricelli [68].
- In 1957, Nils Aall Barricelli authored a paper that discussed a symbio-genetic evolution process that was achieved by artificial methods [68].
- Also in 1957, Alex Fraser, an Australian quantitative geneticist, authored a set of papers that discussed the artificial selection and its simulation. The work discussed using multiple loci to control a measurable trait [68].
- In 1960, Fraser wrote on the simulation of genetic systems using computers [68].
- The field of artificial evolution achieved prominence during the 1960s and 1970s. This was largely due to the work of Ingo Rechenberg and Hans-Paul Schwefel. These scientists used evolution-based search and optimization techniques to solve difficult engineering problems [68].
- In 1962, Barricelli authored a follow-up paper titled: “Numerical testing of evolution theories.”
- In the 1970s, John Holland researched Genetic algorithms and released a book in 1975 titled “Adaptation in Natural and Artificial Systems”[69].
- Until the 1980s, GAs were mainly used in theory-based research. However, the first International Conference on Genetic Algorithms signaled a new era where GAs became widely accepted as a valid tool for problem solving. The conference was held in Pittsburgh, Pennsylvania [68].

- In the 1980s, desktop computers became more powerful. This allowed more researchers to conduct genetic-based research. This further contributed to the popularity and acceptance of genetic algorithms and genetic-based search techniques [68].
- The first commercially available genetic algorithm product was introduced in the late 1980s by General Electric. The product was a mainframe toolkit designed for use in industrial processes [68].
- Also in the late 1980s, Axcelis Incorporated introduced the world's second genetic commercial product. It was called Evolver, a GA-based problem solving program for desktop computers. It was the first GA commercial product made for desktop computers [68].
- John Koza used genetic programming to solve problems; Koza's technique is different in that computer programs, and not function parameters, are optimized [68].

Some of the early studies were mainly concerned with understanding genetics and evolution as it existed in nature. However, Fraser's work did not differ much from the modern definition of a genetic algorithm. Fraser's goal was to simulate the evolution of a small population represented genetically by a 15-bit binary string. He derived new generations and calculated the percentage of acceptable solutions (those that met or exceeded a pre-set threshold of performance). Fraser came up with a useful optimization method [44]. However, Fraser did not apply nor did he write anything that indicated the application of his algorithm to artificial systems. One of Fraser's contemporaries, John Holland, was also working on computer-based genetic algorithms. Holland is regarded as the father of genetic algorithms. The branch of computer genetic algorithms evolved from these early studies [45].

5.3.1 Genetic Algorithms and Search Space

Genetic algorithms work by searching for a desired solution among a collection of candidate solutions. The search for an optimized solution among a set of solutions is termed as searching within a "search space." There are many approaches to searching a

“search space.” Some are brute force approaches such as an exhaustive search where every possible candidate solution is tried until the best one is found. Exhaustive search works very well, for it guarantees an optimal outcome. However, there is a large time overhead associated with exhaustive search. For very large search spaces, exhaustive search is not practical. There is also random search where a candidate solution is selected at random, evaluated, and at the end of an allotted time period, the best solution found is returned. This is also not a very practical approach. More “intelligent” search algorithms try and use the experience of past searches to “guide” the search. Genetic search is a guided random search that “hones in” on a solution as time goes on.

5.3.2 Terminology

The terminology used in computer science based genetic algorithms is derived from biological terms. In biology, the values that a gene can have are called alleles [70]. For example, hair color alleles are blond, brown, or red. The location of the gene within the string of genes, the chromosome, is called the locus or position on the chromosome. The organism's genome is the term used to describe the collection of all chromosomes bundled together while the term genotype is actually the specific set of genes that are in a genome [70]. Hence, if two organisms have identical genomes, the latter implies that they have the same genotype. The physical characteristics displayed by an organism are referred to as the phenotype which is the manifestation of the genotype in an organism.

Figure 5.2 shows an example of genetic reproduction in which genes of both parents contribute to create an offspring. In figure 5.2, the split point is chosen and denoted with a line. Figure 5.3 demonstrates a mutation step.

In computer based genetic algorithms, a chromosome is usually associated with a candidate solution of the problem being considered. These solutions are usually encoded as bit strings. The “genes” are represented each as a single bit or a series of short blocks that are comprised of adjacent bits. It is these bits that encode a particular atomic element of a

possible solution. The term allele refers to the values that can be set in the chromosome(s); if the encoding is binary based, then the values for the allele are either 0 or 1. A larger alphabet may be used that has more values; hence, more alleles are possible at each locus [68]. The crossover operation usually involves the inter-exchange of genetic material between two parents. The mutation operation involves toggling a bit whose locus is randomly chosen. If the alphabet is larger than a binary alphabet, then mutation involves replacing a symbol with another randomly chosen one (from the alphabet of course) at a randomly chosen locus [68].

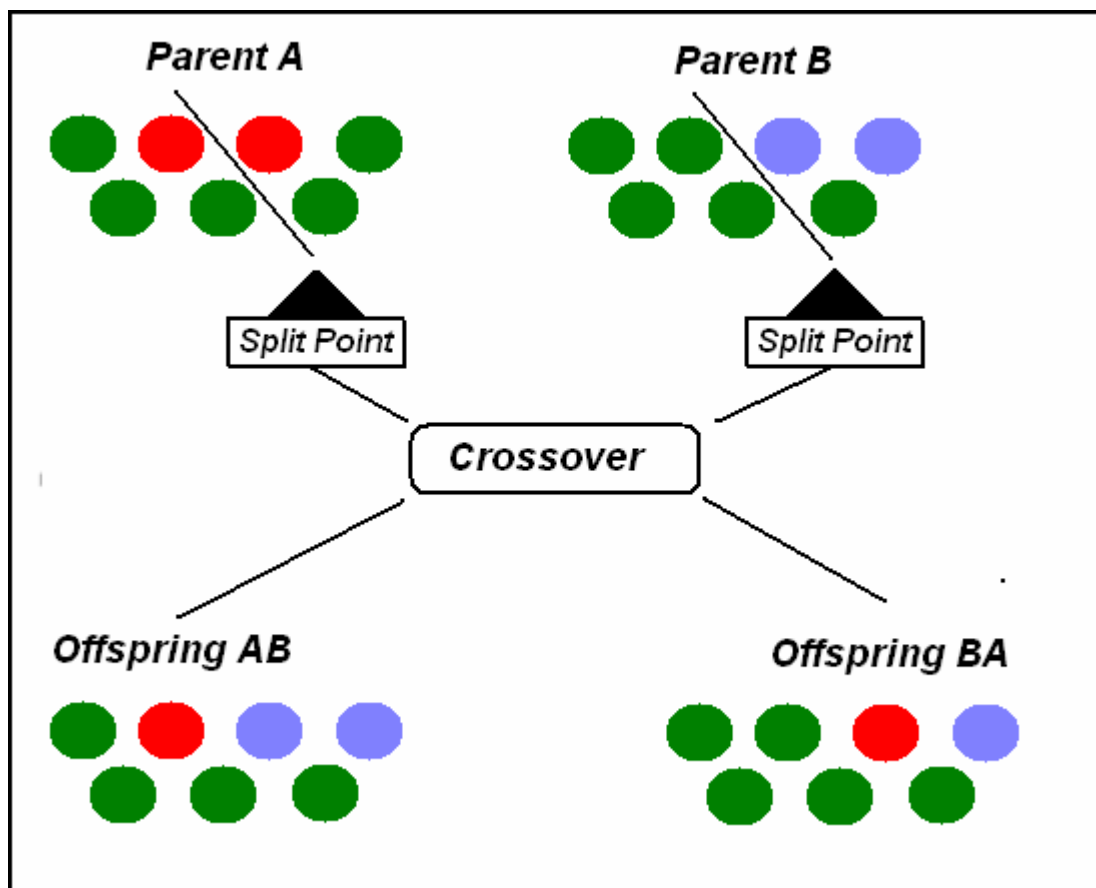


Figure 5.2: Example of Genetic Crossover.

For the most part, applications of genetic algorithms, the genotype of an individual in the population is configured using bits strings. That same individual can be encoded and configured using a more complex alphabet if needed.

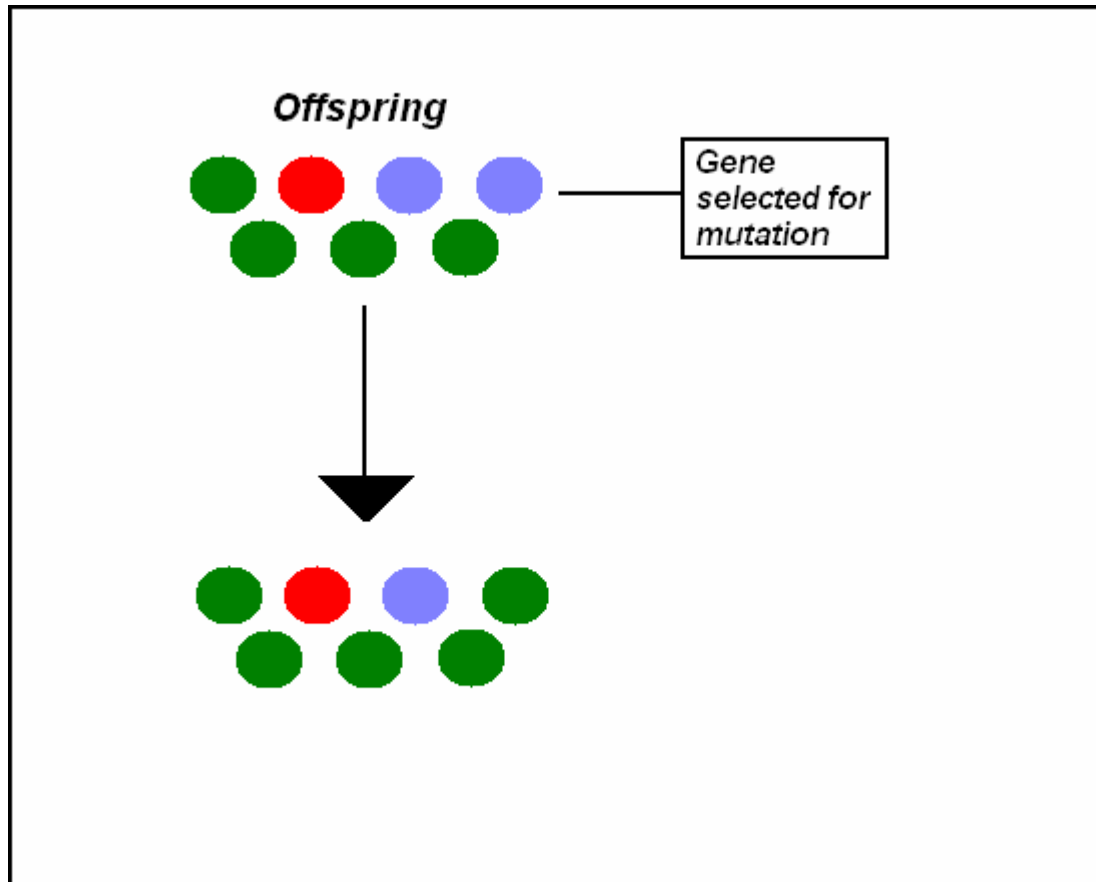


Figure 5.3: Example of Genetic Mutation.

5.3.3 The Search Space

The idea a "search space" refers to a collection of candidate solutions of some problem and the idea of a "distance" between candidate solutions. If an algorithm is used to search for a sequence of numbers that are the combination to a binary key, then the search space is the collection of all possible sequences of 0's and 1's. Hence, depending on the length of the key, this can be a very large number. However, if it is known that the key is no longer than 1000 bits long, a substantial amount of the search space can be

eliminated from the search. Then candidate solutions may have the following format: [011101011010101010...]. The latter is essentially a sequence of 0's and 1's that represent one possible value for a key. The distance between any two given solutions may be defined as the number of positions in which the numbers at corresponding positions are different. As such, the distance between 01100100000 and 01111100000 is 2. An algorithm that searches this space must decide which candidate solutions to combine to produce better "offspring" solutions from two parent solutions. Genetic algorithms assume that high-fitness "parent" candidate solutions from different areas in the search space may be combined to produce a high-fitness "offspring" candidate solution. The term "candidate solution" is used to denote a possible solution that is presented to selection.

5.3.4 The Fitness Landscape

There is also the concept of a "fitness landscape". This concept was introduced by biologist Sewall Wright (1931) while working on population genetics [73]. Therefore, a fitness landscape is essentially a representation of the space of all possible genotypes and their respective fitness values. Where a genotype is an encoded string of l bits that represent a solution, each genotype should have an associated fitness value that is calculated using some fitness function. The distance between two genotypes can be measured as the "Hamming distance" which is the number of locations where the bits differ between the two solutions [72]. Consequently, a fitness landscape can be represented as an $(l + 1)$ dimensional plot where each genotype is given a point in the l dimensions, and its corresponding fitness value is plotted along the $(l + 1)$ axis; the latter gives rise to a graphical representation of the fitness landscape. For example, to keep things straight forward, a landscape of $l = 2$ is assumed to exist with real fitness values for each genotype. The fitness landscape associated with this example may look like the one in fig.5.4 where the peaks represent high fitness or local maxima and the valleys low fitness or local minima.

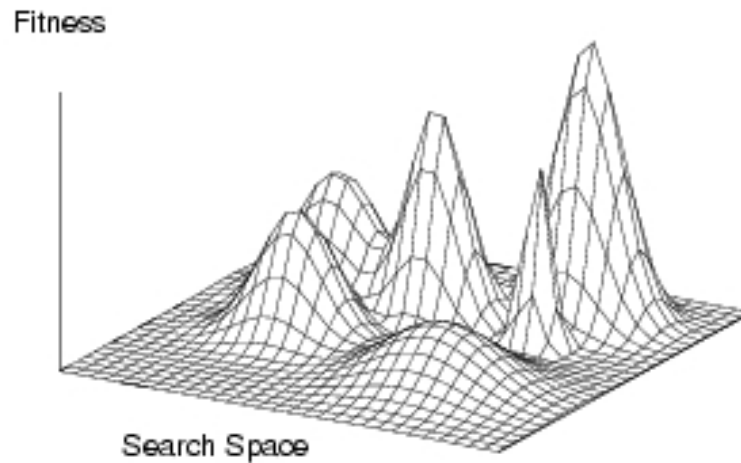


Figure 5.4: An Example of a Fitness Landscape. (source: <http://www.i3s.unice.fr/~verel/recherche.html>)

The term “landscape” is used because plotting the fitness values in this manner may form “hills” and “valleys” and other topographical attributes that are similar to actual physical landscapes. Wright proposed that evolution causes populations to migrate on these landscapes in certain patterns and that “adaptation” to an environment is evident in the movement toward local hill tops which represent local maxima [73]. The local maxima may not necessarily be the most optimal point in the landscape, but any movement away from this point (even very small movement) may result in degradation of fitness. Genetic algorithms use crossover and mutation operations as tool to move the population around the fitness function landscape as shown in fig.5.4.

5.3.5 Possible Definition of a Genetic Algorithm

Computer science literature does not present a universal definition of what a genetic algorithm should be [68]. However, different genetic search methods share a set of elements that makes them a genetic search algorithm. These attributes are:

- Chromosome populations.
- Survival of the fittest, meaning a higher fitness of a solution makes it more likely to be selected.
- A crossover algorithm to produce new offspring.
- Mutation of new offspring, which is usually done randomly.

The chromosomes in a GA population typically take the form of bit strings. Each locus in the chromosome has two possible alleles: 0 and 1. Each chromosome can be thought of as a point in the search space of candidate solutions. Survival of the fittest individual solution usually requires a fitness function that assigns a score or fitness to each member of the current population. Crossover and mutation are strategies used to generate new offspring.

Genetic algorithms can be defined as adaptive approaches that can be used to solve search-and-optimization problems in an efficient manner [48]. They operate much like natural genetic algorithms in utilizing the concepts of natural selection and survival of the fittest to produce the next generation. A solution is evolved over time, granted that the search domain is reasonably understood, and the genetic algorithm is properly coded.

5.3.6 Genetic Algorithm Cycle

Computer genetic algorithms work in the following manner: a group of solutions, also known as a population, is maintained where each solution is usually encoded as a string of zeros and ones or other symbols. The encoded parameters represent some point in the solution plane. Also, each solution is represented as a single independent unit. There are many ways to encode information that represent a solution, the most popular of which is using 0's and 1's.

A new population, the next generation, is derived from the current generation through a process similar to that found in nature; as such, genetic search uses reproduction

and mutation of genetic information resulting in a new unit that is unlike either parent. The new population of solutions is not identical to the current generation but contains pieces of it. The pieces are selected in a way that the average fitness of the new population increases with each new generation. Thus, each successive unit in the new population contains bits or some genes of their predecessors. However, each new unit is different and thus each successive population represents a class of new solutions that are more optimized. Reproduction occurs by selecting two parent units from a pool of units pre-selected for reproduction. This pool consists of units that have high performance or fitness values. For example, the GA may select the top-performing 30% of solutions for reproduction. Once the two parents are selected, the GA derives one or more new units from them using crossover reproduction techniques. The crossover process works by selecting a split point; bits to the right of this split point are swapped with the corresponding bits of the second parent. This produces a child unit that has bits of both parents [49].

The type of split just described is called single point crossover. A split point is usually selected at random and thus can be anywhere. In case the first bit to the left is selected, the resulting offspring is essentially a copy of one of its parents. Another type of crossover is called the 3-point crossover and involves selecting two split points on both parents. Next, the bits between the split points are swapped to create a new unit. This second technique offers more flexibility in the sense that it's possible to slice a string of bits and exchange it with another to create a new unit. Another type of crossover, multiple point crossover, involves selecting many split points at random. This crossover technique takes the 3-point split a step further by allowing the algorithm to create many bit strings of varying lengths.

Regardless of the crossover method utilized, the recombination process is repeated until the desired number of new units is generated so that the next generation has the correct number of individuals. Once the new population is created, a small number of genes are randomly selected and then mutated. Generally, mutation is applied very conservatively and is used as a sort of wildcard to introduce a random element in the genetic pool;

sometimes it works to the disadvantage of the search. However, research has shown that, overall, mutation works [46]. It works by jumping to a random point in the search domain; mutation sometimes leads to the discovery of maxima that may have been missed or may not be discovered for many generations. Therefore, it speeds up the search and makes it more effective.

The new population is derived and evaluated for fitness. A subset population consisting of better performing units is then selected. Next a new generation is derived from this subset [49]. The process is repeated over and over until a pre-set fitness value is achieved. However, it is not always known what the best possible solution is; this is usually due to a problem domain that is not well understood. In this case, the search is limited to a fixed number of generations or a fixed amount of computer time. In fact, genetic algorithms search a group of solutions, called a population, in parallel. Generally speaking, genetic algorithms are relatively simple to implement; usually, all that is needed is a random-number-generator function, a string copying-and-manipulating function, and a fitness-measuring function [47, 48].

5.3.7 Why Genetic Search

Regardless of differences in crossover or mutation strategies, the basic steps mentioned above inherently provide genetic algorithms with certain advantages when compared to other search techniques. For a start, genetic algorithms avoid getting stuck in local maxima by searching in parallel from a number of different individual solutions. This sharply contrasts to other search techniques, which are based on gradient methods that tend to search from a local area. Another advantage is that genetic algorithms are not hindered by discontinuities in solutions expressed by mathematical formulas since they directly manipulate a string representing those formulas [49]. Genetic algorithms offer versatility and flexibility needed for this application since genetic algorithms deal with parameters of finite length. These parameters are coded using a finite alphabet. This is advantageous, because rather than directly manipulating the environment parameters themselves, genetic algorithms manipulate solutions that are represented in a finite alphabet. However, out of

this finite alphabet an almost infinite number of solutions can be constructed [50]. Thus, the search in a problem domain is unconstrained by continuity of the function in question. Performance, fitness of a given solution, is evaluated objectively using a given fitness function. This makes the search domain transparent to the algorithm and frees it from the constraint of having to use auxiliary or derivative information [50]. The search for a solution starts from many points and then evolves, in contrast with neural networks which tend to start from just one point and refine the search using a gradient- descent method or some other error reduction technique.

It is parallelism that makes genetic algorithms potent. It often insures, given enough generations, that the search will not become trapped in local maxima. It is important to make sure that genetic algorithms do not localize their search by using mutation. Some neural algorithms try to avoid localizing the search too much by starting the search at arbitrary points. However, information from previous runs is not used to enhance the search. The very search-like nature of genetic algorithms means that transition rules used by the algorithms are probabilistic, not deterministic and, hence, the chance of finding the optimal solution increases with each subsequent generation [46]. A distinction exists between the randomized operators of GAs and other methods that are simple random walks. The difference is that GAs use random choice to guide a highly exploitative search [46].

5.3.8 Main Advantages of Genetic Search

5.3.8.1 Direct Manipulation of Encoded Variables

Genetic algorithms manipulate variables at the most primitive level, the genetic string level. Thus, genetic algorithms can optimize results by combining features of the highest performing strings. Since genetic algorithms deal with parameters of finite length (coded using a finite alphabet) rather than dealing with the direct manipulation of parameters, the search is not hindered by the continuity of the function under investigation

or the existence of a derivative [49]. Performance evaluation of a possible solution is investigated using payoff information.

5.3.8.2 Search from a Population, Not a Single Point

A typical genetic algorithm maintains a population of solutions. Each solution starts at a different point in the search domain and continues from that point. The algorithm optimizes the solution until it reaches the maximum possible performance level within the allotted time. Thus, the possibility of getting trapped at local maxima is minimized. The likelihood of getting stuck at local maxima can be reduced even further, almost eliminated, if a measure of inbreeding-avoidance is incorporated into the algorithm.

5.3.8.3 Search via Sampling, a Blind Search

Genetic algorithms have another advantage in that they ignore most of the information in a search domain except that pertaining to payoff [50]. Other search methods rely on external information to fine-tune their search; the external information is usually rules that define the behavior of the domain in question. However, not every problem domain is well defined or well understood, so methods that rely mainly on external information break down when the necessary information is not available or difficult to obtain. Genetic algorithms remain general by relying on information gained from the search space itself.

5.4 Simulated Annealing

Simulated annealing is a general-purpose optimization algorithm that traces its origins to statistical mechanics. Kirkpatrick invented this algorithm in 1982; it is based on the hill-climbing method. This algorithm is straightforward as it starts from a random point in the search environment, but it differentiates itself from other search methods in the way the search is directed [77]. The search begins by choosing a random point and evaluates it

for fitness. If the random point selected has better performance than the starting point, then it is adopted unconditionally. Otherwise, it is accepted with a probability $p(t)$ where p is the probability function of accepting a solution with respect to time (t). Initially $p(t)$'s value is close to 1, but as time passes it gets closer to 0. This allows the algorithm to accept almost all moves early in the search but to become more selective as time goes on [77]. The probability of accepting a negative move becomes less and less as time passes. The non-optimal, negative moves are accepted early in the search because they help the search algorithm avoid local maxima.

However, simulated annealing still has some of the drawbacks of random search. In addition, it lacks the ability to construct a big picture of the problem domain as time passes. This is because information gathered from previous moves is not utilized when selecting new moves. Regardless of the drawbacks, simulated annealing can be quite a powerful search method and according to some articles, this technique has been quite successful in VLSI circuit layout applications [47]. The simulated annealing algorithm is shown in figure 5.5. The term “annealing” is from metallurgy, and it refers to metals being cooled slowly or annealed. Metals treated in this manner have superior strength to metals that have been cooled rapidly or quenched. The reason behind the higher strength has to do with the internal molecular states of lower energy [77]. This is similar to the problem of minimizing a function $f(x_1, \dots, x_n)$ where it is assumed that $f \geq 0$ everywhere. In this case, f represents the energy of a statistical mechanical system that has states $s = (x_1, \dots, x_n)$. The probability that the system is in state s at temperature T is found by the Boltzmann-Gibbs distribution $P(s) = P(x_1, \dots, x_n) = e^{-f(s)/kT}/Z$ [77].

The main point is that the Boltzmann-Gibbs distribution is dominated by the states of lowest energy (at low temperature) and, hence, becomes the most probable. Moreover, assuming that there are m states that represent the minimum of the function f , then the latter can be summarized in formula 5.1.

Formula 5.1:
$$\lim_{T \rightarrow 0} P(s) = \begin{cases} 1/m & \text{if } s \text{ is a ground state} \\ 0 & \text{otherwise} \end{cases}$$

Should there be a way to simulate the system at temperatures near 0, then that would result in the ground states being reached almost immediately and, hence, the minima of f . However, it is not that easy because these systems usually fail to reach the Boltzmann-Gibbs equilibrium distribution in a reasonable time.

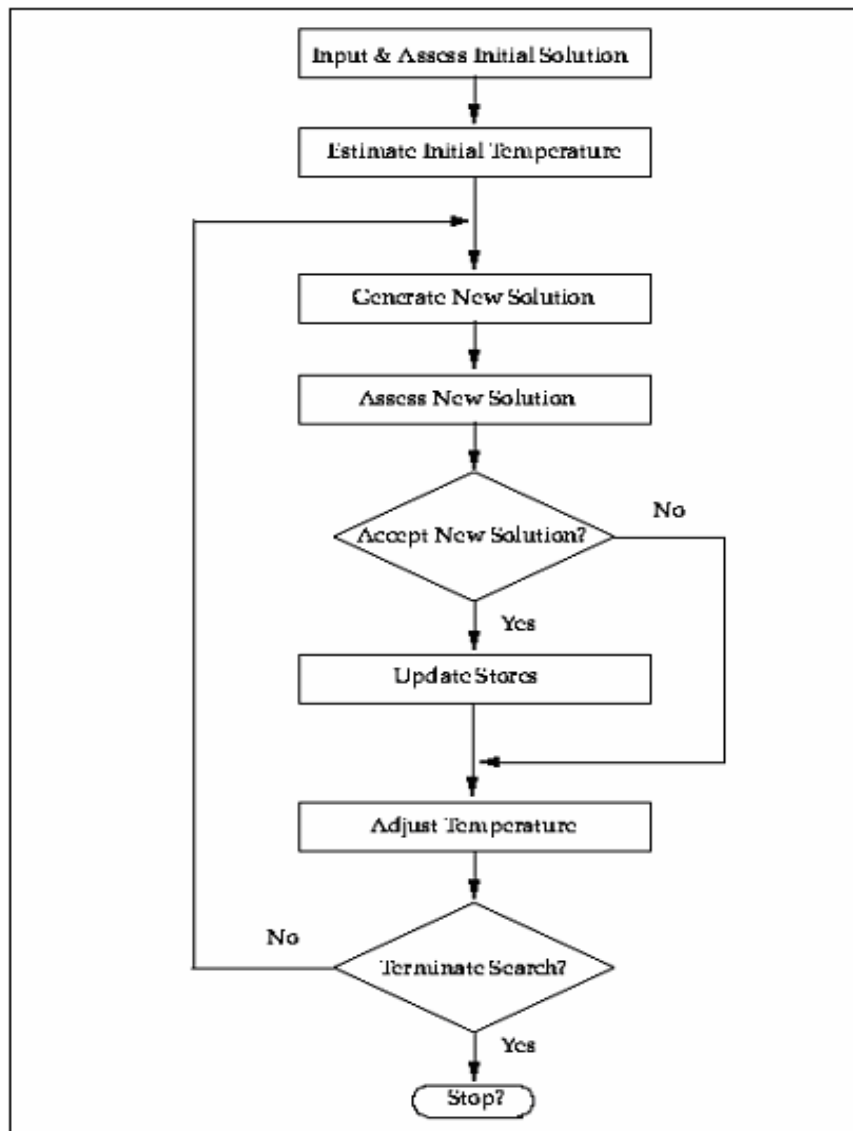


Figure 5.5: The Simulated Annealing Algorithm. (source: <http://www.phy.ornl.gov/csep/MO/NODE28A.html>)

This is due to the fact that movement in state space is restricted by very low probability regions or. Stated another way, this is due to high energy barriers. Simulated annealing deals with this limitation by starting the search with a high temperature where the Boltzmann-Gibbs distribution is essentially uniform. The temperature is slowly lowered using an annealing schedule that is unique to the application. Local and global minima and the energy states between them are shown in figure 5.6.

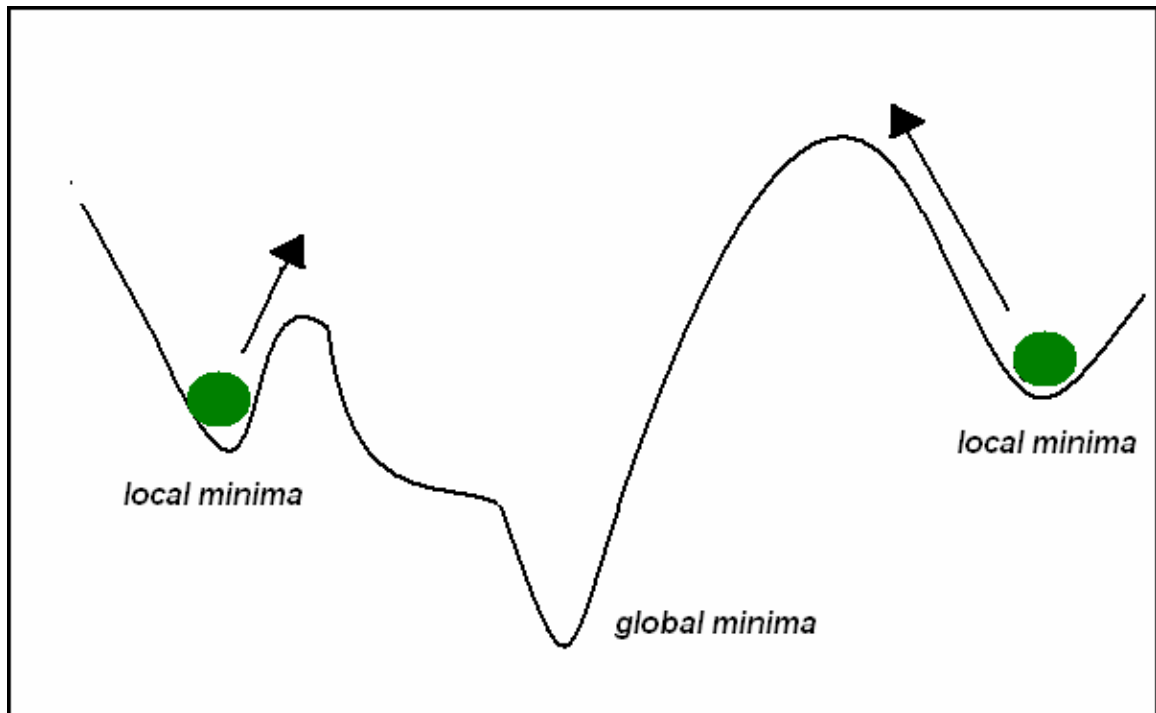


Figure 5.6: Simulated Annealing Solutions Showing Local and Global Minima.

The annealing schedule is critical because annealing too fast will not result in close to optimal solution while annealing too slow will take a very long time and, therefore, will not be practical. A typical annealing schedule based on logarithmic annealing is shown in formula 5.2

Formula 5.2: Typical logarithmic annealing schedule $T^t = K/(\log t)$ where $(t \geq 1)$

The algorithm shown above is almost guaranteed to converge to one of the ground states. The value K is constant. The temperature is represented by T the temperature and time

by lower case t . If S is a state that represents a level of energy within the system and hence s_{\max} and s_{\min} define two states with maximal and minimal energy, then it can be inferred from the Boltzmann-Gibbs distribution that:

$$\text{Formula 5.3: } \frac{P^t(s_{\max})}{P^t(s_{\min})} = (1/t)^{(\Delta E/kK)}$$

Then the value $\Delta E = E(s_{\max}) - E(s_{\min})$, If $K = \Delta E/k$ then that implies that

$$\text{Formula 5.4: } P^t(s_{\max}) = P^t(s_{\min})/t.$$

Hence, it follows that for any state s ,

$$\text{Formula 5.5: } P^t(s) \geq P^t(s_{\max}) = (1/t) P^t(s_{\min}) \geq (1/t) P^t(s_{\min})$$

Specifically, the total number of times that any state s is visited during the annealing process should have a lower bound of $P^1(s_{\min}) \sum_t 1/t$ (this is divergent). It follows then that with K scaled with respect to the highest energy barrier, it is not possible for the search algorithm [77] to get stuck in a local minima. Still, a logarithmic annealing schedule tends to be rather slow and, hence, not useful because it tends to take a very long time to converge. A logarithmic schedule also indicates that the algorithm is going to visit a significant number of all possible states making the search very similar to an exhaustive search. Hence, a logarithmic schedule has a good chance of finding the global optimum [77]. Most practical applications of these algorithms involve a set of problem classes that are typically NP complete with an exponential number of possible states. The latter makes any exhaustive search or similar algorithm impractical. In these applications, simulated annealing should be used with schedules that anneal faster than a logarithmic search such as a geometric annealing schedule that is shown in formula 5.6.

$$\text{Formula 5.6: Typical annealing schedule} \quad T^t = \mu T$$

The equation in formula 3.6 is not an optimal annealing schedule, so there is no guarantee that the global minima will be found; rather, an approximate optimal solution that corresponds to points of low energy is usually returned. It may be necessary to run the simulation a number of times to obtain semi-optimal results. It is also useful to pair the SA search algorithm with another search methodology such as a genetic algorithm to increase the likelihood of finding a solution.

5.5 Summary Comparison of Genetic Algorithm and Simulated Annealing

Table 5.2 below briefly shows how genetic algorithms compare with simulated annealing including their bases, population control, and search guidance.

Table 5.2: Comparing Genetic Algorithms with Simulated Annealing.

Genetic Algorithms	Simulated Annealing
Based on biological principles of evolution	General-purpose optimization algorithm inspired by statistical mechanics
Starts searching with a set of solutions called a population	Starts searching with a single solution
A solution population is optimized by selecting the fittest individuals after each iteration	The single solution is optimized by creating a set of solutions that are mutations of the original solution and then selecting the best performing solution
Uses a fitness function to guide search	Uses a fitness function to guide search
Stops when the maximum number of generations has been reached	Stops when the annealing temperature reaches zero
Random, guided search – avoids local maxima by maintaining a large population	Random, guided search – avoids local maxima by making probabilistic jumps within the search space

5.6 Control Function Generation

The algorithm generates a set of functions that are able to duplicate the Fuzzy Controller's helicopter control signals when given the flight time as input. Hence, the output of the search algorithm is a set of mathematical equations composed of mathematical building blocks. The GA/SA search algorithm takes the data set for each maneuver and derives a corresponding VTOL control equation by combining mathematical primitives into an equation. The fitness function guides the search since the GA/SA algorithm favors solutions with better fitness values. Each equation is capable of duplicating one helicopter control signal over the entire flight time for a given maneuver. The functions are optimized by the GA/SA search algorithm so that they duplicate, as closely as possible, the Fuzzy Controller's output values. We propose that carefully selected mathematical building-block(s) combined by a genetic and simulated annealing search based algorithm like the one presented in this dissertation can effectively mimic commands issued by the Fuzzy Controller and without needing positioning information or set points. The level of desired accuracy of the generated function can be selected by the tester. Hence, the search algorithm is able to generate functions that achieve more accurate signals. However, the tradeoff is function complexity. The mathematical building block is selected after analyzing the input table. The output of the GA/SA algorithm will be four control functions per maneuver (one function per control signal). Figures 5.7 and 5.8 summarize the control signals sent to the helicopter and the sensor readings received from the helicopter as the fuzzy logic controller executes each flight maneuver using set point data. Note that the sensor readings are used in the testing phase.

5.6.1 Control Function Accuracy

The algorithm searches for a formula from a set of mathematical primitives [19, 52, 53, 54, 55]. The generated function $f'(x)$ is derived by the GA/SA algorithm. The simulation is re-run if needed. The goal is an overall accuracy of around E-12 or better. This higher accuracy is needed due to the sensitive nature of the MATLAB model. Open-

loop flight means that even though the control signals are duplicated, the flight conditions are not exactly identical. Since the Fuzzy Controller flies in closed loop, this is not an issue. However, for an open loop controller, errors can build up; this is part of the reason that accuracy had to be high.

5.6.2 Control Functions and Corresponding Functions

The input parameters to the algorithm will be the set of four control signals:

- (1) The collective signal which controls the angle of the rotor blades; this allows the pilot to control the amount of lift.
- (2) The lateral signal (cycle) which controls the pitch angle of the helicopter.
- (3) The longitudinal signal (cycle) which controls the roll angle of the helicopter.
- (4) The pedal signal which determines the amount of yaw angle.

Note that an actual helicopter is more complex than the MATLAB model, as the throttle control was not used in the model; however, for the purposes of this dissertation, the four control signals make up the basis of almost any helicopter design and are the main signals considered in this study. For each observed control signal $f(x)$, the GA/SA algorithm generates a corresponding control function $f'(x)$ that can be compared to the original function during the GA/SA search; each of the 4 test maneuvers contains four control functions $f1(x)$, $f2(x)$, $f3(x)$ and $f4(x)$ where:

$f1(x) = \text{collective control signal (collective)}$

$f2(x) = \text{lateral control signal (lateral)}$

$f3(x) = \text{longitudinal control signal (longitudinal)}$

$f4(x) = \text{pedal control signal (pedal)}$

The GA/SA search algorithm generates the following corresponding functions:

$f1'(x) = GA/SA$ derived collective control signal (collective)

$f2'(x) = GA/SA$ derived lateral control signal (lateral)

$f3'(x) = GA/SA$ derived longitudinal control signal (longitudinal)

$f4'(x) = GA/SA$ derived pedal control signal (pedal)

5.6.3 Function Testing Parameters

The generated control function testing parameters consist of the position information of the helicopter. This data is used to create a “base line” that shows the path of the fuzzy logic controller in the x, y, and z axis. Subsequent simulations using the generated control equations also generate path data, and these are used to show the path taken by the GA/SA control equations. In the testing phase, the GA/SA generated formulas are given control of the helicopter as illustrated in figure 5.9.

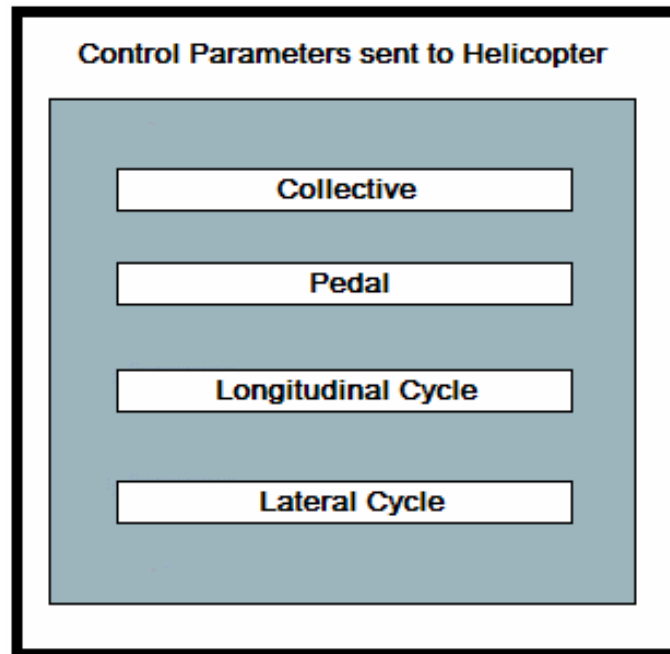


Figure 5.7: Helicopter Control Parameters.

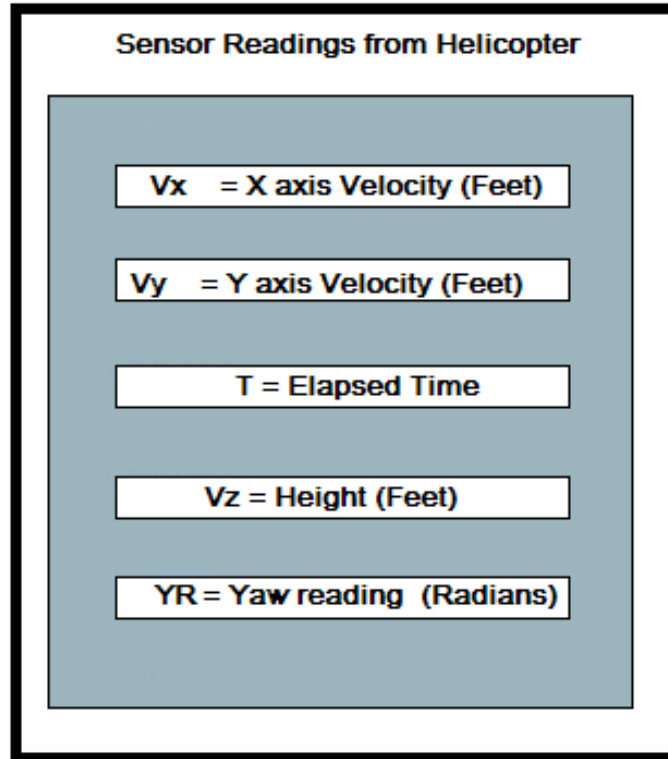


Figure 5.8 Helicopter Sensor Readings.

5.7 Data Collection

A sample table of paired input/output observed data for each control signal to the helicopter is shown in table 5.3. Each row represents 0.5 of a second elapsed time; this is one possible sampling interval. The sampling interval is a parameter chosen to sample the data from the controller flying the helicopter and may vary from 1 to 1000 samples per second. The fuzzy logic controller running in MATLAB adjusts its output 1000 times per second. However, a sampling time of 1 to 2 samples per second was found to be ample.

Each observed control signal is captured in an Excel file format. The file has 9 columns: five are sensor readings, and four are control signals. There are about 12,000 rows in each file. Each file represents a complete maneuver, and there are a total of four Excel files that represent captured data from a run. Table 5.3 shows a sample Excel file.

Table 5.3: A Sample Input Table that is used by the Search Algorithm to Derive Control Equations. The Shaded Area Represents Input Values to the Pilot.

t	xe	ye	ze	yaw	Lat	lon	ped	col
0	0	0	0	0	0	0	0	0
0.5	0	0	1	0	1.78E-39	-3.64E-39	4.03E-24	2.38E-22
1.0	0	0	2	0	2.23E-33	-3.15E-33	6.99E-18	1.85E-16
1.5	0	0	3	0	4.39E-30	-7.76E-30	1.93E-14	3.62E-13
2.0	0	0	4	0	8.94E-28	-1.65E-27	4.11E-12	6.29E-11
2.5	0	0	5	0	5.18E-26	-9.01E-26	2.19E-10	2.92E-09
3.0	0	0	6	0	1.29E-24	-2.05E-24	4.94E-09	5.93E-08
3.5	0	0	7	0	1.74E-23	-2.54E-23	6.16E-08	6.77E-07
4.0	0	0	8	0	1.48E-22	-2.01E-22	4.97E-07	5.08E-06
4.5	0	0	9	0	8.76E-22	-1.12E-21	2.88E-06	2.75E-05
5.0	0	0	10	0	3.87E-21	-4.76E-21	1.28E-05	0.00012
5.5	0	0	10	4.5	1.34E-20	-1.60E-20	4.57E-05	0.00039
6	0	0	10	9	3.75E-20	-4.40E-20	0.00014	0.00109

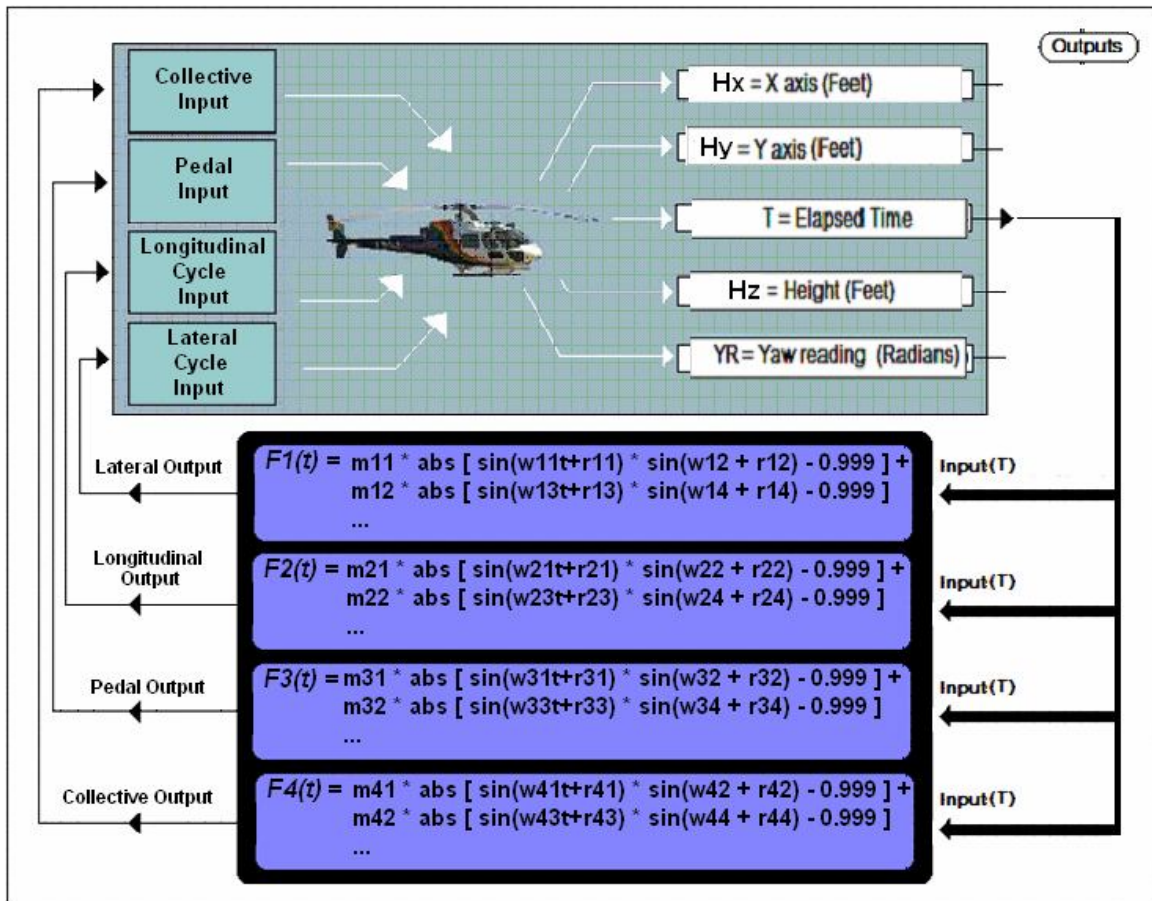


Figure 5.9: Diagram Showing Control Equations Flying the Helicopter.

5.8 Mathematical Definition of a Function

Mathematically, a function is described as a relation where each element of a set is associated with a unique element of another set [101]. In this dissertation, the terms function, mapping, and transformation will be used interchangeably. The set of input values, x_1, x_2, \dots, x_n , shall represent the domain of f , and the set of the actual corresponding output value, y , shall represent the range of f . Hence, $\{f(x_1, x_2, \dots, x_n) = y : \text{where the variables } x_1, x_2, \dots, x_n \text{ are in the domain of } f \text{ and } y \text{ is in the range of } f\}$. A function is named after its range; real functions and complex functions are named as such due to the numerical properties of their range. In this application, the domain is represented by a single input real value that changes with time. The range is the single output real value. Hence, x_1 represents the input to the control equation of the helicopter, and y is the resulting output control signal in response to this input.

More formally, the data consists of a paired input/output set. Let all the input values be represented as vector \mathbf{S} , while their corresponding output values are represented as vector \mathbf{T} . Hence, each input vector \mathbf{S} must have a corresponding output vector \mathbf{T} . The vectors represent the observed values collected from the system starting with the time a given maneuver is initialized and ending when the maneuver has been completed. The result is that each maneuver has a paired \mathbf{S} and \mathbf{T} vectors for each control signal.

The algorithm then generates a function \mathbf{F} that takes as input the set of \mathbf{S} vectors ($\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \dots, \mathbf{S}_n$) in a given 'observed' set and produces a set of \mathbf{T}_c output vectors (where \mathbf{T}_c represents the calculated version of vector \mathbf{T}). Hence, \mathbf{T}_c is an estimate of \mathbf{T} calculated by the derived function \mathbf{F} such that the output of the function \mathbf{F} is $(\mathbf{T}_{c1}, \mathbf{T}_{c2}, \mathbf{T}_{c3}, \dots, \mathbf{T}_{cn})$ where: $\mathbf{T}_{cij} = \mathbf{F}_j(\mathbf{S}_i)$ where: $1 \leq j \leq n$. and n is the size of \mathbf{T} .

The algorithm compares the set of calculated \mathbf{T} vectors (\mathbf{T}_c) to the set of \mathbf{T} vectors that were observed when the data was collected. The average error rate e is calculated for each row in \mathbf{T} as shown in formula 5.7; the error rate will be later modified to become the

fitness function. This process is repeated for every function **F** generated. The algorithm would then select the **F** function(s) with the lowest error level and continue evolving a solution until **F** is within the pre-determined error level or the maximum number of generations/annealing time has been reached.

Formula 5.7: Error rate calculation $e = (abs [T(i) - Tc(i)] // T(i)) * 100$

5.9 Selecting Mathematical Primitives and Assembling Formulas

The mathematical primitives selected must be chosen carefully so that the derived solution is as optimal as possible. The “building block” function chosen for this application was constructed based on analysis of the collected data and the corresponding data graphs. The building block includes variables that allow for the configuration of the mathematical primitives to change their response as needed. For example, the sine function was selected as one of the mathematical primitives to be used in constructing a function building block. This function was modified to allow the search algorithm to control the frequency, the altitude, and the starting and ending points of the basic sin function. The start and end points are essentially range parameters that switch the function on and off such that the function is active within a selected input value range. Note that outside that range its value becomes 0. The control parameters are variables that are manipulated by the search algorithm.

5.9.1 Assembling Formulas

Assembling the mathematical primitives into functions can be done in many ways. A format for assembling the final function was chosen because it offered flexibility. A template was developed that allows the algorithm to combine each building block with other building blocks according to a pre-determined pattern. Note: this method of learning may be classified as a supervised regression learning approach that uses batch data [100].

5.9.2 Selecting Operators and Encoding Genes

The application of these operators to the vector \mathbf{S} of input variables transforms these variables into a resulting calculated value in the range of $f^*(x)$. This resulting function, also known as the generated function, is compared with the original output values in the range of $f(x)$ stored in the corresponding result set \mathbf{T} . Hence, for some pair $\mathbf{Si}():\mathbf{Tij}$, the error value is calculated by comparing each calculated value with its corresponding actual value using the error function. The resulting error value is a percentage that shows the output deviation of the generated function in relationship to what the expected output value should be. The returned error value is referred to as the fitness of the function. The lower the error value, the better the overall fit of the generated function. The effectiveness of a solution is dependent on the power of the mathematical “building blocks” and the encoding of the search variables into a genome that can be manipulated by the search algorithm.

5.9.3 Performance Criterion

The fitness function or performance measure will be based on the goodness of the fit of the solutions produced by the algorithm. The method selected for this task is a variation of Mean Average Population Error (MAPE). The fitness function that was used for each control signal is shown in formula 5.8. The values produced by the derived function $f^*(x)$ are then compared with the values produced by the fuzzy logic controller represented by $f(x)$ and a fitness value is generated that shows the average fitness over the entire range.

$$\text{Formula 5.8: fitness function} = \left[\sum_{n=0}^s (|f(x_n) - f^*(x_n)| / f(x_n)) * 100 \right] / s$$

where;

$s \rightarrow$ number of input rows in the data,

$f(x) \rightarrow$ expected value of the control signal with input x

$f^*(x) \rightarrow$ actual control signal produced by the generated function with input x

The above fitness function shows the error rate over the whole data range. When the algorithm is searching for a solution, it is minimizing the value of f since lower values represent better function performance. The fitness function calculates the absolute difference between the value calculated by the GA/SA generated function (y') and the expected value y , which is also known as the observed value. The difference is then divided by the expected value y to determine error. The result is multiplied by 100 to get the percentage error.

5.10 The Solution Form

The algorithm generates a control signal function $F(x)$ which is the combination of many sub-functions; these sub- functions (building blocks) are connected by a combination operator c . The set of possible values for the combination operator are:

$$\text{Combination operator} = c = \{+, -\}$$

The general form of the generated solution $F(x)$ becomes:

$$F(x) = (\text{Building Block 1}) (\text{combination}) (\text{Building Block 2}) (\text{combination}) \dots (\text{Building Block } n)$$

Where each ‘Building Block’ parameter is a stand-alone mathematical function with a set of values that can be adjusted as needed by the GA/SA search algorithm. The goal of **combination operator** is to add or subtract one mathematical “Building Block” function to the next. The resulting function, $F(x)$, is then evaluated; its output is compared with the observed outputs to determine the mapping error. Each building block also has two control variables that outline a sub-range \mathbf{R} on the input set; the two variables allow the building block to return an equation when the input variables fall within the range \mathbf{R} ; otherwise the Building Block returns a 0. This allows each Building Block to be active within a specific range \mathbf{R} of inputs and then return 0 or some other desirable value outside

that range. These variables allow the building block to have an internal switch that can be turned on or off within a specific range. The two variables are “start point” and “range” respectively.

More specifically, the way the range control is implemented within each building block is as follows:

```
if (start_value < input_value < (start_value + range))  
    return fl(input_value);  
else  
    return 0;
```

The function *fl* can be any function and can be modified as needed. However, in this implementation, *fl* is a formula that is designed to map an arbitrary number of points. The function can be adjusted as needed by the GA/SA search algorithm based on its control variables. The variable “*input_value*” represents flight-time. The *start value* and *range* operators are used to limit the range of *fl* and are also controlled by the GA/SA search algorithm. The function *fl* is composed of the sum of an oscillating sine function $y(t)$ that can be used to map one or more points in space and a line function $(mx + b)$, which can map two or more points. The combination of these two functions can be used to map many points. Figure 5.10 shows an instance of the two functions $y(t)$ and $(mx + b)$. Figure 5.11 shows what happens when the *start_value* and *range* parameters are applied. Hence, $fl = y(t) + (mx + b)$.

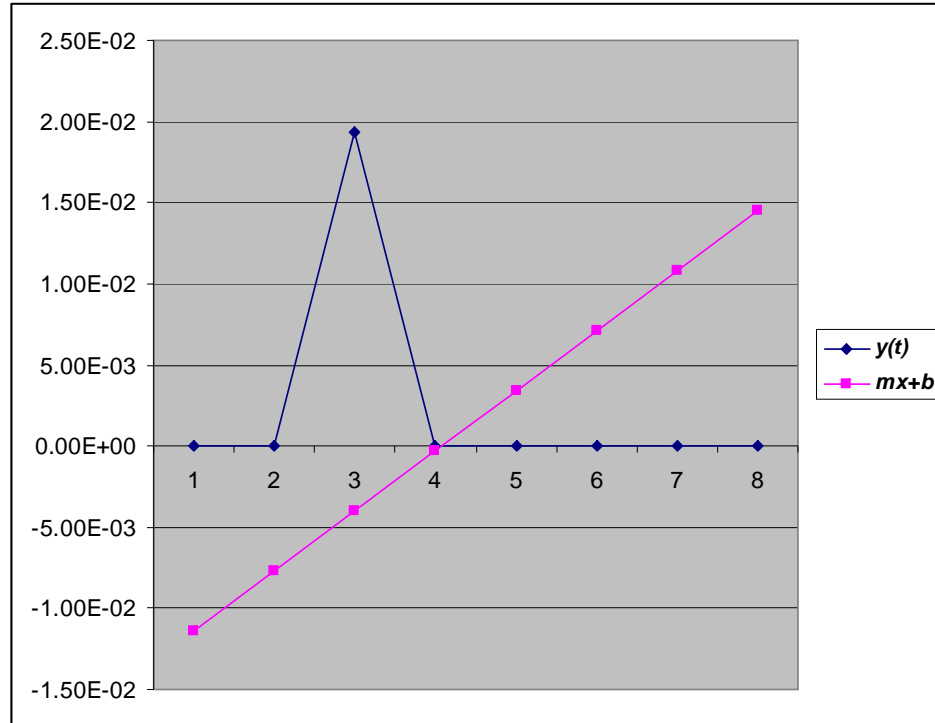


Figure 5.10: Two Functions Make Up $f1$, $y(t)$ is a Function that Peaks at One Point in the Range of $f1$ and the Line Function ($mx + b$)

Consider the basic oscillating function $s(t) = \sin (wt+r)$, where t is the *input_value* from the observed data set; this value can be time or another input signal that changes with time. The variable w (**omega**) is the frequency parameter or angular velocity, which is also a variable controlled by the search algorithm. The larger omega becomes, the higher the frequency of $s(t)$. The variable r is the offset or phase shift variable which indicates the start value of $s(t)$ at time $t = 0$ (another variable controlled by the search algorithm). The variable r allows the GA/SA search algorithm to ‘fine tune’ the response in relation to the input.

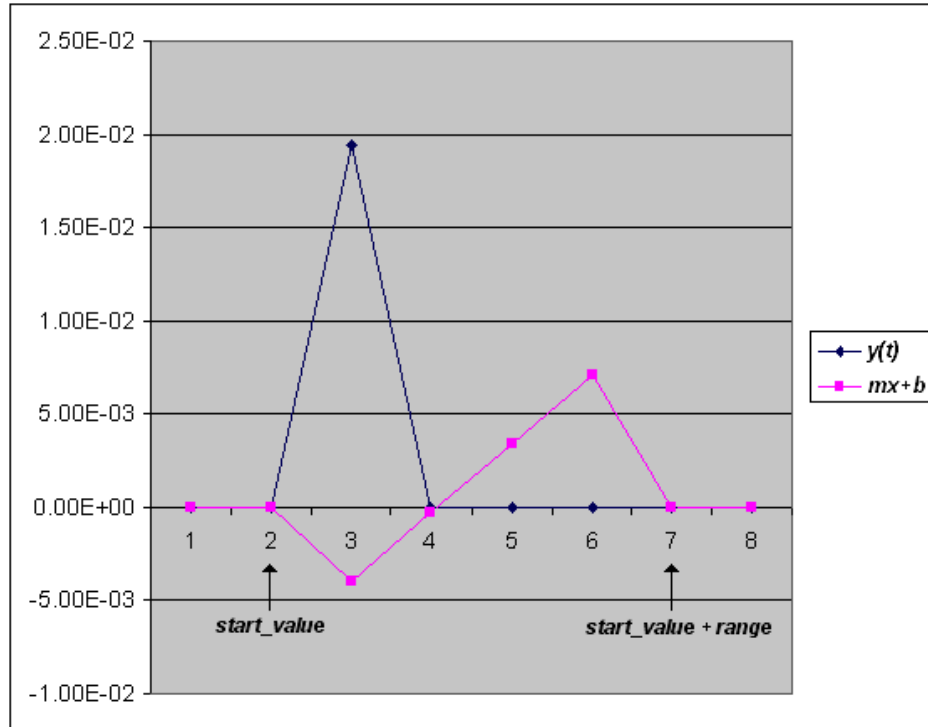


Figure 5.11: The Same Two Functions Within $f1$ but with Range Control Implemented that Makes their Value 0 to the Left of $start_value$ and 0 to the Right of $(start_value + range)$.

Let $u(t)$ be a sine function that can be positive in one or more locations (depending on w) on its range. Hence, $u(t)$ can be positive at some point $p1$ of the input set. Using a modified version of $s(t)$ and then subtracting a value close to 1 , causes $u(t)$ to be negative for most of its range. The function $u(t)$ is shown in formula 5.9, while figure 5.12 shows a graph of $u(t)$.

Formula 5.9: $u(t) = \sin(input_value * w + r) - 0.999$

The next two functions are derived using a combination of $u(t)$, $s(t)$, and the absolute value function (abs). These two functions are $v(t)$ and $w(t)$.

Let $v(t)$ be a function that is positive or zero for all values on its range with point $p1$ being the a small positive value on its range as shown by formula 5.10.

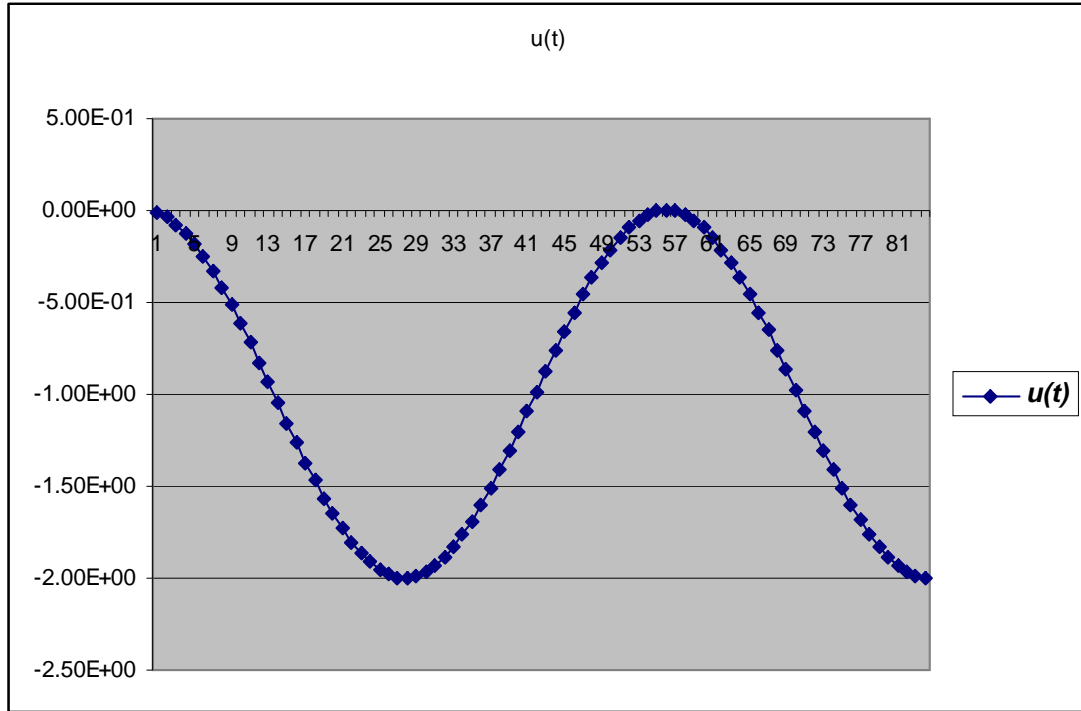


Figure 5.12: Diagram Showing $u(t)$.

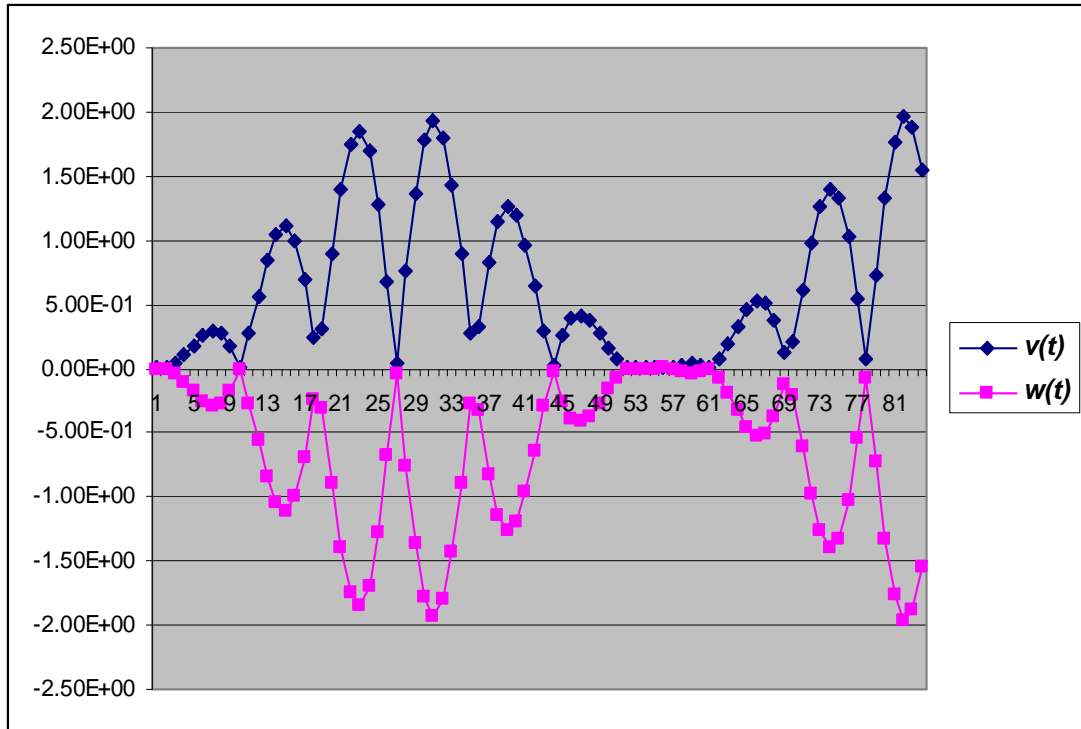


Figure 5.13: Diagram Showing $v(t)$ and $w(t)$.

Let $w(t)$ be a function that is negative for all values on its range except for point $p1$ where it is positive as shown by formula 5.11. A graph showing $v(t)$ and $w(t)$ is shown in figure 5.13, while figure 5.14 shows $u(t)$, $v(t)$, and $w(t)$.

Formula 5.10: $v(t) = abs [s(t) * u(t)]$

Formula 5.11: $w(t) = abs [s(t)] * u(t)$

Adding $v(t)$ to $w(t)$ to create function $y(t)$ will produce 0 for the entire range of $y(t)$ except at the point $p1$ where the value will be positive. The function $u(t)$ controls how many positive points, similar to $p1$, occur on the range of $y(t)$. The variable w controls the frequency of $y(t)$, while the variable r specifies the location of $p1$ on the range of $y(t)$. The function $y(t)$ is shown in formula 5.12. Figure 5.15 shows the function $y(t)$ along with point $p1$ peaking in the middle of the graph while the rest of $y(t)$ shows no response.

Formula 5.12: $y(t) = v(t) + w(t);$

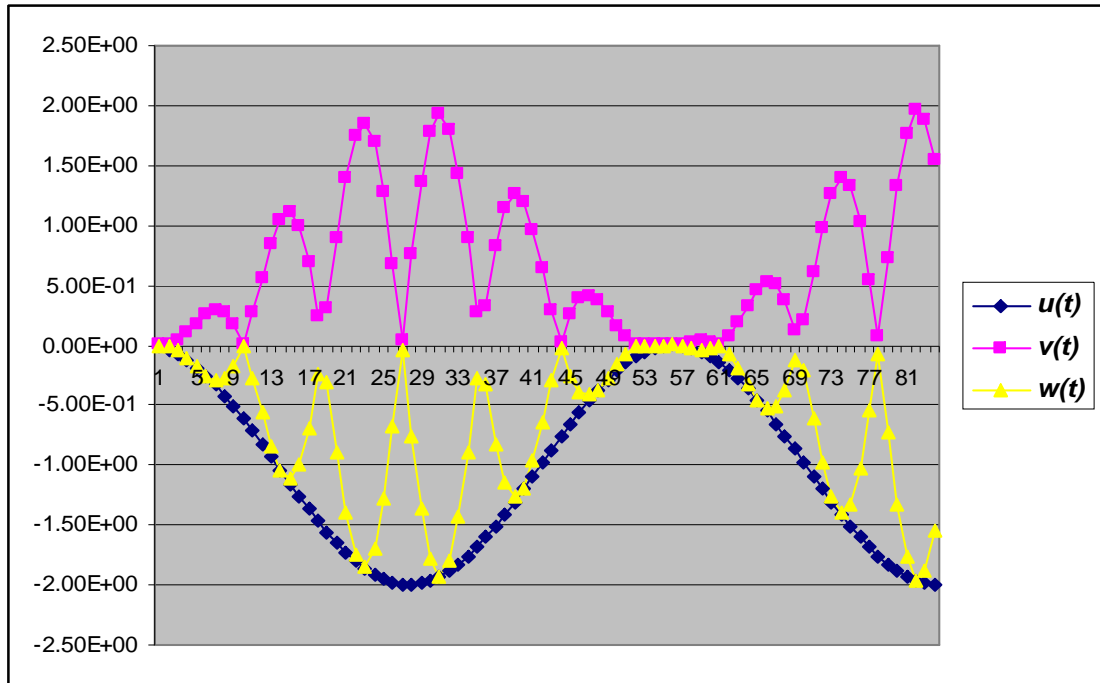


Figure 5.14: Diagram Showing $u(t)$, $v(t)$ and $w(t)$.

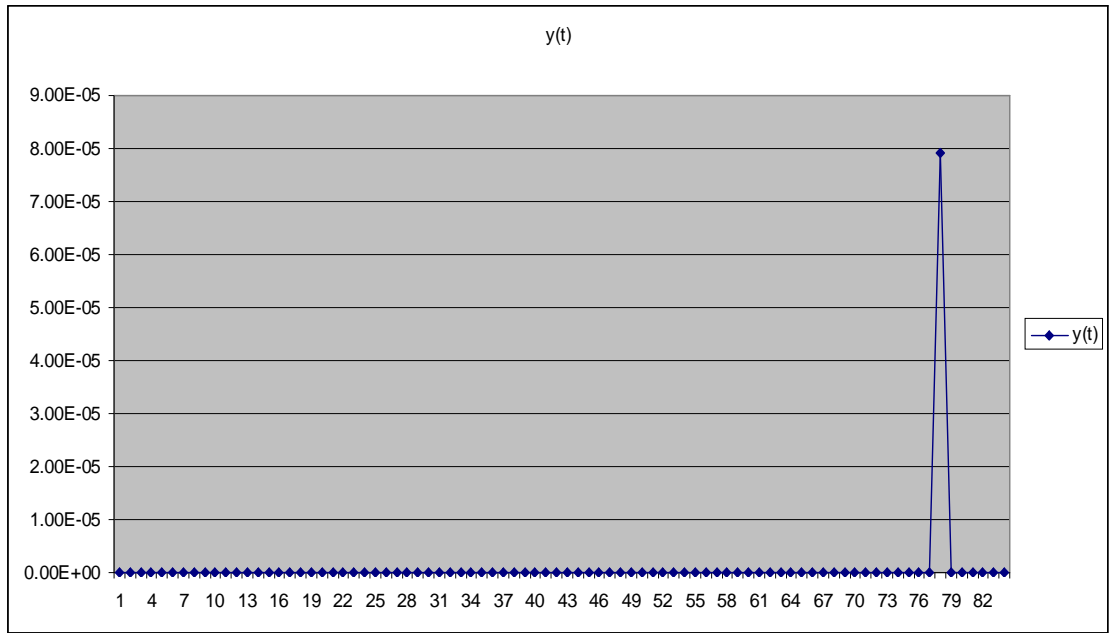


Figure 5.15: Diagram Showing the Resulting Function $y(t)$.

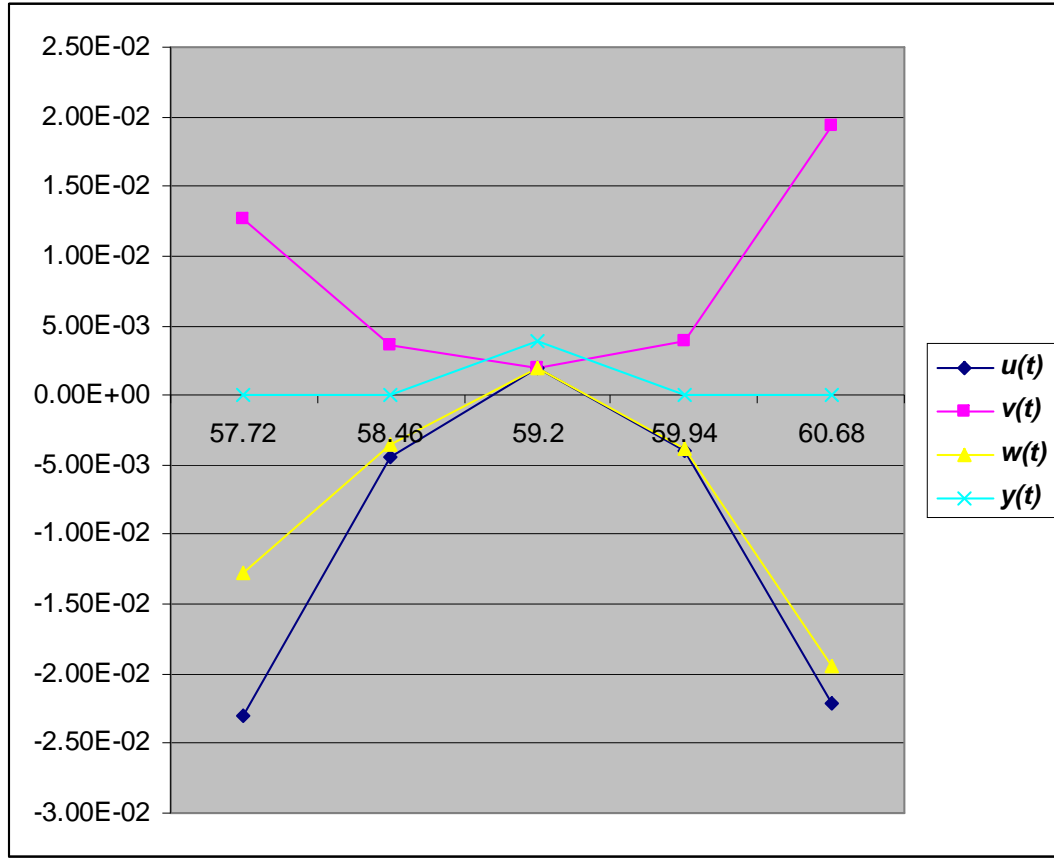


Figure 5.16: Diagram Showing how $y(t)$ is Generated at Point $p1$.

Figure 5.16 shows how the functions $u(t)$, $v(t)$, $w(t)$ interact to produce $y(t)$. The resulting function $y(t)$ produces a small positive value. In order to control the magnitude of $y(t)$, it is multiplied by the variable x . The function $y(t)$ is then combined with the equation of the line to produce the final form of the function that will be manipulated by the GA/SA search algorithm; the new function $g(t)$ is shown in formula 5.13 below.

Formula 5.13: The final solution form is $g(t) = x * y(t) + mt + b$;

Where x controls the magnitude of $y(t)$ and $mt + b$ is the equation of the line.

The algorithm keeps track of all values associated with function $g(x)$ in a values array that holds six values, so the above solution becomes:

$$g(x) = \text{Operation Result} = \text{values}[3] * y(t) + \text{values}[4] * y1(t) + \text{values}[5] * \text{input}[0] + \text{values}[6];$$

There is one $g(x)$ function and one value data array per Building Block; the value data array can hold six real values. The first two values ($\text{values}[1]$ and $\text{values}[2]$) correspond to *start value* and *range* value respectively, and these are range control values. The first value is the start of the range, and the second is the length of the range interval. The rest are used as values to control the behavior of the function $g(x)$ within the Building Block. The values array has the format:

$$\text{values}[] = \{\text{value1}, \text{value2}, \text{value3}, \dots, \text{value6}\}$$

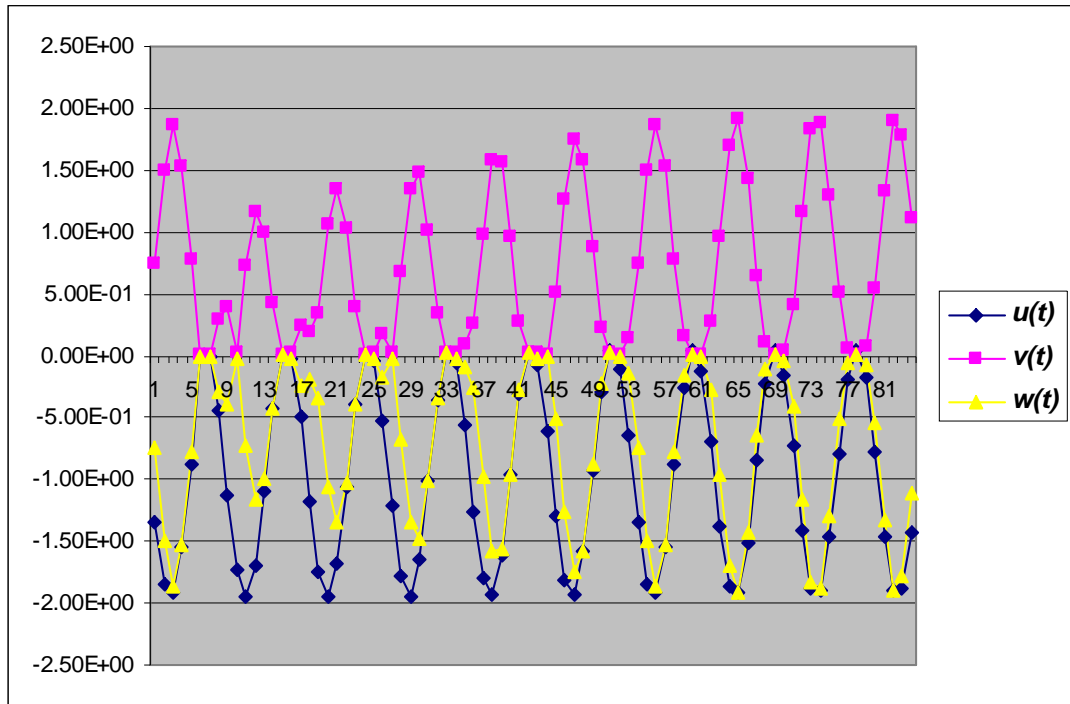


Figure 5.17: Diagram Showing $u(t)$, $v(t)$ and $w(t)$ Generated at Multiple Points.

Figure 5.17 shows the resulting positive points when ω is increased in $u(t)$ which affects $v(t)$ and $w(t)$. The latter results in $y(t)$ peaking at more than one point across

its range as shown in figure 5.18. These peaks allow for the mapping of multiple points at the same time. However, accuracy may be decreased as one or more points will not be an exact fit. On the other hand, fewer functions are needed create a mapping. Hence, generating a resulting function is a tradeoff between accuracy and number of building block functions used.

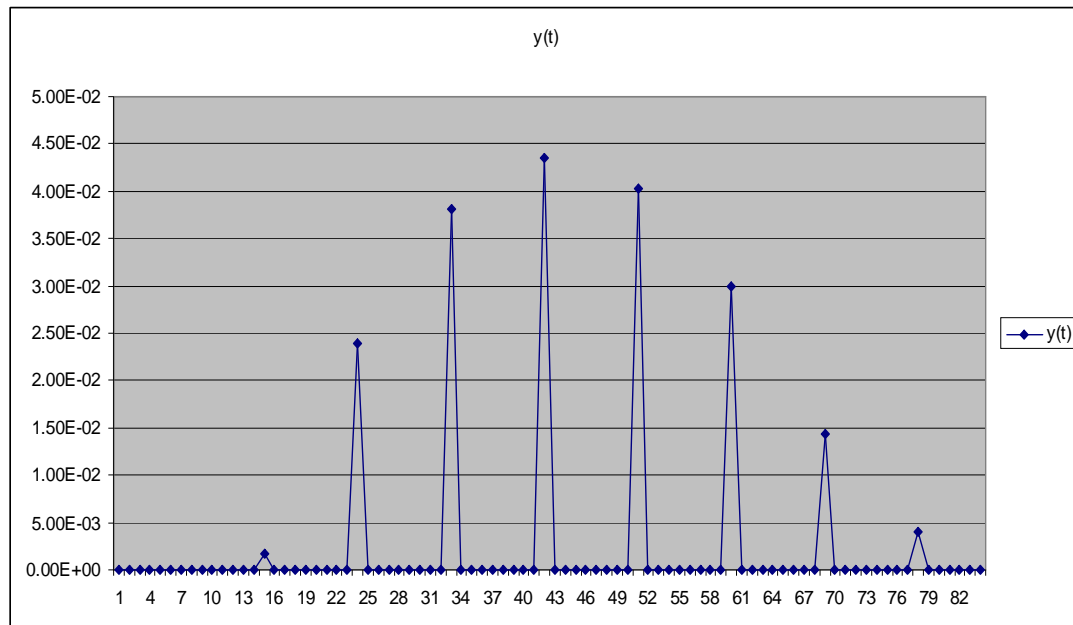


Figure 5.18: Diagram Showing Multiple Points on $y(t)$.

The search algorithm starts with one building block and appends to it as the search runs. In the case of more than one building block type, the search algorithm can select different Building Blocks that contain different f functions. However, in this application, there is only one type of building block. The search algorithm also manipulates the values array in search of the configuration that has the least error. The search within all the possible combinations of the above operators to find the best combination, that when applied to the input values in \mathbf{S} , yields a calculated value that has as little error as possible when compared to the actual desired value. If the algorithm were to search every combination, then the problem would be unmanageable for large input sets.

5.10.1 Assembling Building Blocks That Do Not Need Range Control

A building block may also be built using no range control by selecting the parameters so that the function itself is 0 across its range except one or more locations. These (positive) locations can be ‘selected’ by controlling the frequency parameter w .

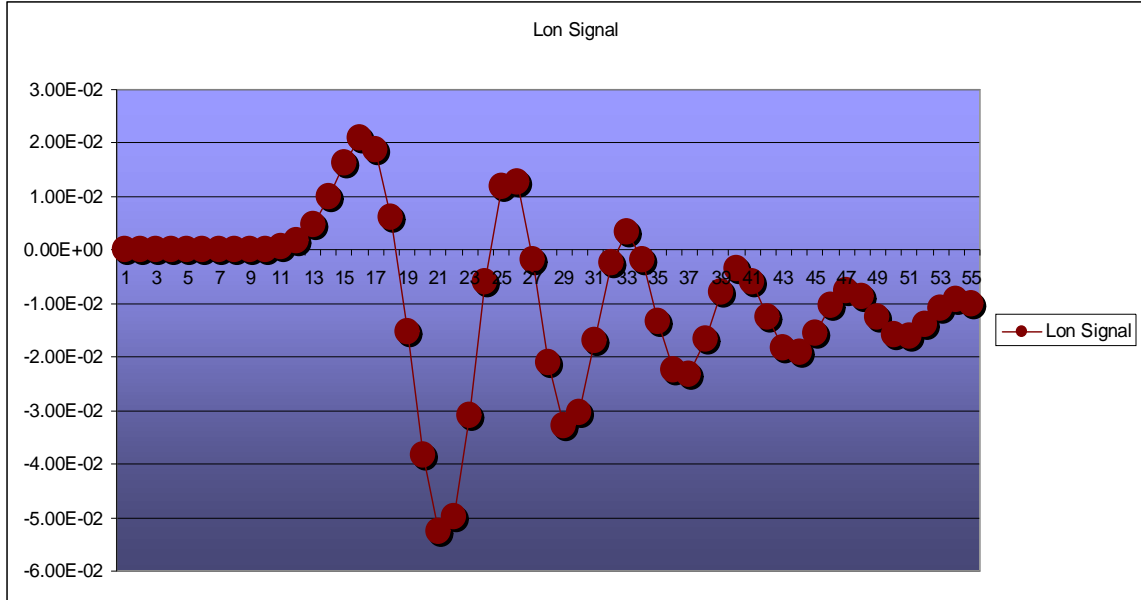


Figure 5.19: Sample Longitudinal Control Signal.

Figure 5.19 shows a sample longitudinal control signal. This signal can be mapped by having a collection of building blocks each mapping a portion of it. In figure 5.20, the same longitudinal control signal is shown along with 5 functions mapping a part of the signal. These 5 functions all have the same structure but they have different values for their parameters allowing them to be ‘active’ or non-zero for a specific interval on their range.

$$\text{Let: } u(t) = \sin(w t + r) - d \text{ (where } d < 1, \text{ around } 0.9 \text{ to } 0.999999 \text{)}$$

$$\text{Let: } f(t) = u(t) + \text{ABS}(u(t))$$

Adding $u(t)$ to the absolute value of itself makes $f(t)$ 0 except at points where the value of $u(t)$ is positive. The positive areas of the range of $u(t)$ will double their value in $f(t)$ resulting in 'peaks' similar to the ones in figure 5.18. The variable d controls the 'span' of the positive portion of $f(t)$. The smaller the value of d the wider the positive areas of $f(t)$ will be.

Adding a multiplier to control the magnitude of $f(t)$:

$$f(t) = m * [u(x) + ABS(u(x))]$$

Again, the following variables control the function and are manipulated by the GA/SA search algorithm:

- w controls the frequency of $f(t) > 0$ over its range.
- r controls where this positive value appears on the range.
- d controls the 'span' of the value.
- m controls the magnitude of the value.

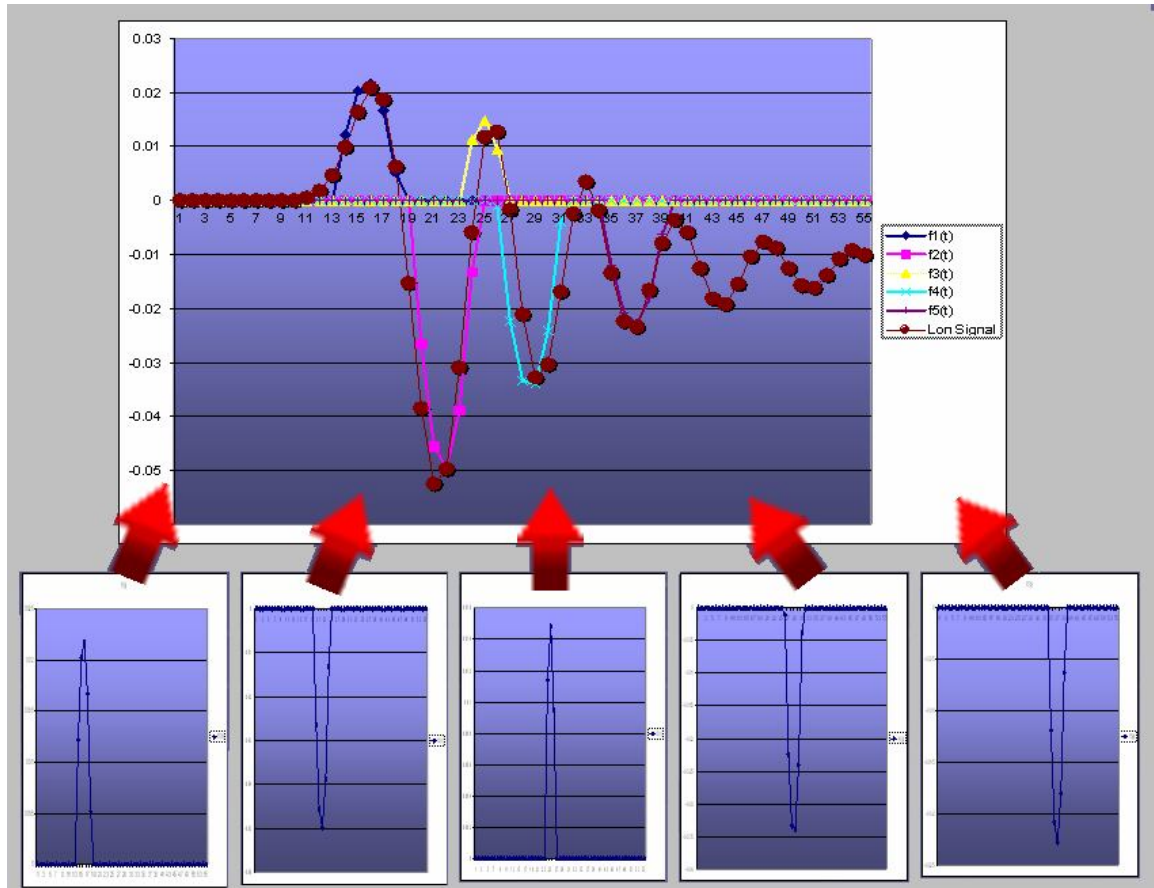


Figure 5.20: A Longitudinal Signal being Mapped by 5 Building Block Functions.

The function $f(t)$ is a single building block. The actual control function for a signal F has many building block functions each with a unique set of control variables as selected by the GA/SA search algorithm.

$$\text{Let: } F(t) = f1(t) + f2(t) + f3(t) + \dots + fn(t)$$

Where n is the number of building blocks needed to map the signal.

5.10.2 Constraints, Parameters and Assumptions

In order to make the scope of the problem more manageable, constraints were placed as follows: (1) The algorithm needs to learn a finite set of maneuvers. (2) The

GA/SA search algorithm must be able to derive control functions in a reasonable time period (a subjective value). The other variables that control the GA/SA search such as population size and Markov chain length are to be set to values that yield the best results through experimentation not to exceed 1,000 respectively. The disadvantages of the proposed method are related to the fact that simulated annealing and genetic search are not perfect, and there are disadvantages in using them. Mainly, these search methods are also not always efficient, and there is no guarantee of convergence within a given time. However, unlike gradient search, they are unlikely to get stuck in local maxima.

5.11 Running a Simulation

Simulations are run starting at low values (50,000 generations) or a starting temperature of 50,000. The value of generations/temperature is then incremented by 20,000 until an optimal (E -12 % accuracy) is found or until the pre-selected limit value of approximately 300,000 per iteration is reached. If no optimal solution is found at a generation/temperature value of 200,000, then the resulting formulas are analyzed to determine a possible cause for their lack of convergence. Based on this analysis, the mathematical primitives set is then updated and the search cycle starts over with a low value of 50,000. In case an optimal solution is found, the function is saved and the search ends.

5.12 Testing Methodology

Once results are ready, their performance must be compared to that of the fuzzy logic controller that generated the original flight path. The data from the resulting equations are entered into MATLAB and a simulation is run essentially allowing the functions to control the helicopter. The resulting flight path is then calculated and graphed along the original set points. The graph has path data of the Fuzzy logic controller as well as other controllers.

5.12.1 Test Goals

The testing phase is concerned with answering two main questions:

(1) A “static” test whereby the resulting values from the generated control equations are compared with the values in the observed data table. This test is a part of the derivation process. As the GA/SA searches, it is constantly performing a static test to determine the fitness of the formula. Hence, the goal is to map functions that fit the test data with an accuracy of E-12% or better.

(2) A “real-time” test where the generated formulas are given control of the VTOL. The resulting path is compared with the set-points (the desired path) and the path of other controllers executing the same maneuver. Hence, the ability of the generated functions to follow the flight path correctly when given control of the VTOL is tested. The functions are graphed to see how well the response curve generated by the new function $f'(x)$ matched up against the desired control signal $f(x)$.

5.13 Implementation Platform

The platform used in deriving the control equations is the Tomcat Java Server on which a program using Java Server Pages (jsp) was written. The programs were all implemented in Java in the form of jsp files. All simulations were run within Windows Explorer. The current simulation time runs anywhere from 9 to 12 hours depending on the number of rows in the input sample and other environmental variables such as the number of Building Blocks used. Figure 5.21 shows a simulation run inside Windows Explorer with the following areas being of interest:

- The current temperature.
- The current operation (op) set.
- The current combination set.
- For each op value, there is a corresponding value array.

- The current performance.
- The output values–these can be graphed.

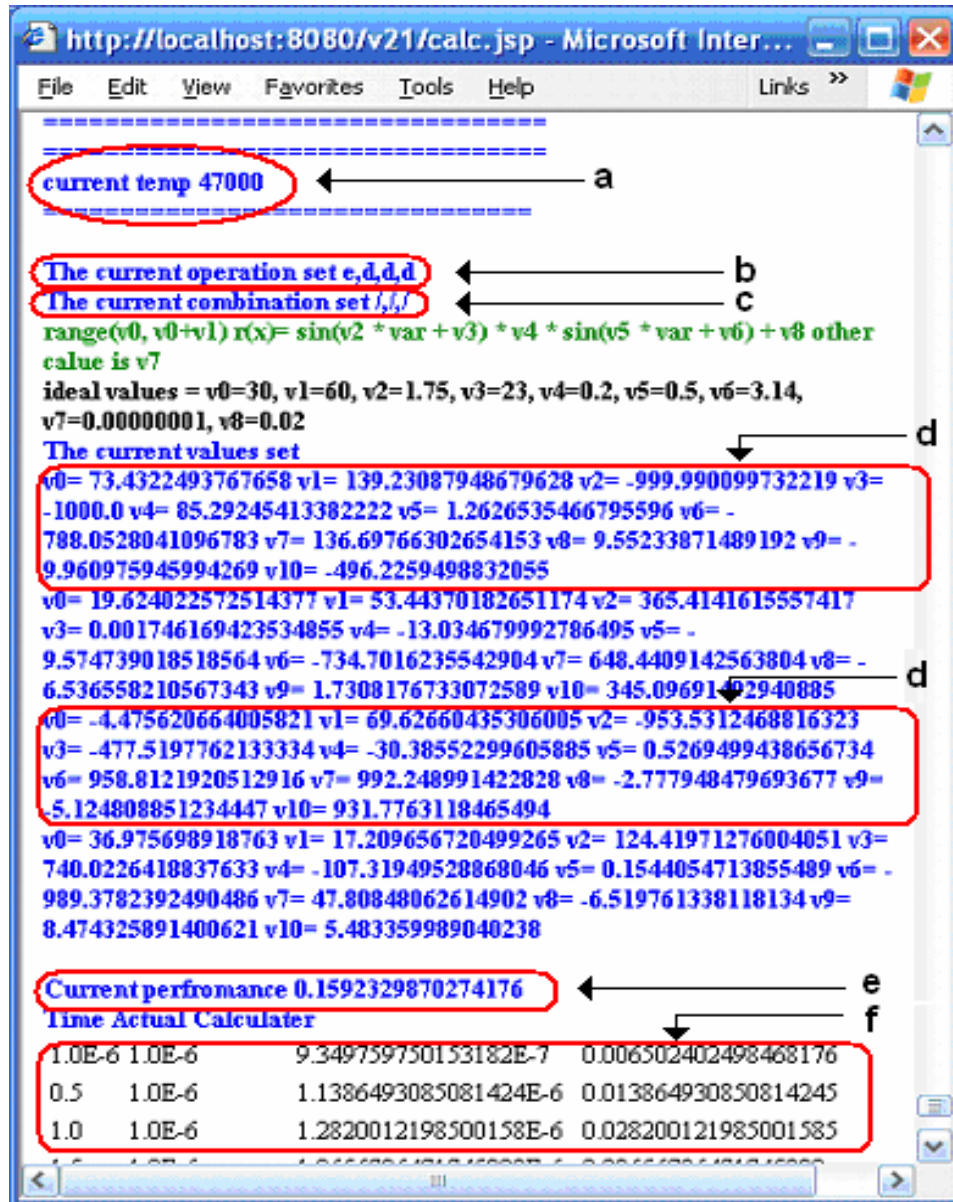


Figure 5.21: The $u(x)$ Function being Derived from the Longitudinal (lon) Control Signal.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Testing Summary

The generated GA/SA formulas are tested for fitness or goodness of fit. The closer the overall fit of the generated function to the original signal, the better the solution. The second step involves loading the data into the MATLAB helicopter model and allowing the control equations to fly the helicopter. The results for both test steps are recorded and tabulated in the following sections.

6.1.1 Summary of Static Test Results

Table 6.1 shows all 16 generated control signals (four for each maneuver) along with their approximate fitness values. Simulation run times were around 12 hours on a dual core 2.2 GHz AMD processor. The next four graphs (Figures 6.1 through 6.4) show the original fuzzy controller signal compared with the signal of the GA/SA generated function for a “take-off and u-turn” maneuver. The graphs show the four derived control signals over the flight time compared with the actual output of the fuzzy controller.

Table 6.1: Summary of Test Results.

Flight Maneuver	Control Signal	f'(x) Mapping Accuracy %
(1) Figure 8 in flight	Longitudinal	1.52184 x 10 ⁻¹³
	Lateral	9.07987 x 10 ⁻¹⁴
	Collective	3.51163 x 10 ⁻¹³
	Pedal	6.29430 x 10 ⁻¹³
(2) Takeoff and U-Turn	Longitudinal	9.17765 x 10 ⁻¹⁴
	Lateral	3.11450 x 10 ⁻¹³
	Collective	2.82027 x 10 ⁻¹³
	Pedal	9.51773 x 10 ⁻¹³
(3) Ascending Spiral	Longitudinal	3.38715 x 10 ⁻¹³
	Lateral	1.57193 x 10 ⁻¹⁴
	Collective	8.8501 x 10 ⁻¹⁴
	Pedal	1.50658 x 10 ⁻¹⁴
(4) Figure 8, changing height	Longitudinal	2.23580 x 10 ⁻¹⁰
	Lateral	5.33174 x 10 ⁻¹⁴
	Collective	8.90410 x 10 ⁻¹³
	Pedal	1.57193 x 10 ⁻¹⁴

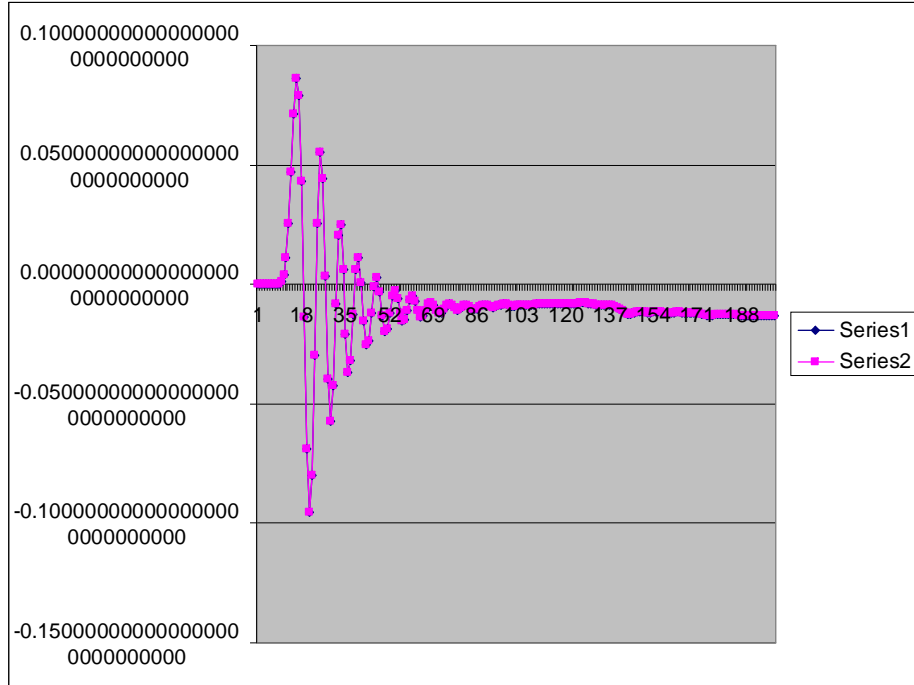


Figure 6.1: Collective Signal Mapping Accuracy is E-12 or Better. Note that the Blue Line is Indiscernible from the Pink Line Due to the High Mapping Accuracy.

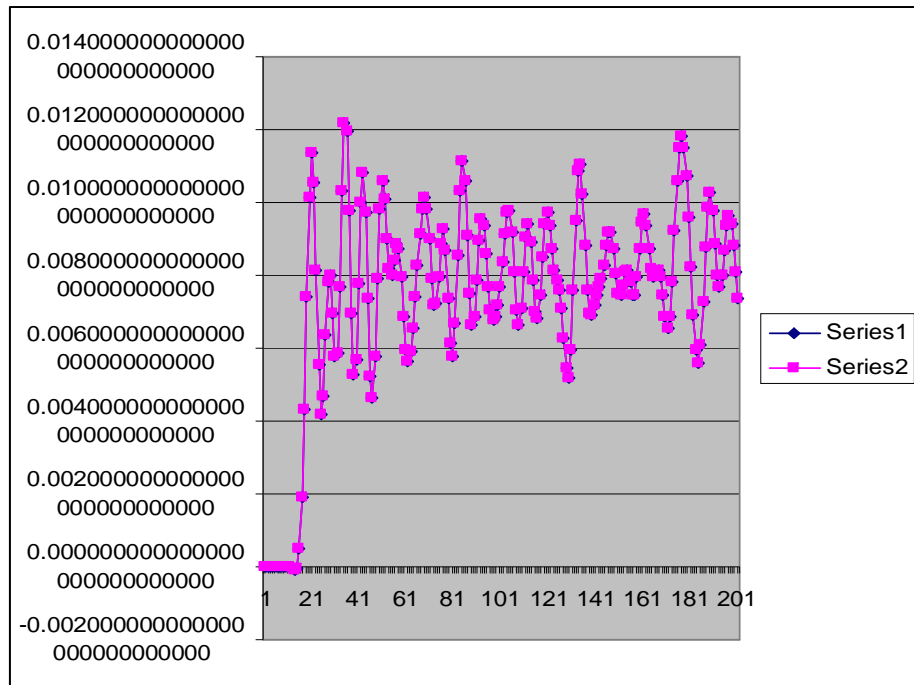


Figure 6.2: Lateral Signal Mapping Accuracy is E-12 or Better.

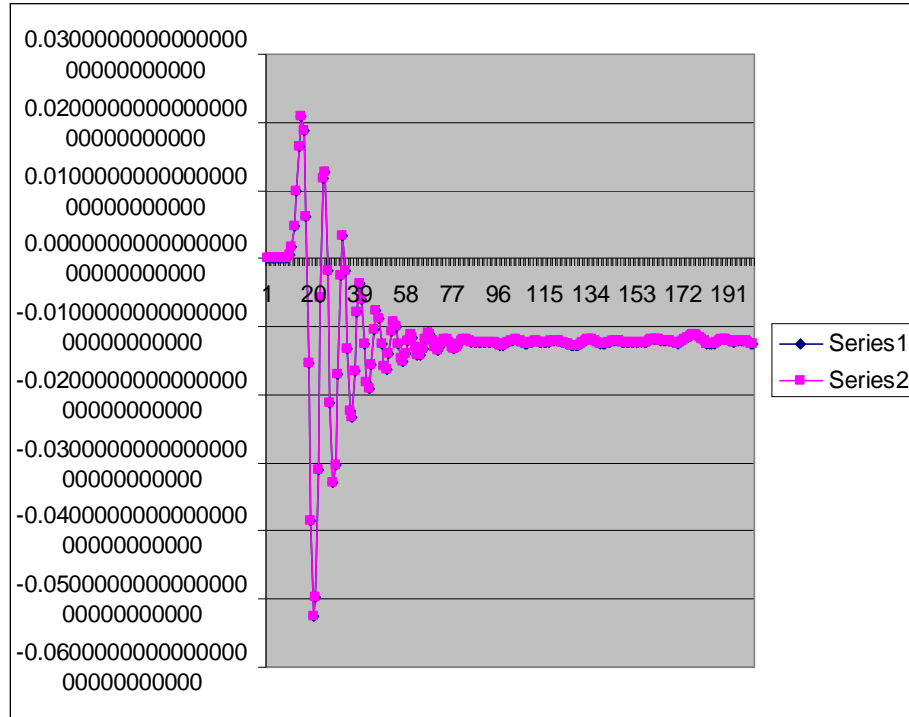


Figure 6.3: Longitudinal Signal Mapping Accuracy is E-12 or Better.

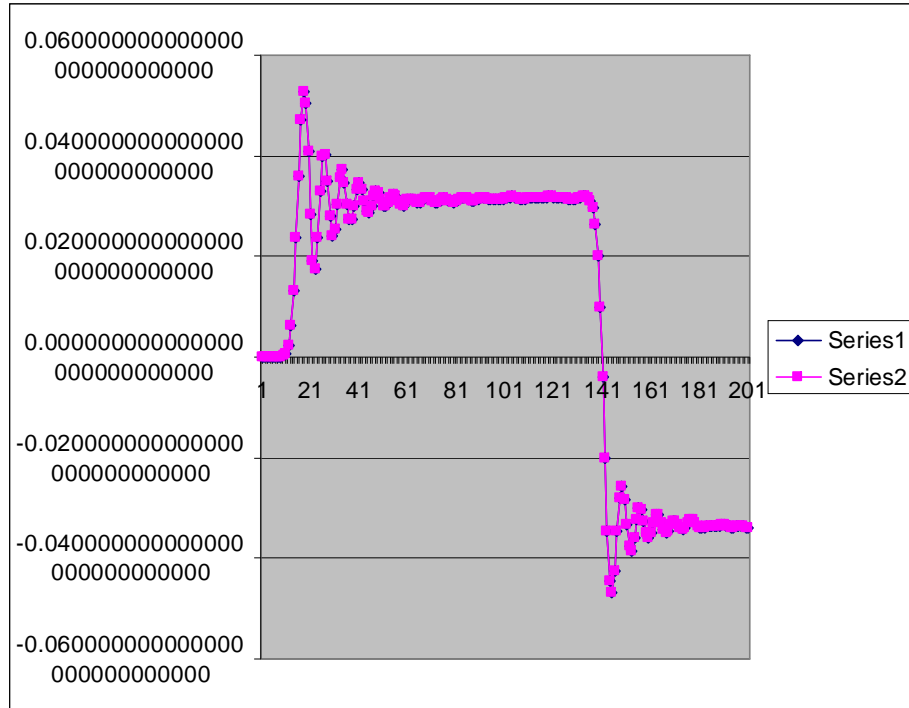


Figure 6.4: Pedal Signal Mapping Accuracy is E-12 or Better.

It is important to note that the predicted value starts with a large error and then as the annealing temperature approaches zero, the generated function gets better at approximating the output signal. The annealing temperature used in the run that generated the graphs started at 90000. The same holds in the genetic algorithm based search, the error is large at the start of the search and then decreases as the number of generations increase.

A sample solution is shown below:

$$\begin{aligned}
 g(x) = & -7.808822922194903 * (\text{Math.abs}(\text{Math.sin}(-14.739860223124678 * 84.0 + - \\
 & 3.8552882647706337) * \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + \\
 & \text{Math.abs}(\text{Math.sin}(-14.739860223124678 * 84.0 + -3.8552882647706337)) * \\
 & \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + -2.7234666674927006E-13 * \\
 & (\text{Math.abs}(\text{Math.sin}(18.162399204037563 * 84.0 + 1.1859891106685911)) * \\
 & \text{Math.sin}(\text{values}[7] * \text{input}[0] + \text{values}[8]) - \text{values}[15]) + \\
 & \text{ath.abs}(\text{Math.sin}(18.162399204037563 * 84.0 + 1.1859891106685911)) * \\
 & \text{Math.sin}(\text{values}[7] * \text{input}[0] + \text{values}[8]) - \text{values}[15]) + 4.082428387314575E-4 * \\
 & 84.0 + -0.01893125338865113 + 0.0 -9.256986097378053E-9 * (\\
 & \text{Math.abs}(\text{Math.sin}(2.331363503902123 * 2.4 + -0.31857784396828004)) * \\
 & \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + \\
 & \text{ath.abs}(\text{Math.sin}(2.331363503902123 * 2.4 + -0.31857784396828004)) * \\
 & \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + -2.960800703770896E-7 * (\\
 & \text{Math.abs}(\text{Math.sin}(-9.06226053214201 * 2.4 + 1.214464920141008E-4)) * \\
 & \text{Math.sin}(\text{values}[7] * \text{input}[0] + \text{values}[8]) - \text{values}[15]) + \text{Math.abs}(\text{Math.sin}(- \\
 & 9.06226053214201 * 2.4 + 1.214464920141008E-4)) * \text{Math.sin}(\text{values}[7] * \text{input}[0] + \\
 & \text{values}[8]) - \text{values}[15]) + -1.0750681422502352E-14 * 2.4 + -1.0991822929979367E- \\
 & 15 + 0.0 + 0.09836145446757635 * (\text{Math.abs}(\text{Math.sin}(1.0923894221893784 * 4.8 + \\
 & 3.4040567600035927) * \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + \\
 & \text{ath.abs}(\text{Math.sin}(1.0923894221893784 * 4.8 + 3.4040567600035927)) * \\
 & \text{Math.sin}(\text{values}[3] * \text{input}[0] + \text{values}[4]) - \text{values}[17]) + 8.973723301401056E-5 * (
 \end{aligned}$$

$$\begin{aligned} & \text{Math.abs}(\text{Math.sin}(-25.979710813021498 * 4.8 + 0.03716609615194822) * \\ & \text{Math.sin}(\text{values}[7] * \text{input}[0] + \text{values}[8]) - \text{values}[15]) + \text{Math.abs}(\text{Math.sin}(- \\ & 25.979710813021498 * 4.8 + 0.03716609615194822)) * \text{Math.sin}(\text{values}[7] * \text{input}[0] + \\ & \text{values}[8]) - \text{values}[15]) + 4.4140232393466586E-5 * 4.8 + 1.1684463372896668E-5 \end{aligned}$$

6.2 Summary of Dynamic Test Results

In this section, the equations are loaded into the MATLAB model of the virtual helicopter. The equations are then given control of the helicopter to determine how well they can follow the fuzzy logic controller's path. The only input provided to the equations is flight time. The next sections summarize each maneuver and the results associated with that maneuver, organized one section per maneuver. For each of the four maneuvers, the helicopter was first flown using the fuzzy logic controller with its feedback intact (closed-loop). The helicopter is then flown using the fuzzy logic controller in open-loop mode. The resulting path data of the fuzzy logic controller for both the open-loop and closed-loop modes are shown in a graph. The graph compares the closed-loop (pink) vs. the open-loop mode (blue) flight path. This is done to show the difference between closed-loop and open-loop operation for each maneuver. The fuzzy logic controller is operated in open-loop in order to collect control signal data that can be used to derive control equations. Note that the GA/SA algorithm derives formulas that will operate in open-loop mode. Hence, collecting data should be done from an open-loop fuzzy logic controller.

Next, the derived equations are given control of the helicopter model and the resulting path is compared with the fuzzy-logic controller path (open-loop mode). The resulting path is also shown for each axis the graphs that follow. Lastly, the flight path of the derived equations, the fuzzy logic controller (open-loop mode), other controllers and the set-points are all shown together in one graph.

6.2.1 Figure-8 Maneuver

This maneuver requires the VTOL to perform a figure-8 in flight. Figure 6.5 shows the open-loop/closed-loop paths. Note that when the fuzzy logic controller is placed in open-loop, it becomes unable to complete the whole maneuver as can be seen in the figure below. This is due to the feedback circuit being disconnected. Throughout testing, it was noted that the VTOL was more sensitive to some maneuvers when placed in open-loop mode. Some of the maneuvers exhibit large errors in the flight path towards the end. Meanwhile, other maneuvers seem to run well regardless of the open-loop or closed loop status of the controller. As such, different maneuver and helicopter models will likely respond differently to the proposed approach; that of flying the VTOL in open loop.

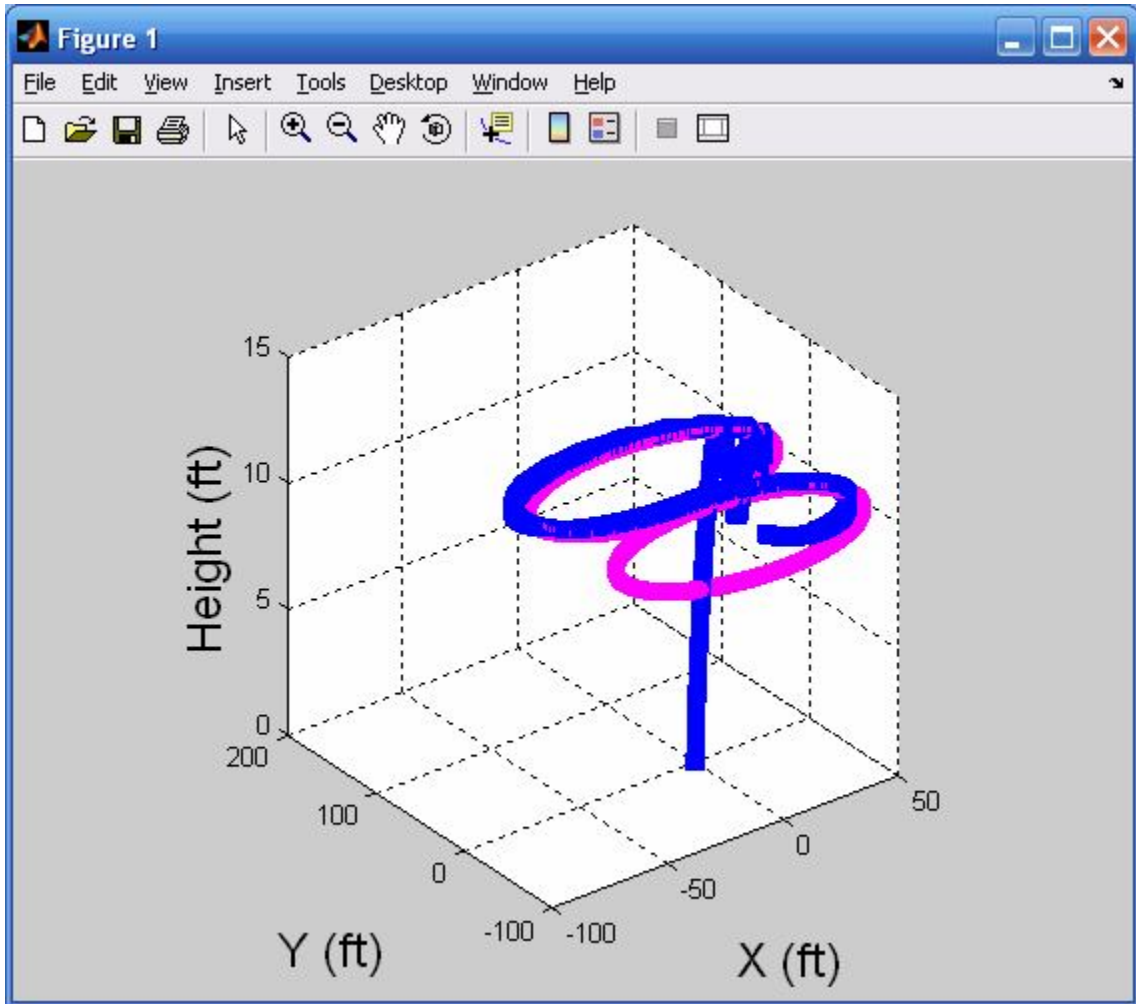


Figure 6.5: Figure-8 (in flight loop) Closed Mode and Open Mode; Showing. x, y, z vs. Time Plot, with Blue being the Fuzzy-logic Controller (open-mode) Path and Pink being the Fuzzy-logic Controller (closed-mode) Path.

6.2.2 Figure-8 Maneuver Testing Results

The figure-8 maneuvers are shown again below. The graph in figure 6.6 compares the GA/SA based control formula flight path with that of the fuzzy logic controller (open mode) flight path. The same results are shown one axis at a time in figures (6.7), (6.8), and (6.9) respectively.

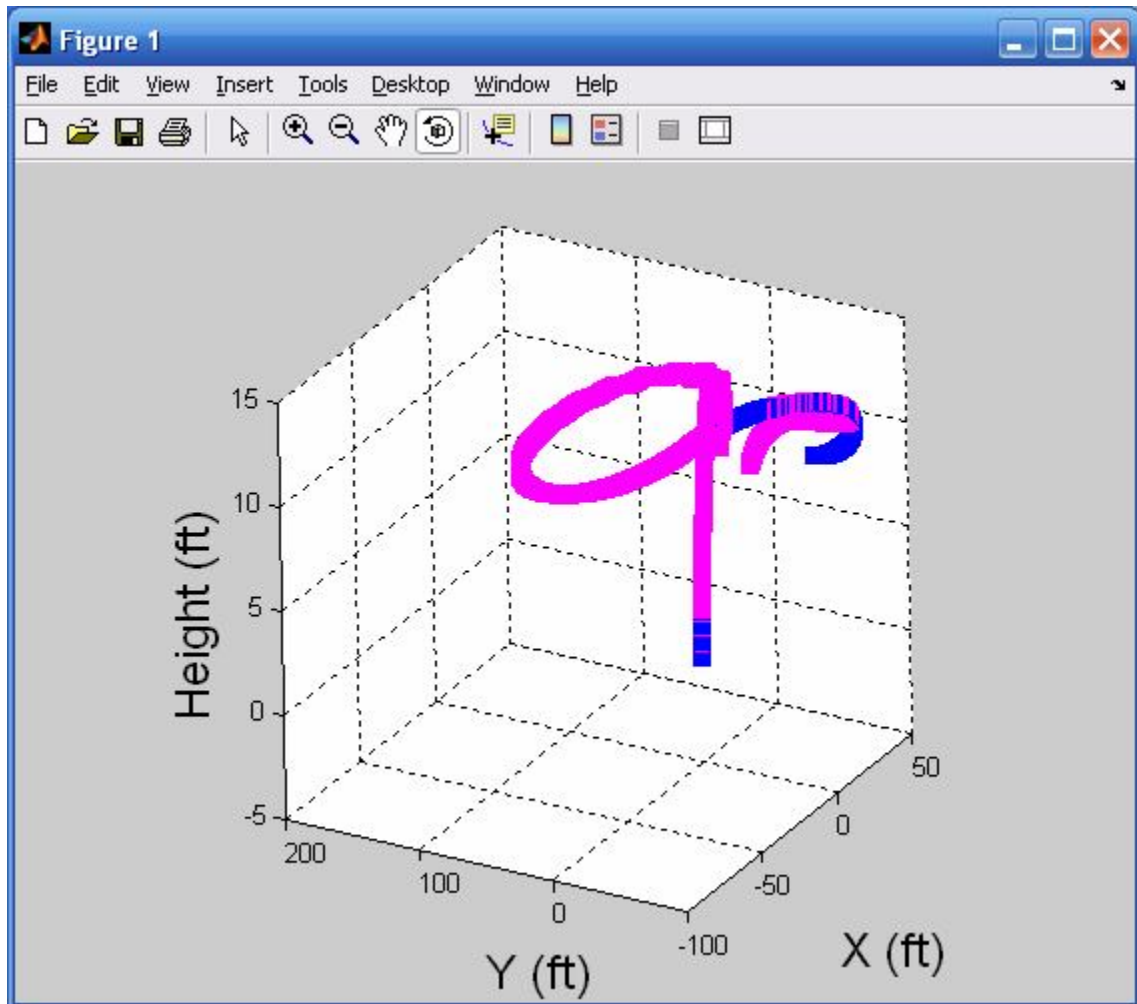


Figure 6.6: Figure-8, GA/SA Path and Fuzzy Logic Controller Path; Showing x , y , z vs. Time Plot, with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

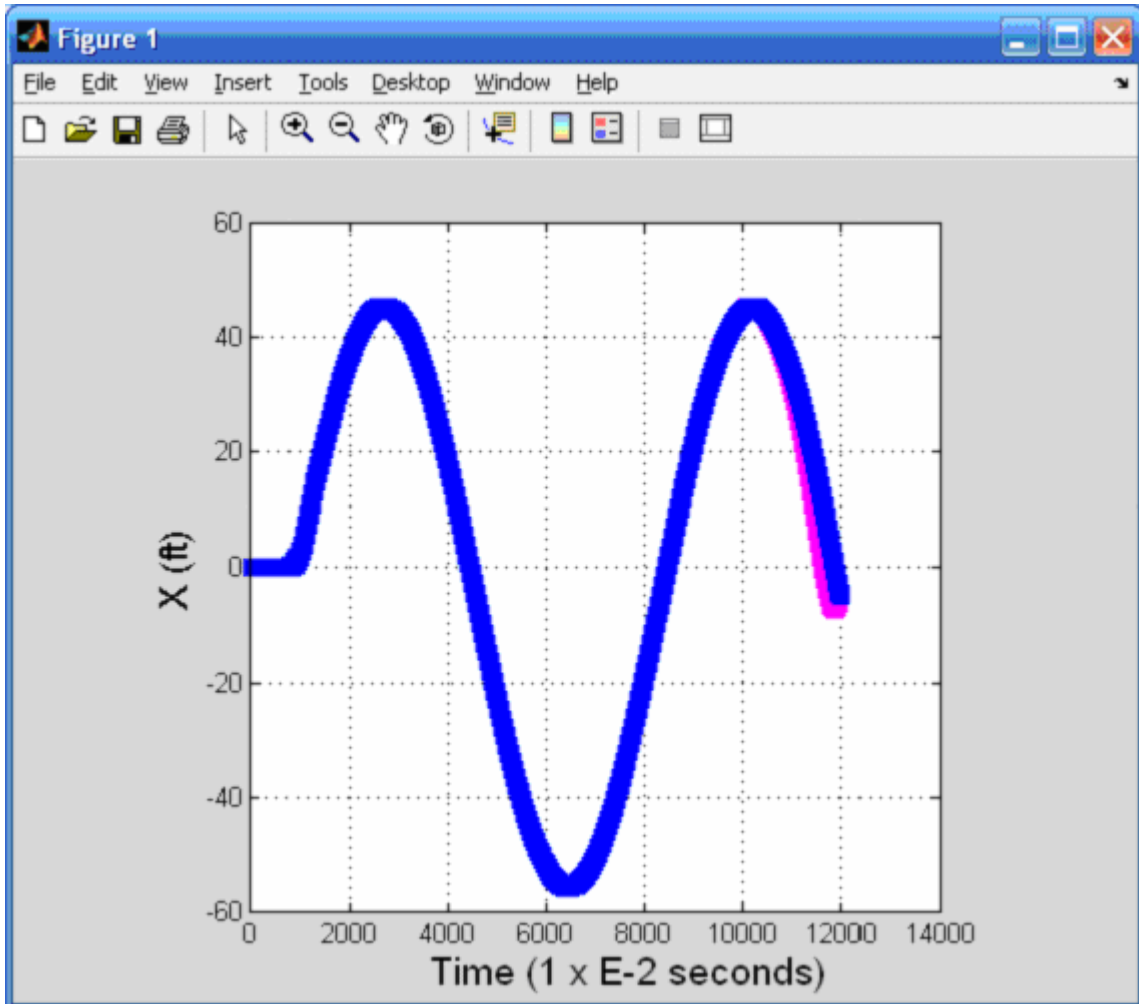


Figure 6.7: *Figure-8, x vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.*

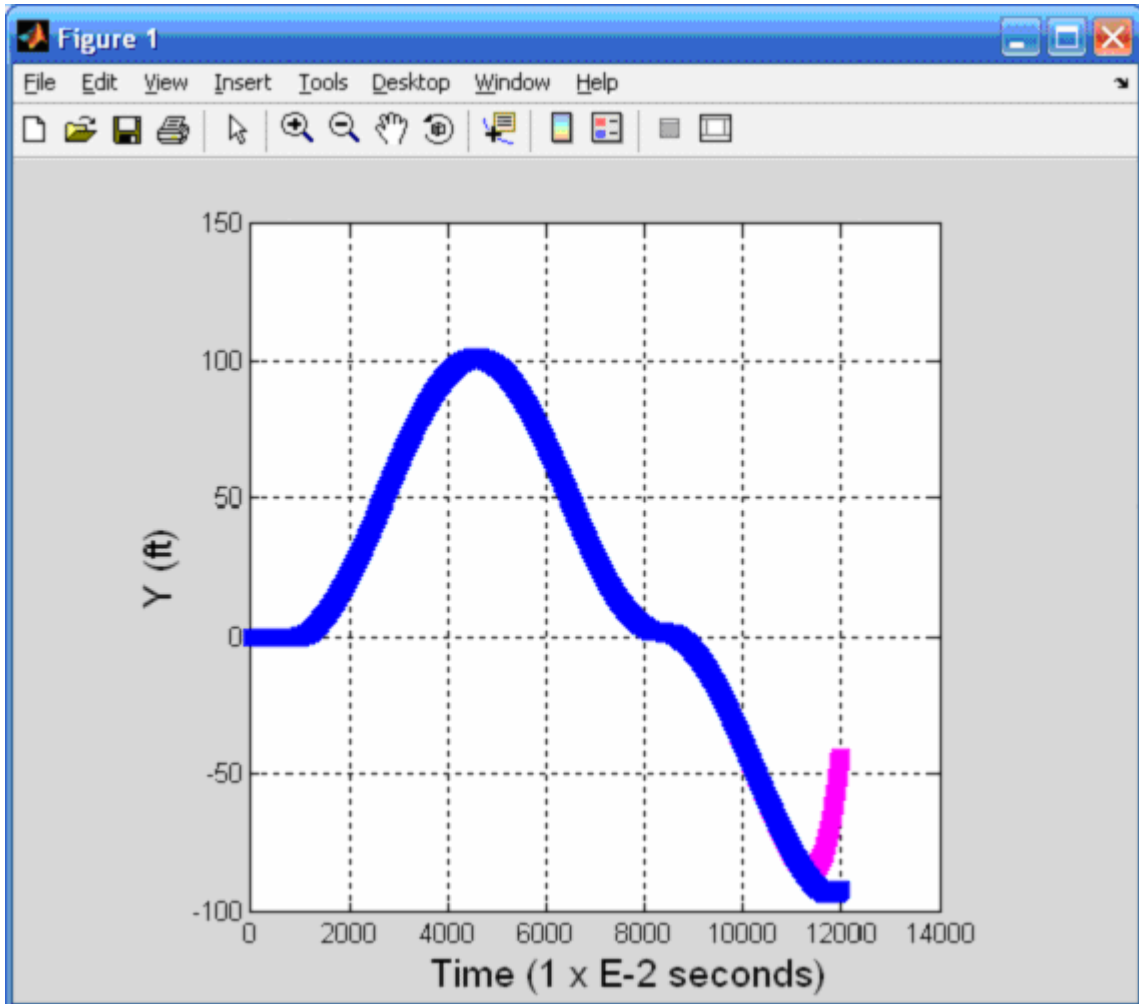


Figure 6.8: *Figure-8, y vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.*

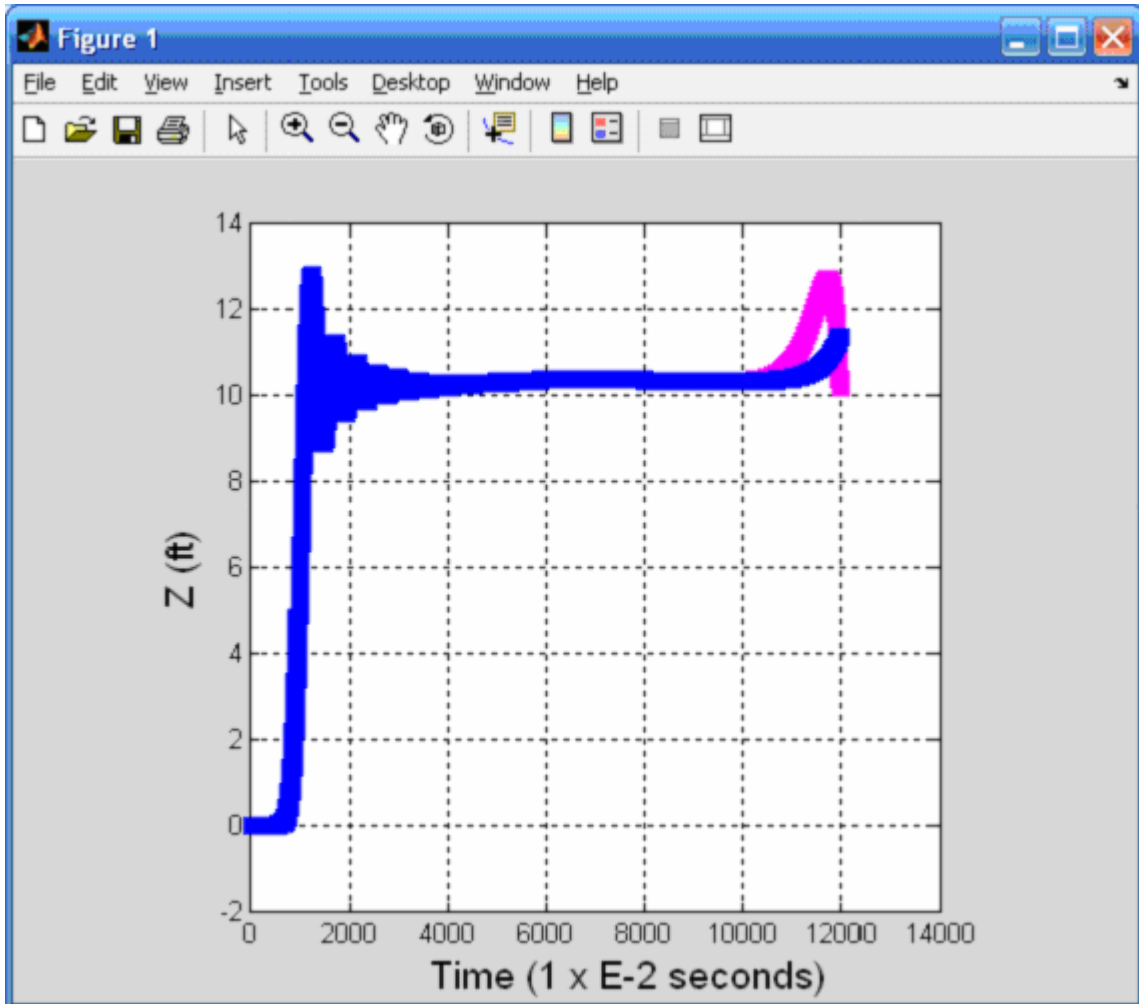


Figure 6.9: Figure-8, z vs. Time Plot with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

Once the flight path for the figure-8 in flight maneuver is completed, it is then compared with the set-points path (the desired path), a PID Velocity Tracking controller, a PID Position Tracking controller, a Model Predictive Control (MPC) Position Tracking controller, the GA/SA derived equations path, and the fuzzy logic controller path which was the basis for the derived equations. Figure 6.10 shows the flight paths of all the controllers running a figure-8 within MATLAB. The results show that this maneuver was difficult for some controllers, such as the PID Velocity tracking controller, to follow. The GA/SA controller, running in open-loop, seems to follow the set-points reasonably well.

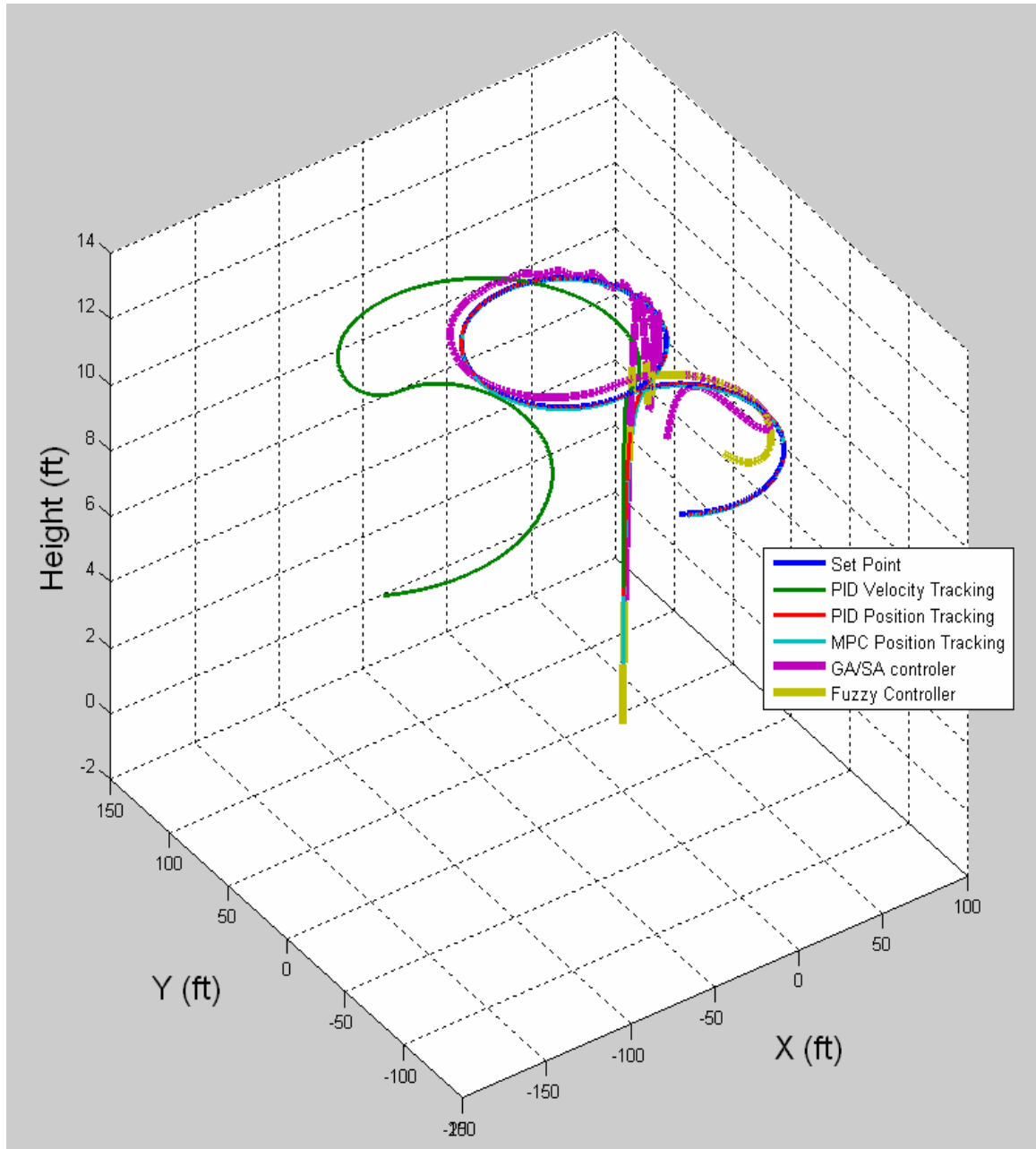


Figure 6.10: Figure-8, Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB.

6.2.3 U-Turn Maneuver

This maneuver requires the VTOL to make a u-turn in flight. Figure 6.11 shows the open loop/closed loop paths. In this maneuver, the open-loop and closed loop flight

paths were close. Note that the fuzzy-logic controller is not affected by open-loop significantly.

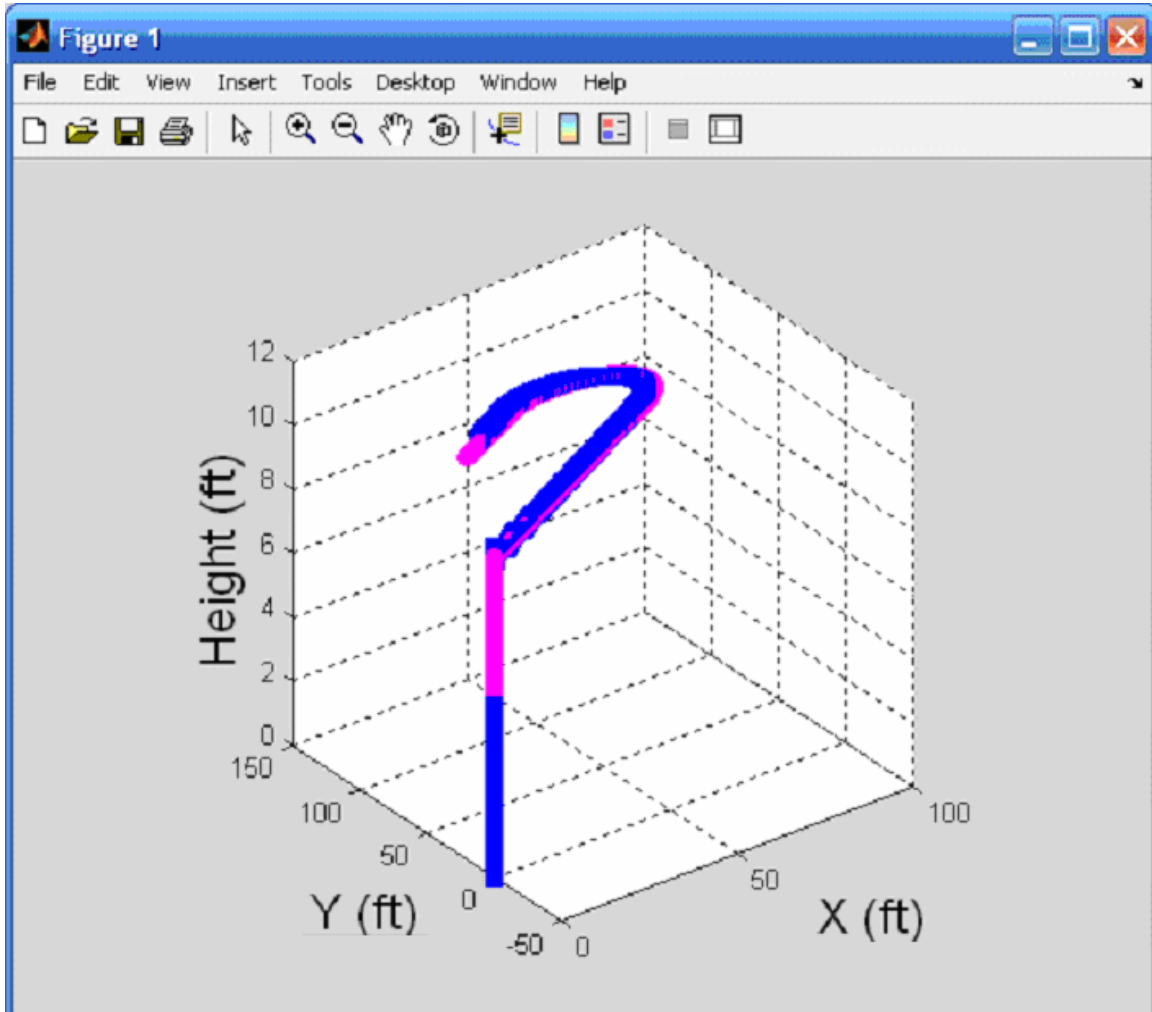


Figure 6.11: U-turn (in flight) Closed Mode and Open Mode; Showing x, y, z vs. Time Plot, with Blue being the Fuzzy-logic Controller (open-mode) Path and Pink being the Fuzzy-logic Controller (closed-mode) Path.

6.2.4 U-Turn Maneuver Testing Results

The comparison of the fuzzy logic controller's open-mode path and the GA/SA derived control equation's flight path is shown in figure (6.12). The same results are shown one axis at a time in figures (6.13), (6.14) and (6.15) respectively.

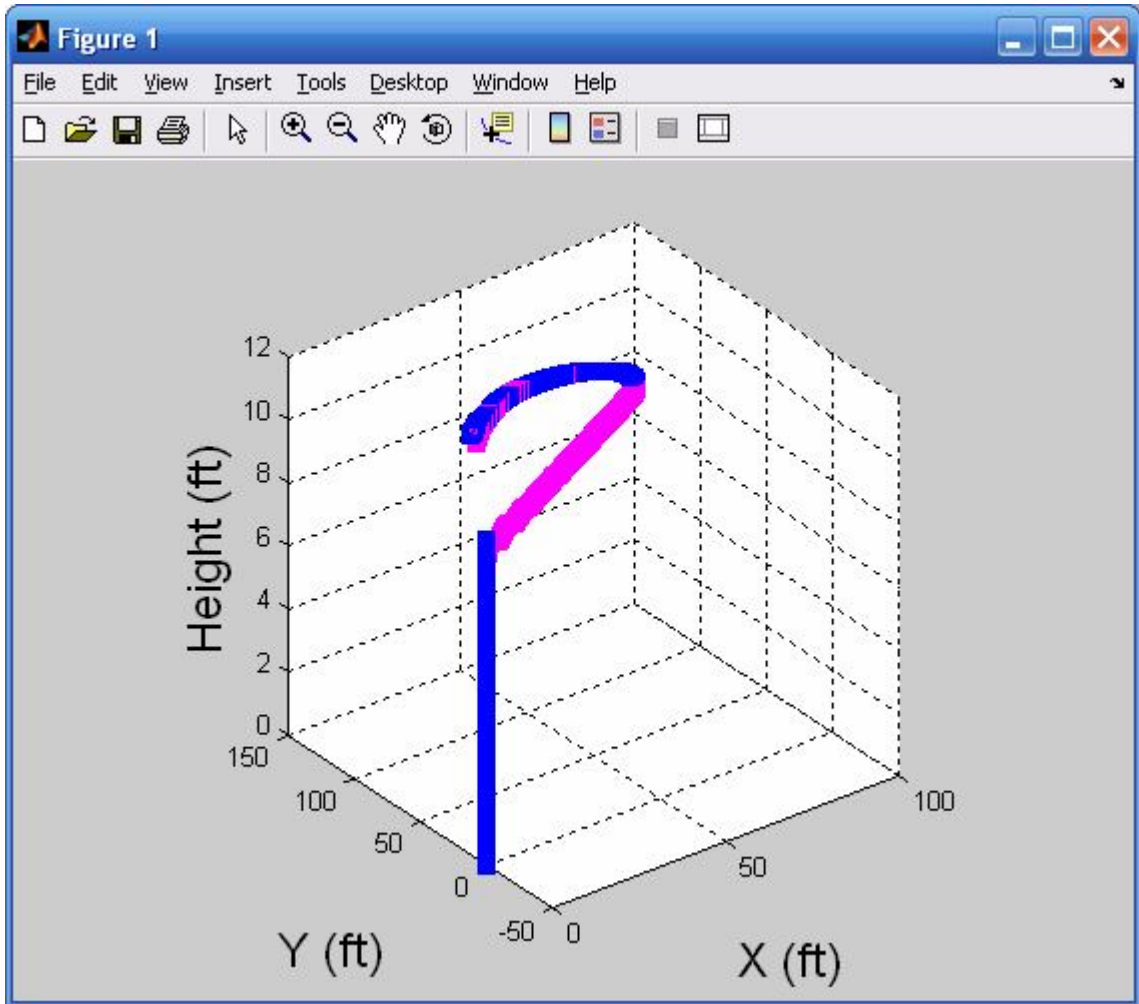


Figure 6.12: U-turn, GA/SA Path and Fuzzy Logic Controller Path; Showing x, y, z vs. Time Plot, with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

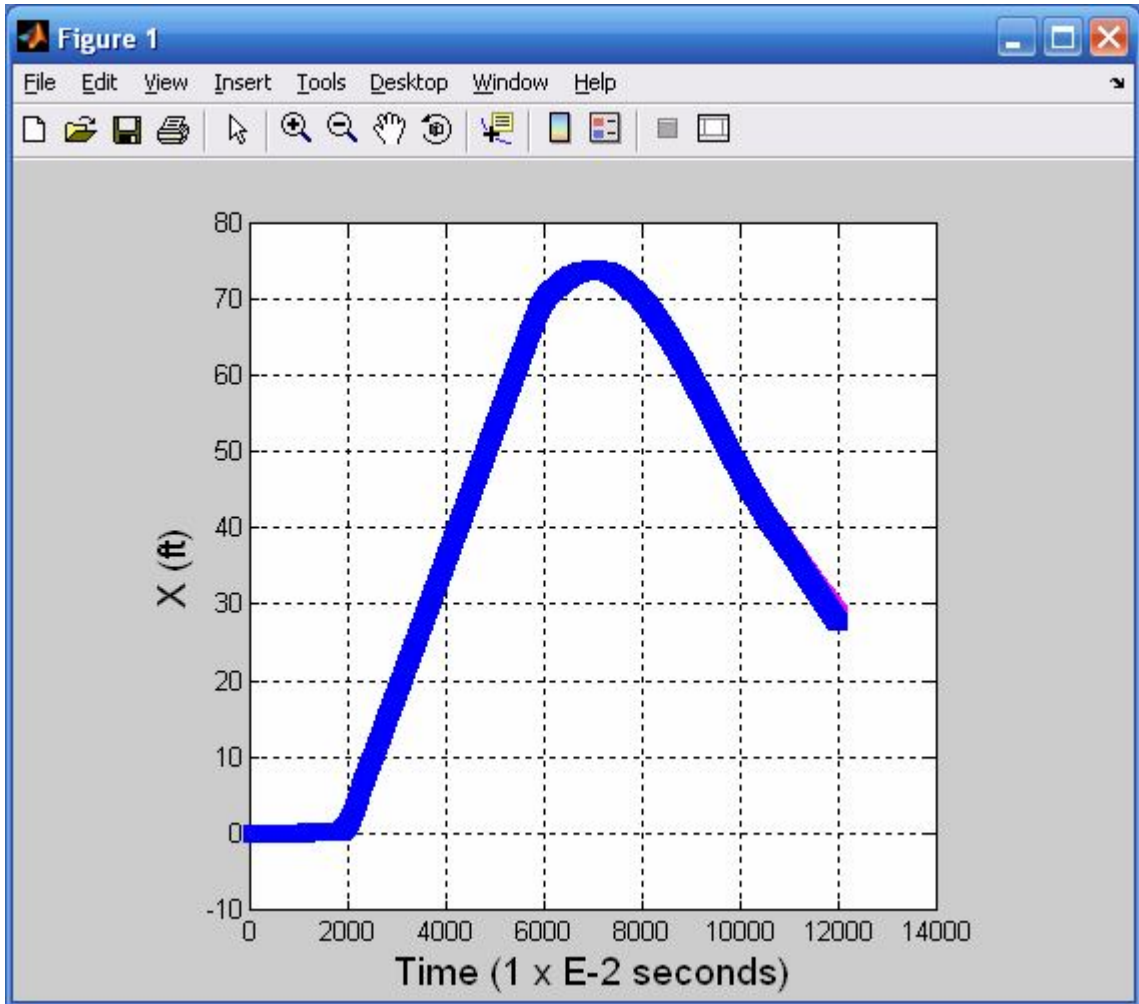


Figure 6.13: U-turn, x vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

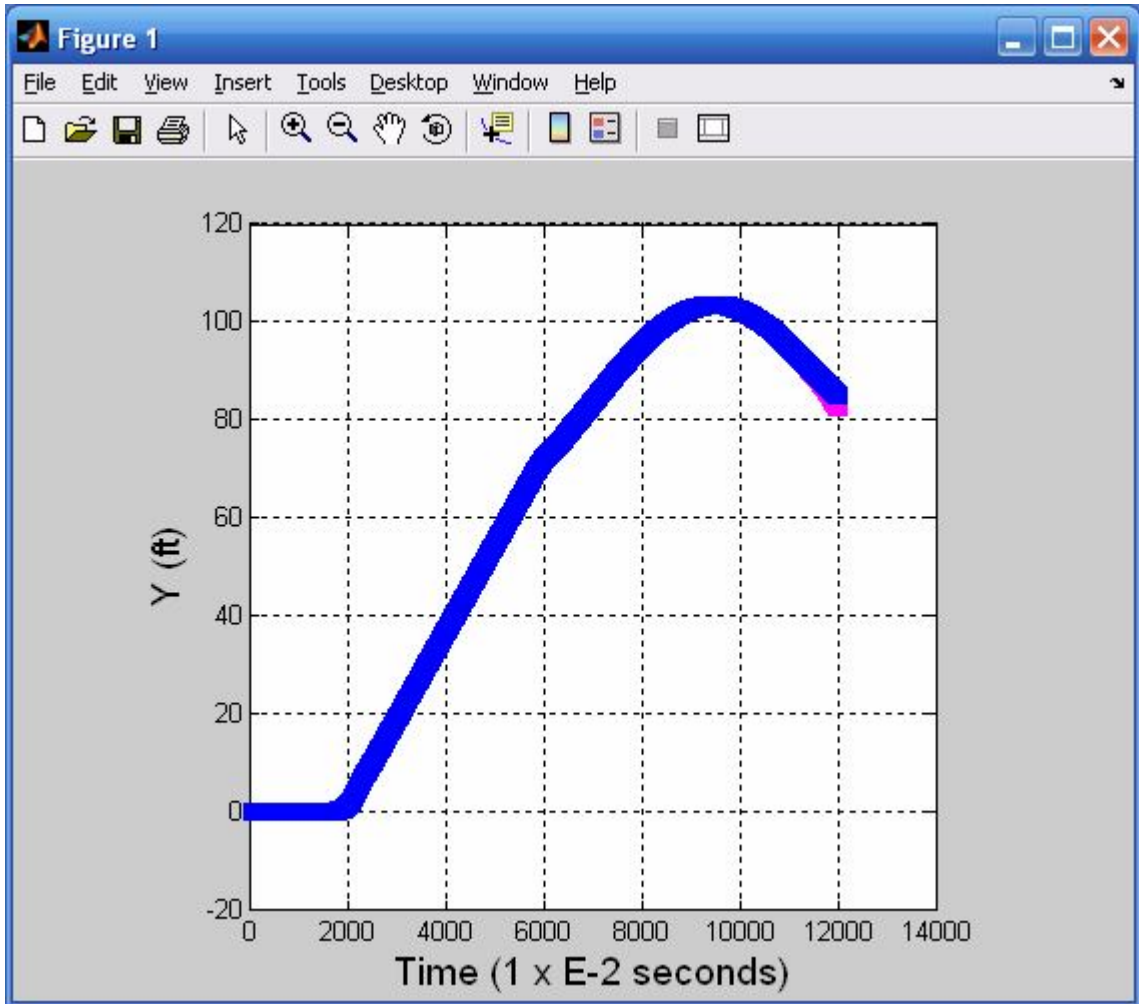


Figure 6.14: U-turn, y vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

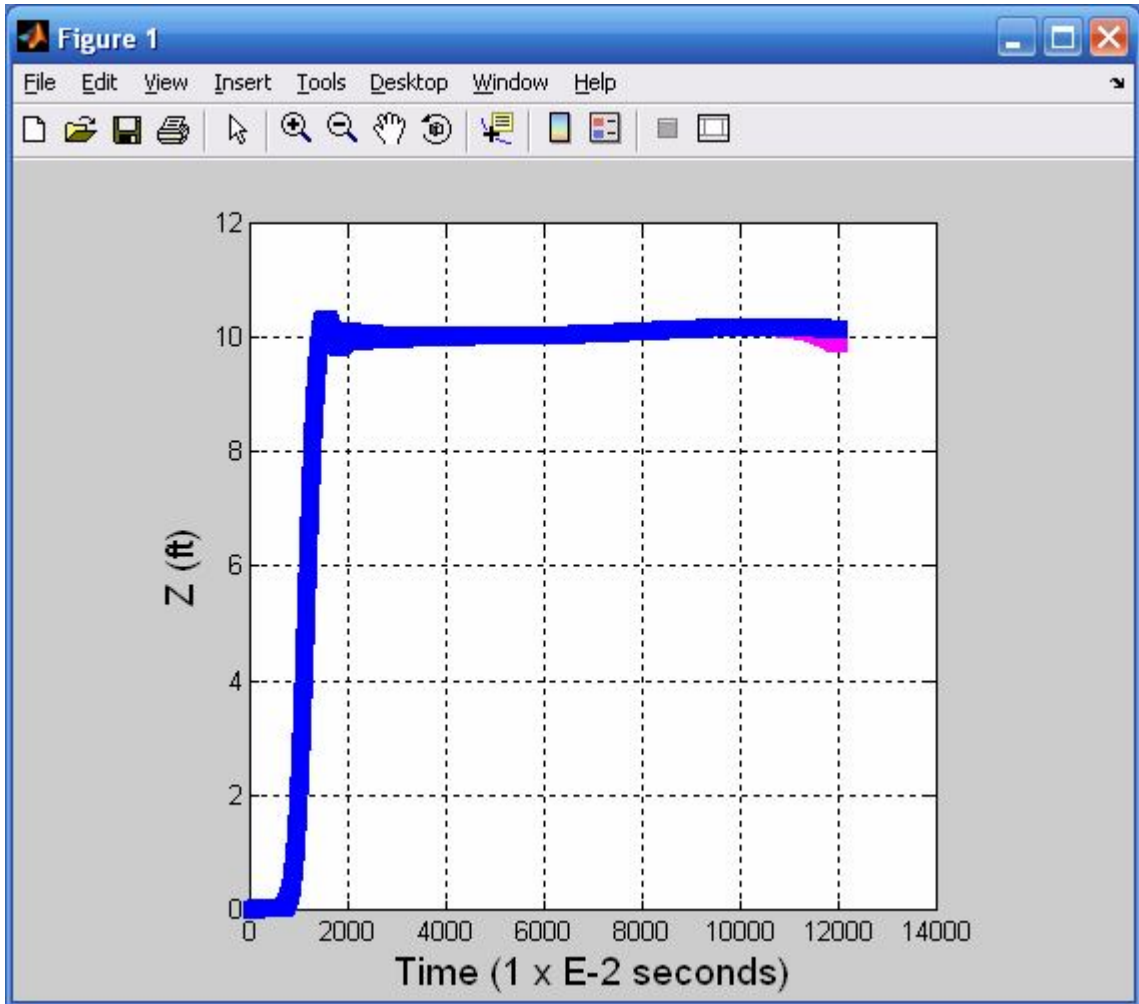


Figure 6.15: U-turn, z vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

Again, the u-turn flight path is compared with the set-points path, a PID Velocity Tracking controller, a PID Position Tracking controller, a MPC Position Tracking controller, the GA/SA controller path, and the fuzzy logic controller path. Figure 6.16 shows the flight paths of all the controllers running within MATLAB. Note that all controllers seem to be able to follow the set-points very well. In this case, open-loop or closed-loop control seems to make little difference in accuracy.

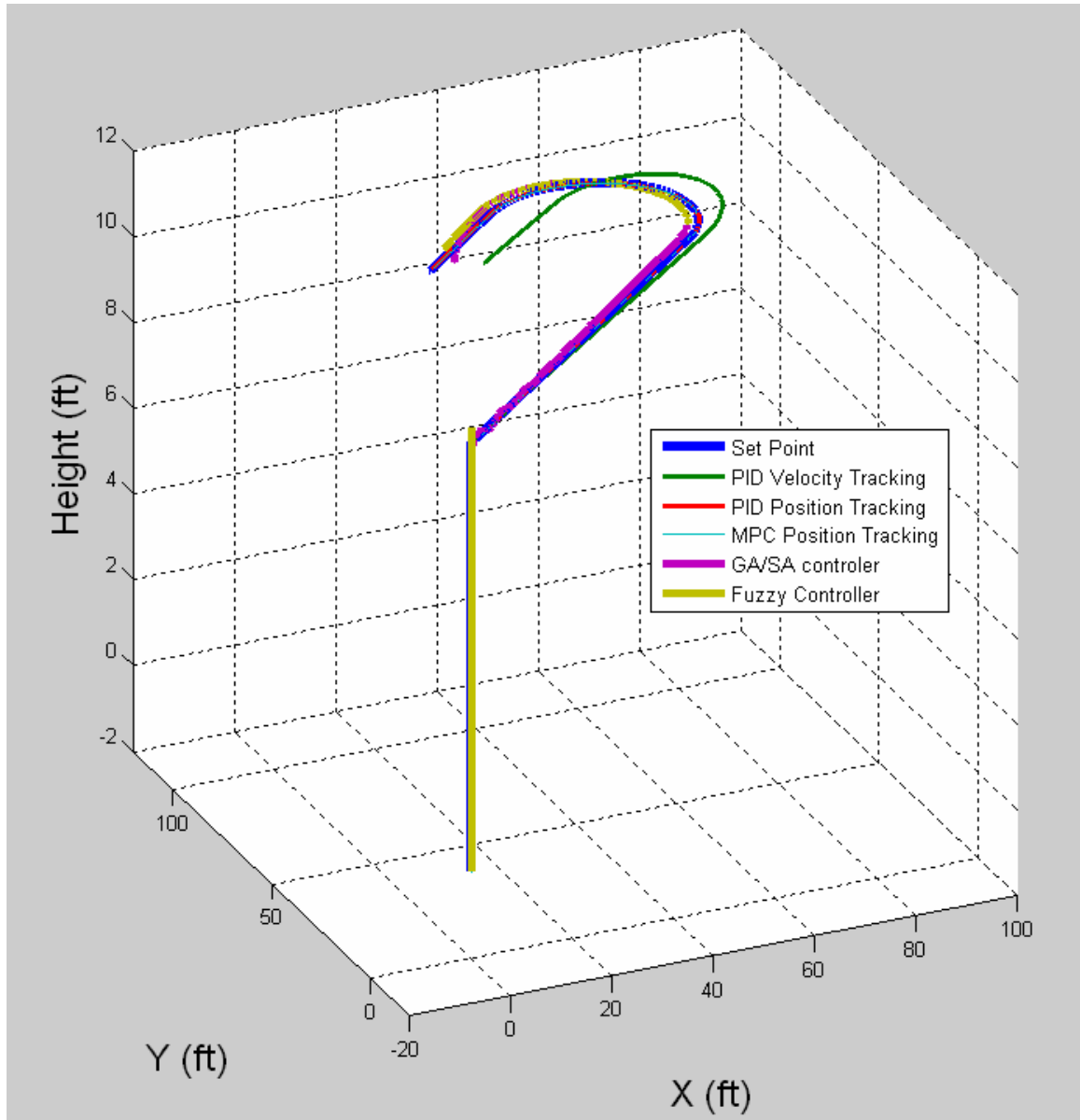


Figure: 6.16 U-turn, Showing the GA/SA Derived Controller Compared to Other Controllers Flying the Same Model/Set-points in MATLAB.

6.2.5 Ascending Spiral

This maneuver requires the VTOL to spiral up. Figure 6.17 shows the open-loop/closed-loop paths. In this maneuver, the open-loop and closed-loop flight paths were different as can be seen in the figure below. Note that the fuzzy-logic controller does not

match the closed-loop mode version of the spiral maneuver perfectly, this is due to it being in open-loop mode.

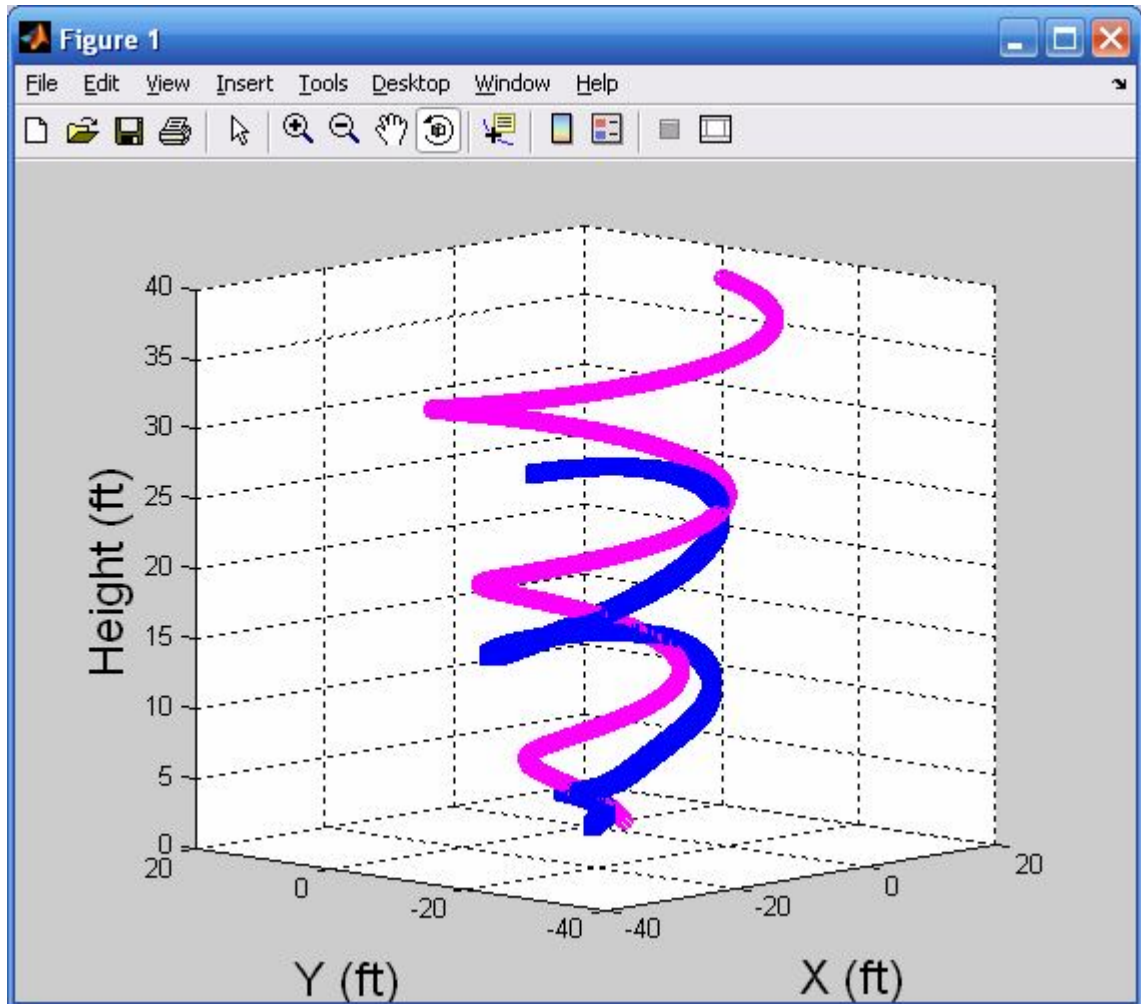


Figure 6.17: *Ascending Spiral Maneuver Closed Mode and Open Mode; Showing x, y, z vs. Time Plot, with Blue being the Fuzzy-logic Controller (open-mode) Path and Pink being the Fuzzy-logic Controller (closed-mode) Path.*

6.2.6 Ascending Spiral Testing Results

The comparison of the open-loop model path and the GA/SA derived control equation's path is shown in figure (6.18). The same results are shown one axis at a time in figures (6.19), (6.20), and (6.21) respectively.

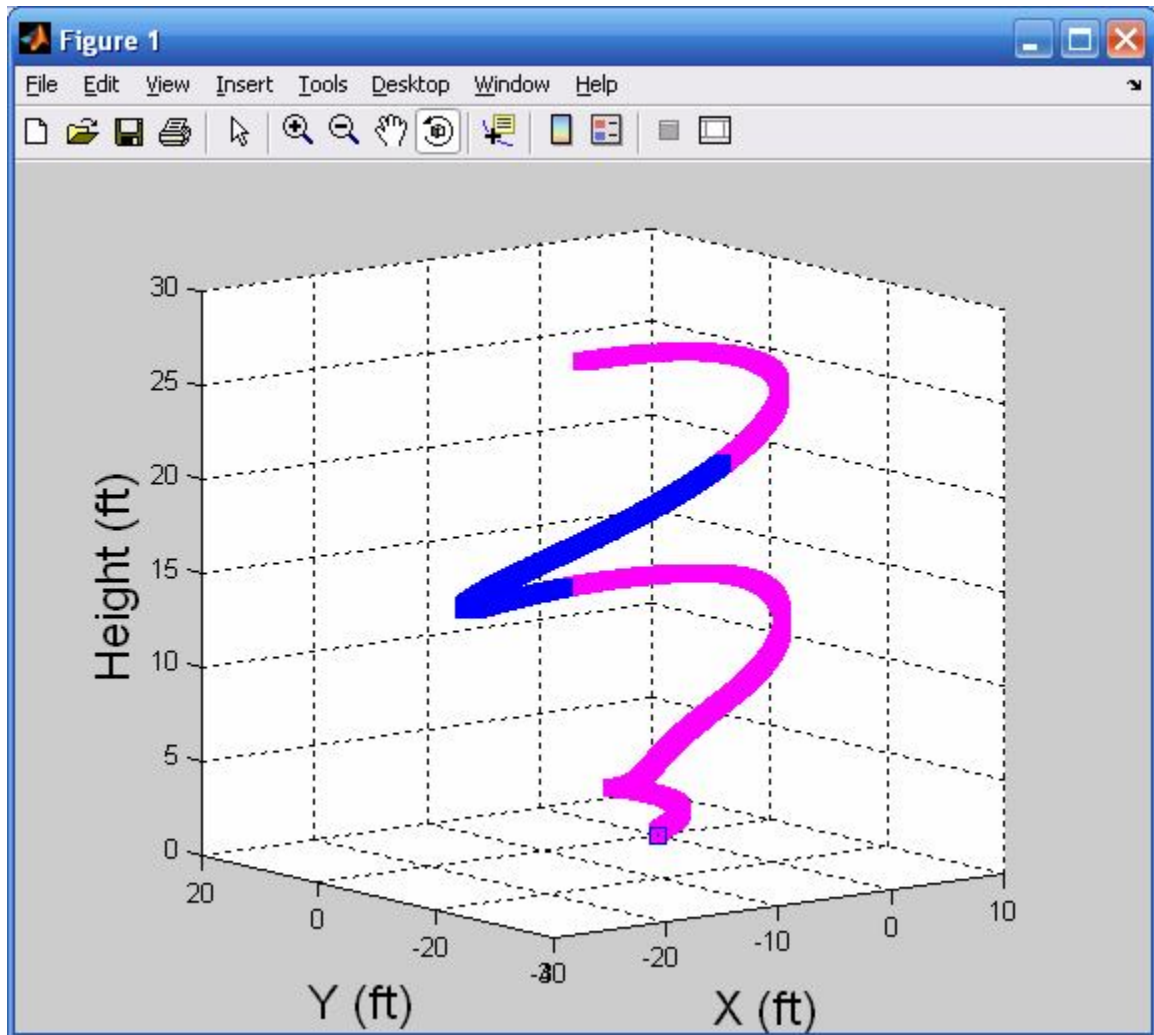


Figure 6.18: Ascending Spiral, GA/SA Path and Fuzzy Logic Controller Path; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

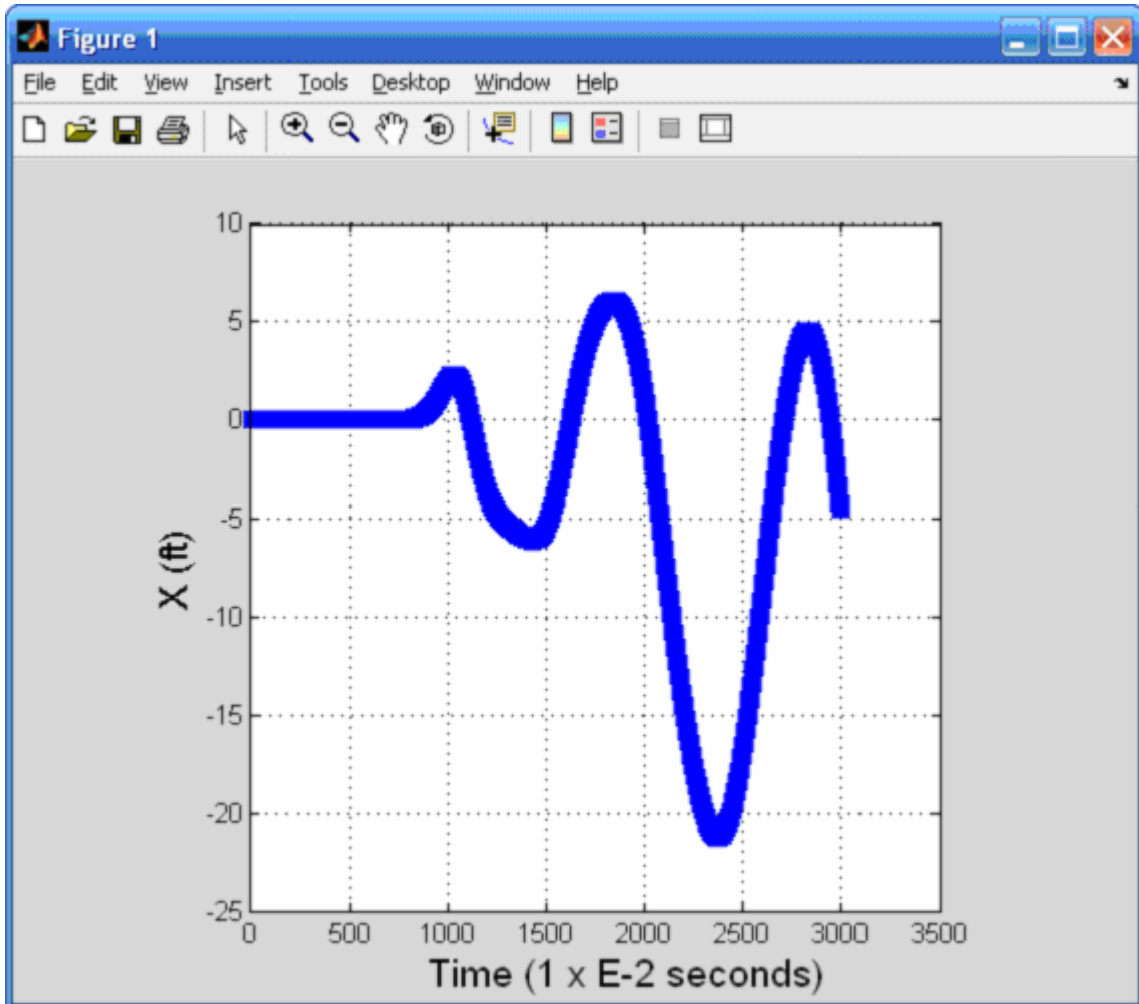


Figure 6.19: Ascending Spiral, x vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

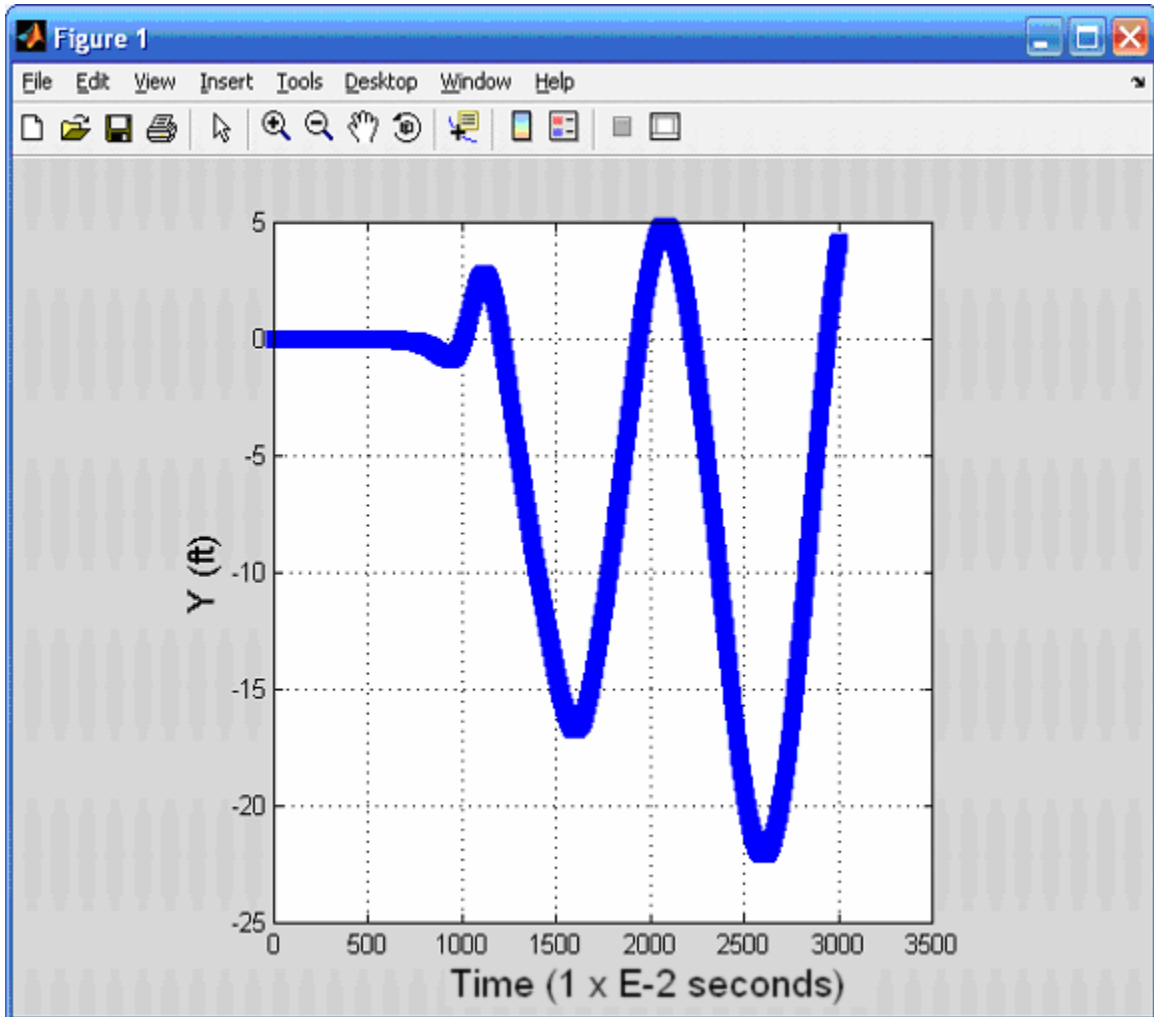


Figure 6.20: Ascending spiral, y vs. Time Plot with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

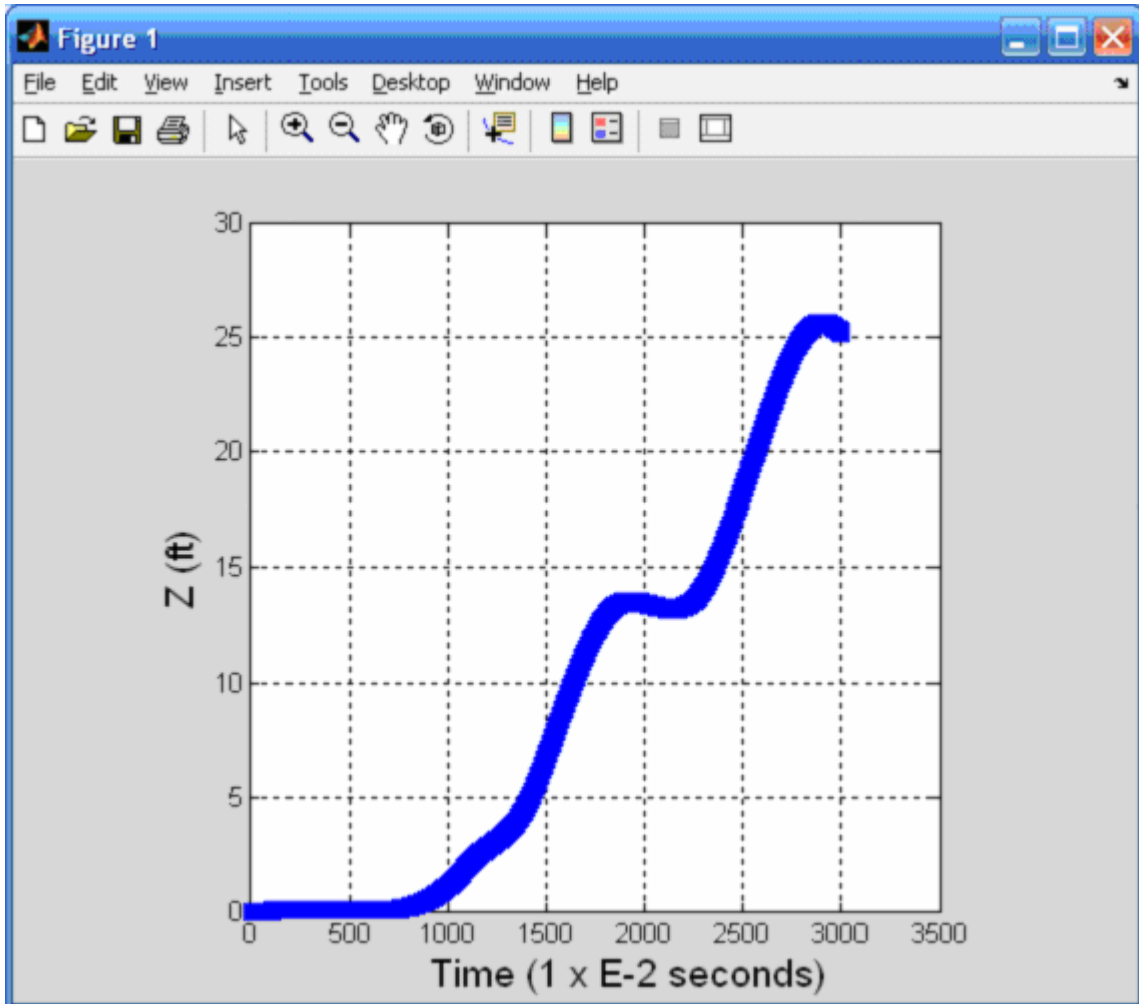


Figure 6.21: Ascending Spiral, z vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

The spiral up maneuver flight path is compared with the set-points, a PID Velocity Tracking controller, a PID Position Tracking controller, a MPC Position Tracking controller, a GA/SA derived equation controller path, and the fuzzy logic controller's path. Figure 6.22 shows the flight paths of all the controllers running within MATLAB. Note that all controllers seem to have difficulty following the set-points exactly even though the other controllers are flying in closed-loop.

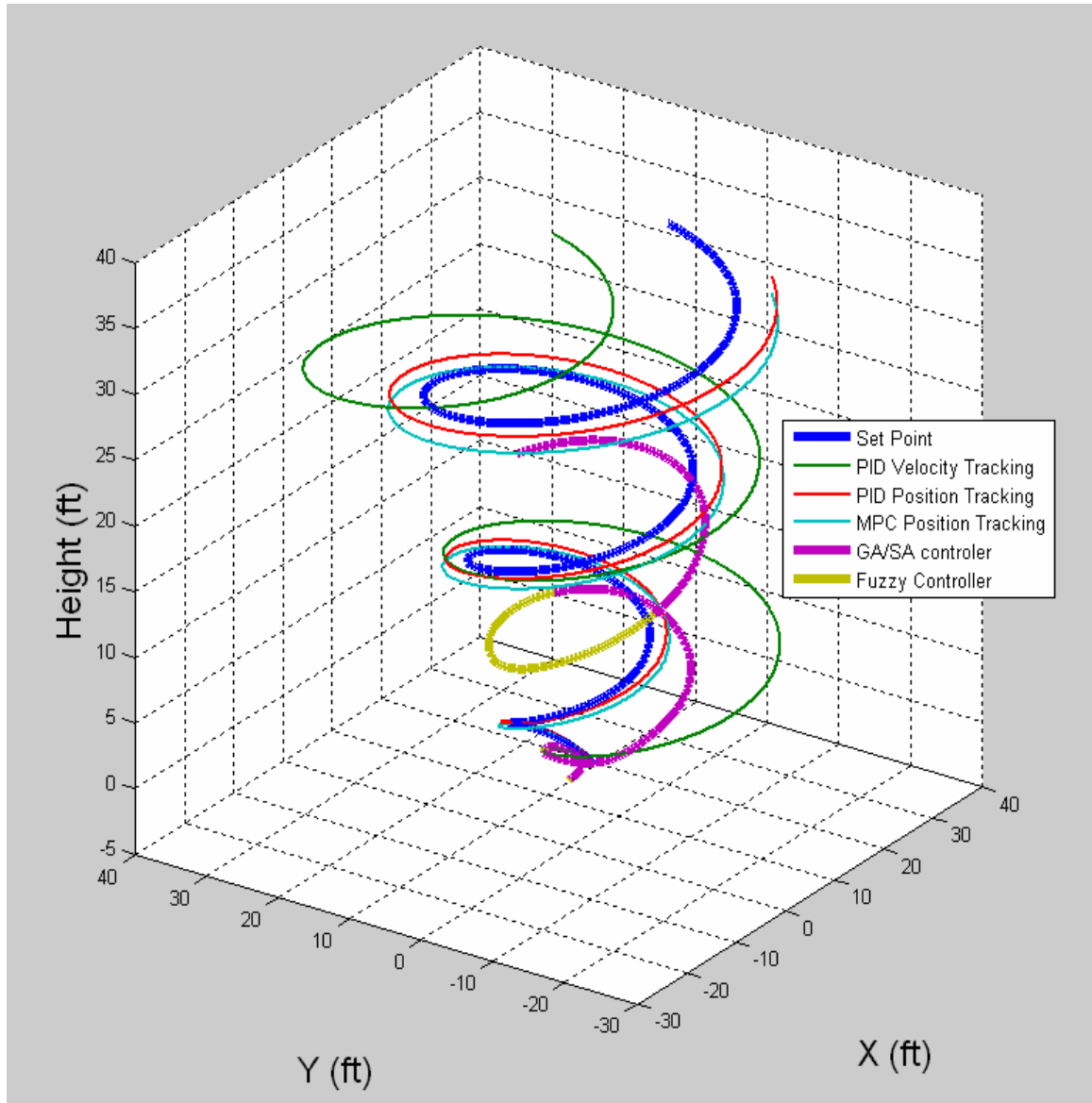


Figure 6.22: Ascending Spiral Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB.

6.2.7 Variable Height Figure-8 Maneuver

This maneuver requires the VTOL to make a variable height figure-8 in flight. Figure 6.23 shows the open-loop/closed-loop paths. In this maneuver, the open-loop and closed-loop flight paths were very different. This can be seen in figure 6.23, the open-loop path was not able to complete the whole maneuver. This is due to open-loop mode where the feedback circuits have been disconnected. One of the challenges of open-loop

mode is that not every maneuver can be successfully copied. The latter depends on the maneuver and on the sensitivity of the particular model to variations encountered when running in open-loop mode.

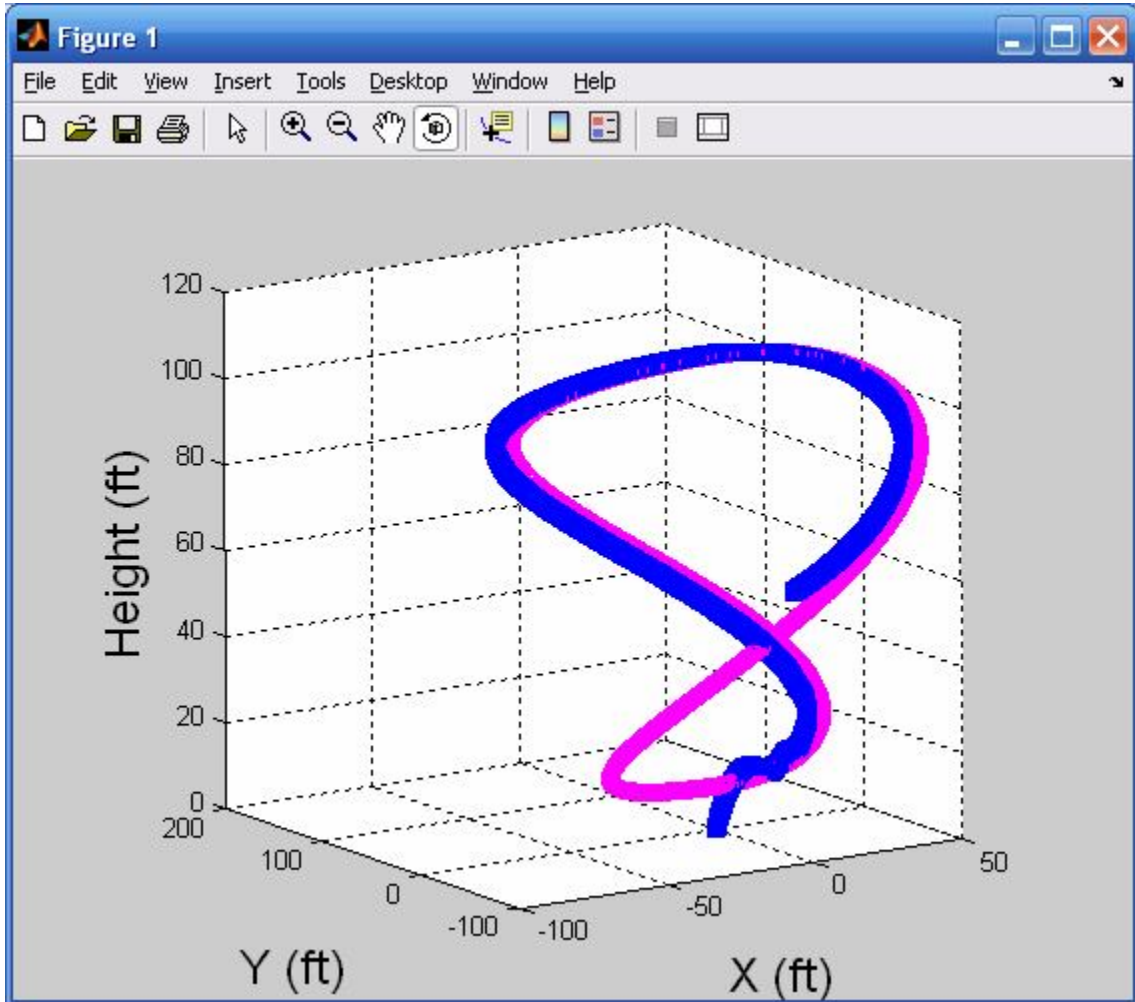


Figure 6.23: Figure-8 Variable Height (in flight loop) Closed Mode and Open Mode; Showing x, y, z vs. Time Plot, with Blue being the Fuzzy-logic Controller (open-mode) Path and Pink being the Fuzzy-logic Controller (closed-mode) Path.

6.2.8 Variable Height Figure-8 Maneuver Testing Results

The comparison of the open-mode path and the GA/SA derived equations path is shown in figure (6.24). The same results are shown one axis at a time in figures (6.25), (6.26), and (6.27) respectively. Again, note that the VTOL model seems to be particularly sensitive to this flight maneuver. Even with high mapping accuracy, the flight path deviates at the end of the run.

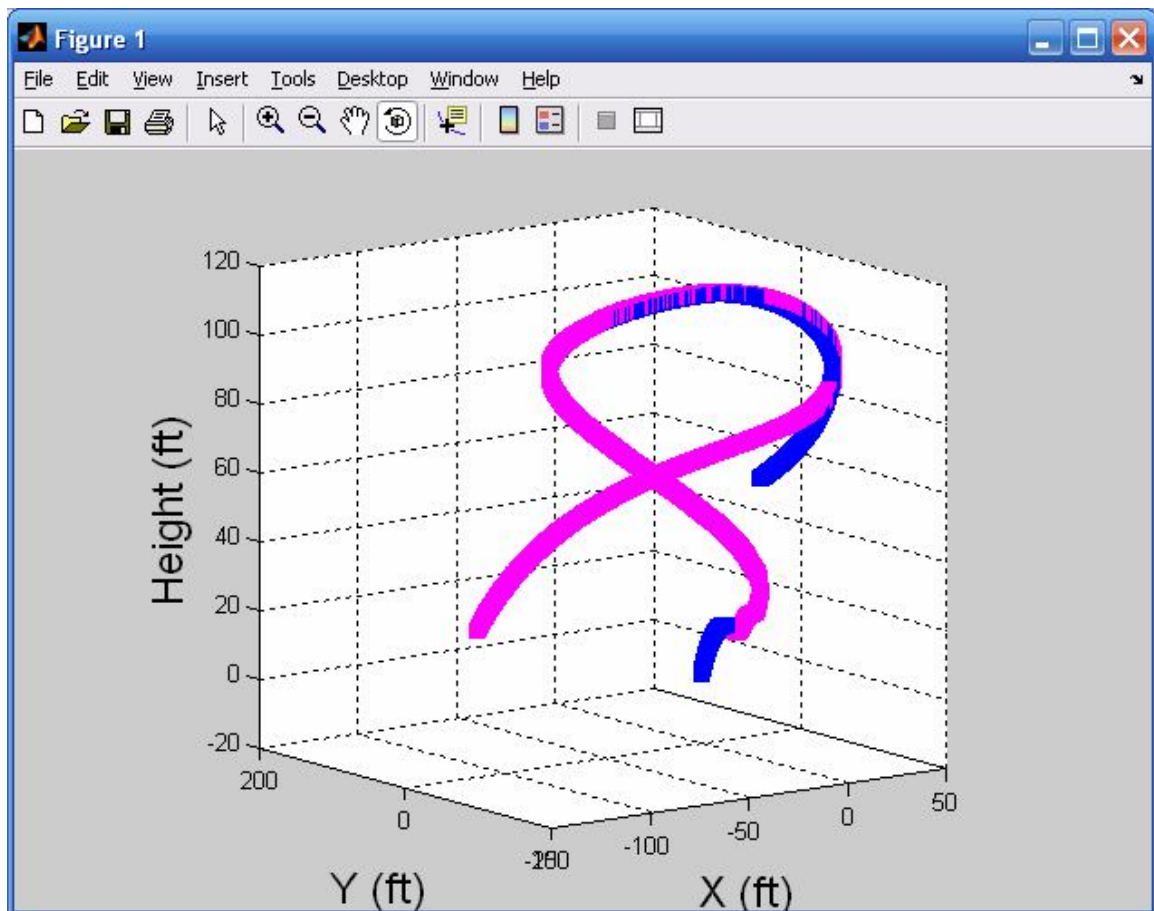


Figure 6.24: Figure-8 Variable Height (in flight loop), GA/SA Path and Fuzzy Logic Controller Path; with Blue being the Fuzzy-logic Controller Path (open-loop) and Pink being the Path Generated by the Derived Equations.

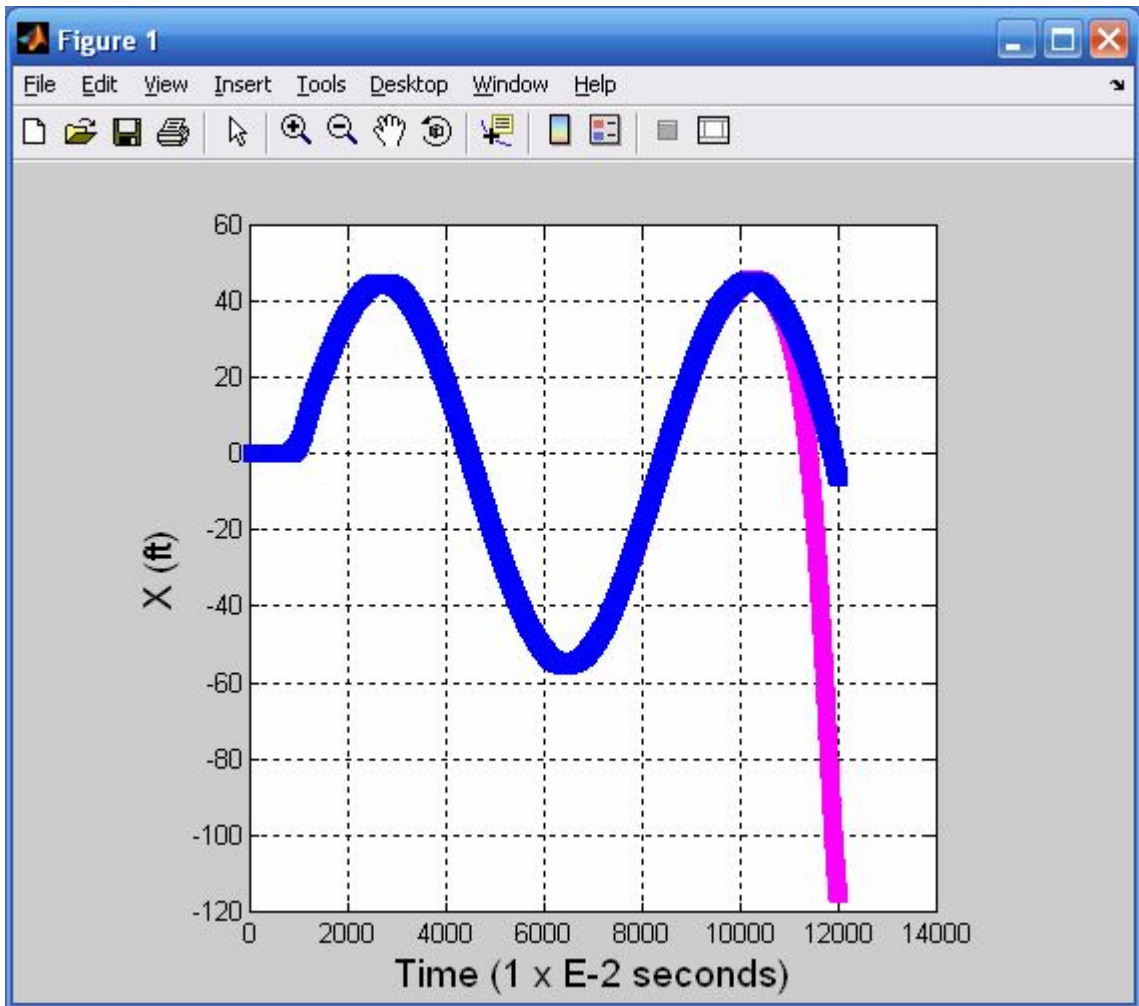


Figure 6.25: Figure-8 Variable Height, x vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

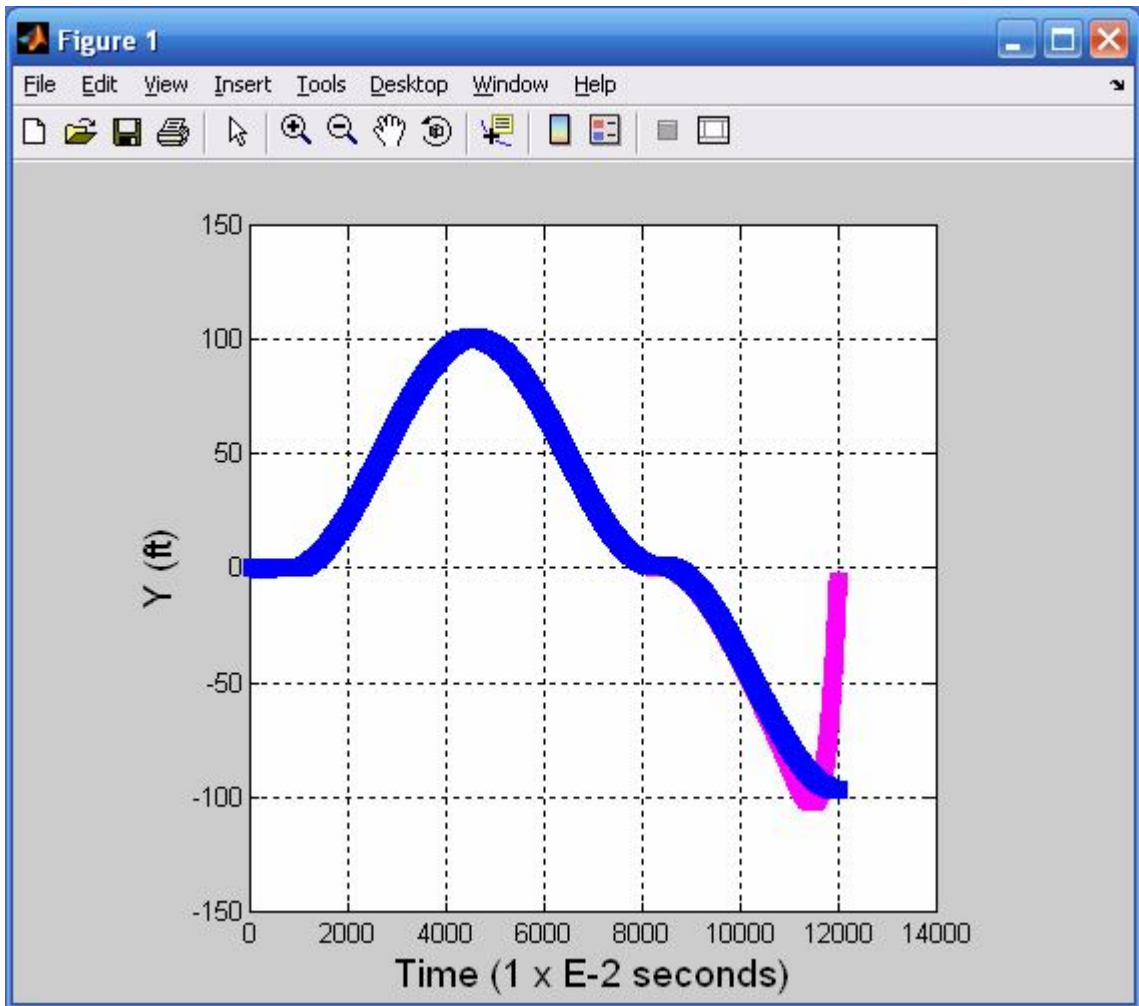


Figure 6.26: Figure-8 Variable Height, y vs. Time Plot; with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

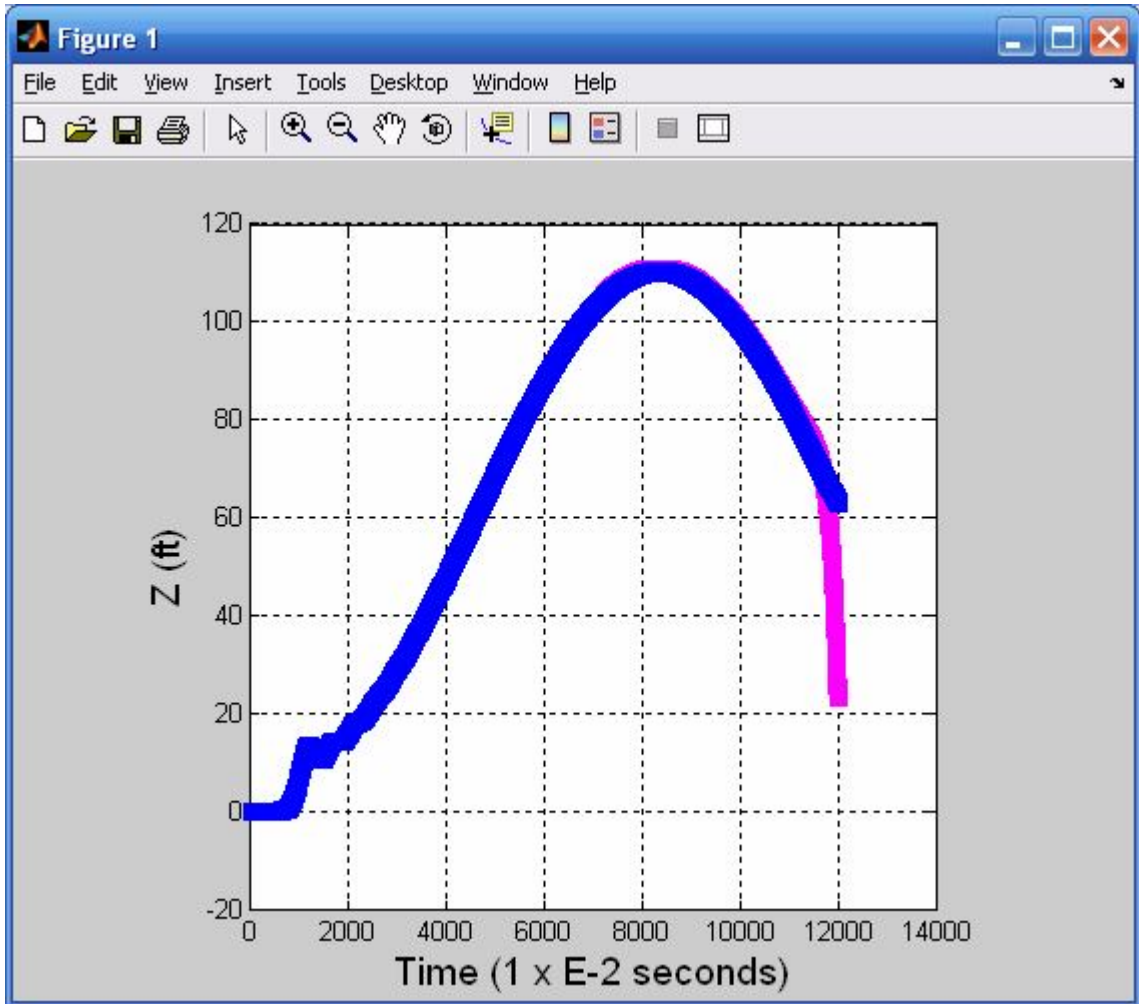


Figure 6.27: Figure-8 Variable Height, z vs. Time Plot with Blue being the Fuzzy-logic Controller Path and Pink being the Path Generated by the Derived Equations.

The variable height figure-8 flight path is compared with the set points, a PID Velocity Tracking controller, a PID Position Tracking controller, a MPC Position Tracking controller, the GA/SA equation controller path, and the fuzzy logic controller path. Figure 6.28 shows the flight paths of all the controllers running within MATLAB.

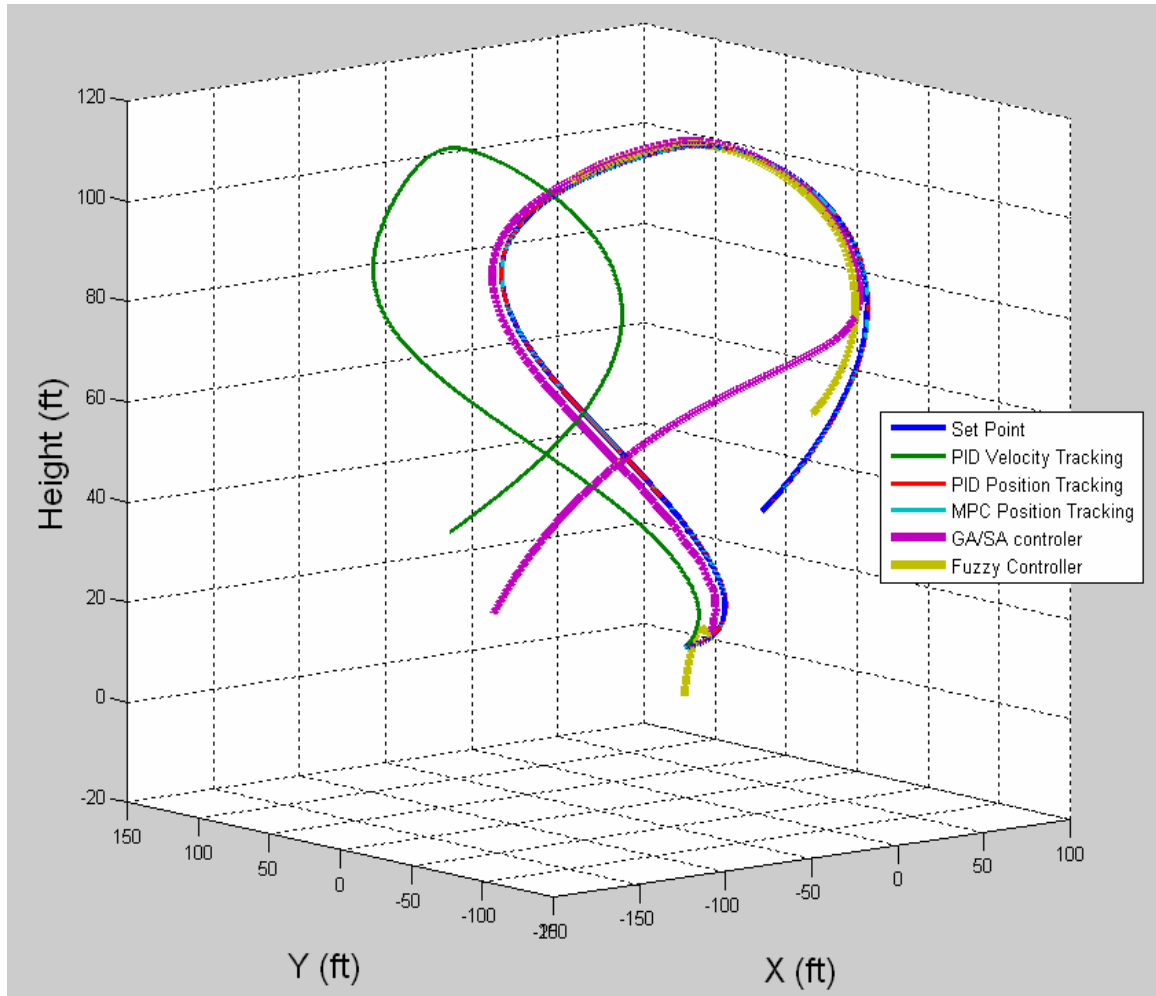


Figure 6.28: Figure-8 Variable Height Showing the GA/SA Derived Controller Compared to other Controllers Flying the Same Model/Set-points in MATLAB.

6.3 Sample Output of GA/SA Algorithm

Figure 6.29 shows a sample output of a GA/SA algorithm as it generates control equations. The first column shows the elapsed time in seconds, the second column shows the actual signal $f(x)$ from the fuzzy logic controller and the third column shows the generated signal $f'(x)$. The last column shows the error. The error in some instances is zero. The full results tables will be shown in the appendix.

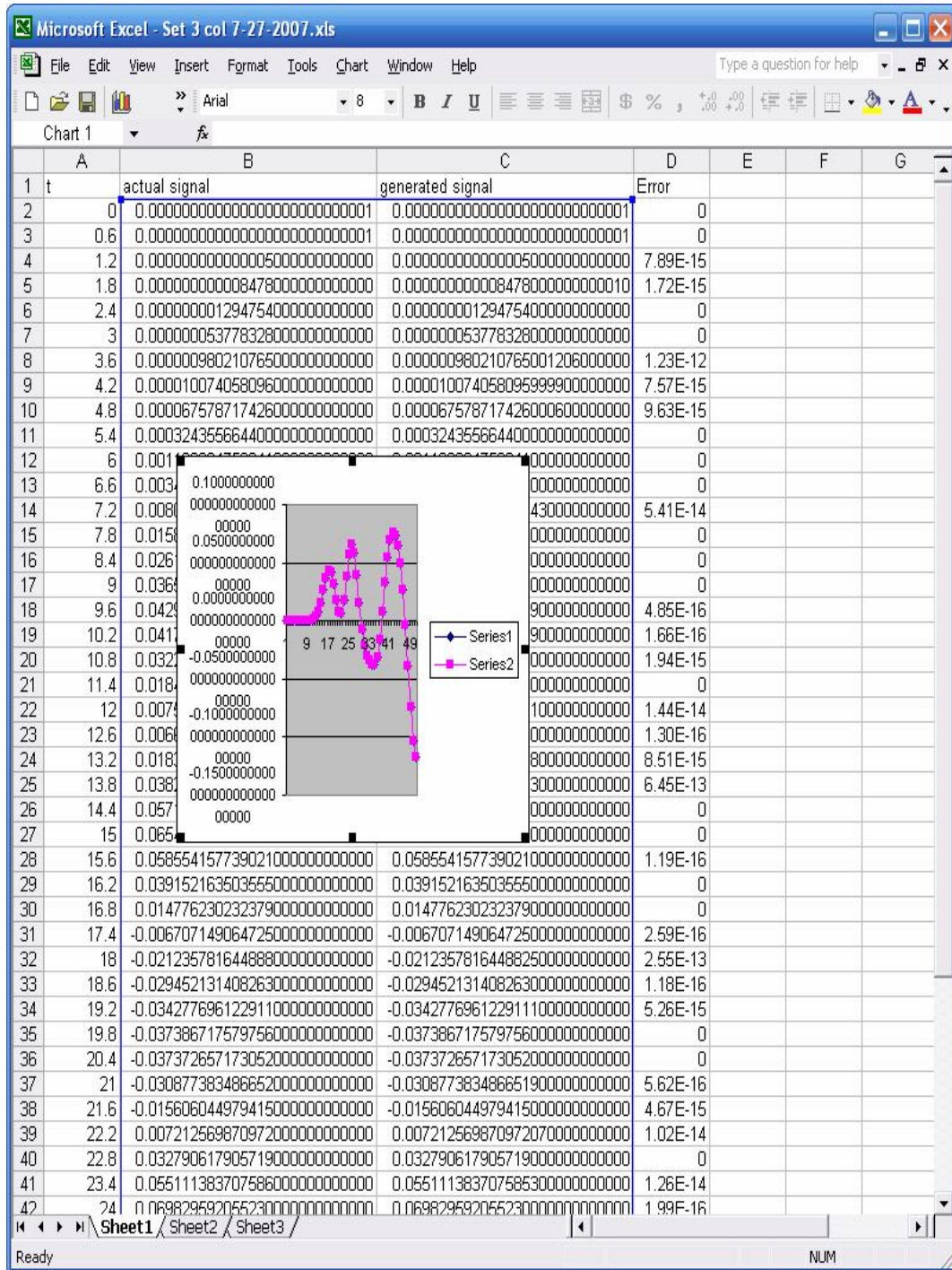


Figure 6.29: Ascending Spiral Sample Result Showing Fit. The Sample Data Shows How Well the Signal is Being Duplicated, Column B is the Actual Signal while Column C is the Algorithm Generated Signal, and Column D Shows the Error.

6.4 Summary of Flight Path Error

This section summarizes flight path errors using graphs that show the deviation (in feet) of the flight path of each controller from the set-points. In addition, the average distance of each flight path from the set points for each maneuver is presented graphically and in a table. Of course, the set-points represent the intended path and each controller followed the intended path as best as it could. The distance from the set-points was calculated using a “Three-dimensional distance” formula or Euclidian Distance [102]. The formula defines distance between any two points in three dimensions as follows:

Let **A** be some point in 3D space whose coordinates are: (x_1, y_1, z_1)

Let **B** be some point in 3D space whose coordinates are: (x_2, y_2, z_2)

The distance between the points **A** and **B** is then calculated as follows:

$$\text{Euclidian Distance} = \text{SQRT}((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2)$$

Where **SQRT** is the square root function.

The path error is shown for each of the following controllers:

- PID Velocity Tracking controller
- PID Position Tracking controller
- MPC Position Tracking controller
- GA/SA derived equation controller
- Fuzzy logic controller

Each maneuver will have 3 associated graphs, the first will show the three-dimensional (3D) distance between the controllers and the set-points. The second graph will show the 3D distance between the fuzzy logic controller and the GA/SA derived control equations as they execute the same maneuver. The third graph is a bar graph that

shows the average 3D distance between each controller and the set-points. The third graph will also show the average distance between the fuzzy logic controller and the GA/SA derived control equations. A table shows summarizes the last graph. The graphs are organized per maneuver.

6.4.1 Figure-8 Flight Path Error Summary

The next three graphs and associated table summarize how well the Figure-8 maneuver was executed by each of the controllers by comparing the 3D distance between each controller and the set-points.

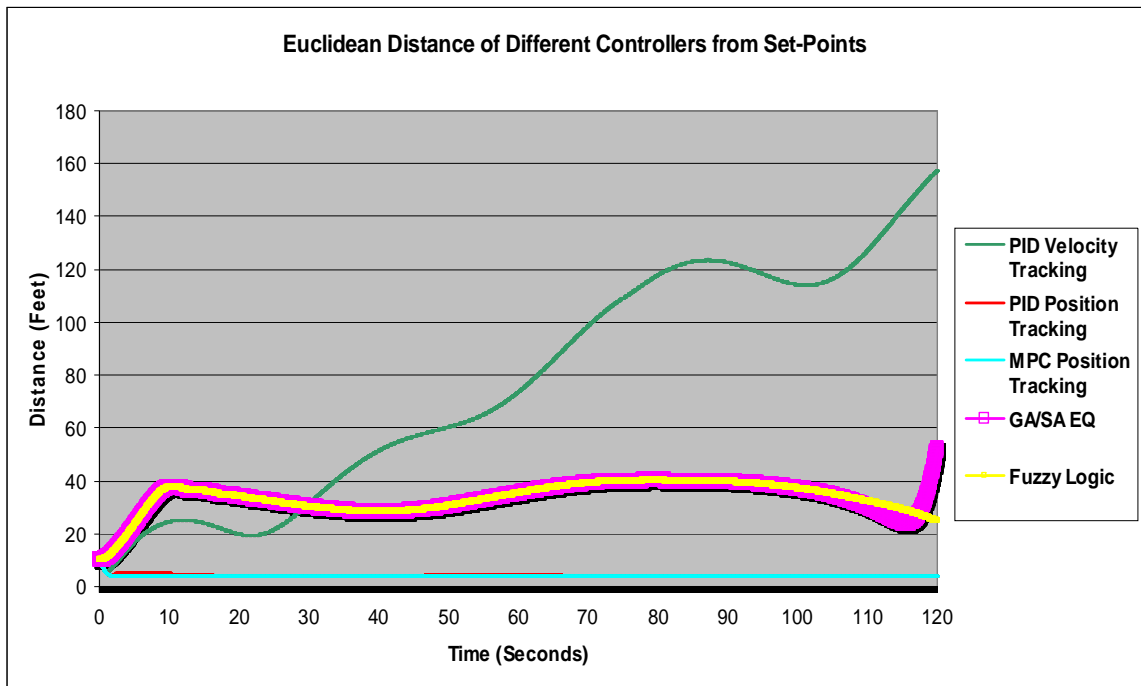


Figure 6.30: Figure-8 Maneuver Showing the GA/SA Derived Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points.

Figure 6.30 shows how well each of the controllers performed in following the set-points. Note that the GA/SA equation based controller follows the fuzzy logic controller's flight path reasonably well and even when it deviates towards the middle of the maneuver, it still manages to shadow the fuzzy logic controller.

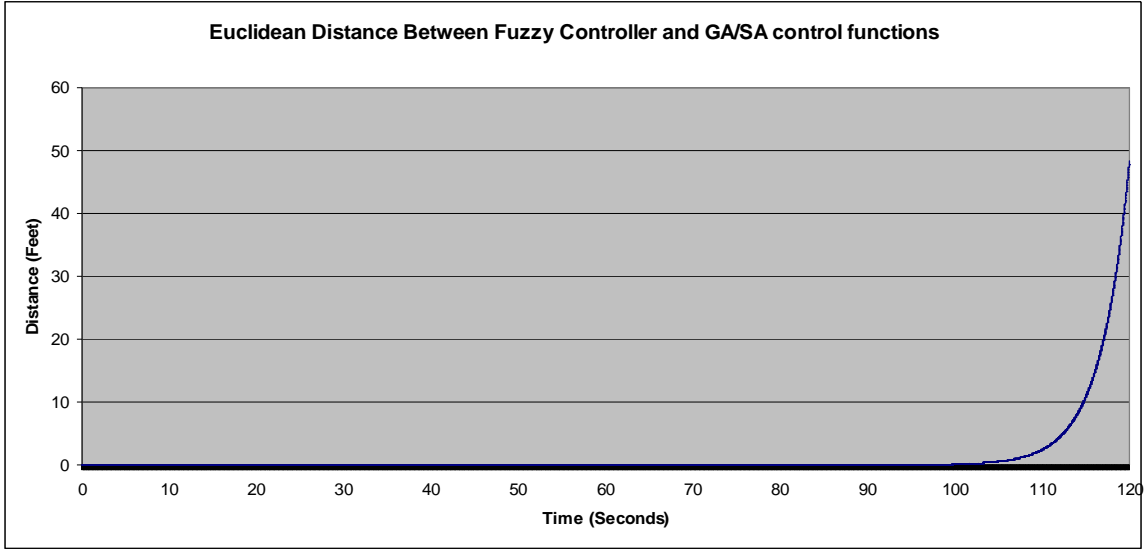


Figure 6.31: Figure-8 Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path from that of the Fuzzy Logic Controller Path (Baseline).

Note that in figure 6.31 the GA/SA derived equation controller does not keep up with the path of the fuzzy controller and deviates from that path by as much as 150 feet towards the end.

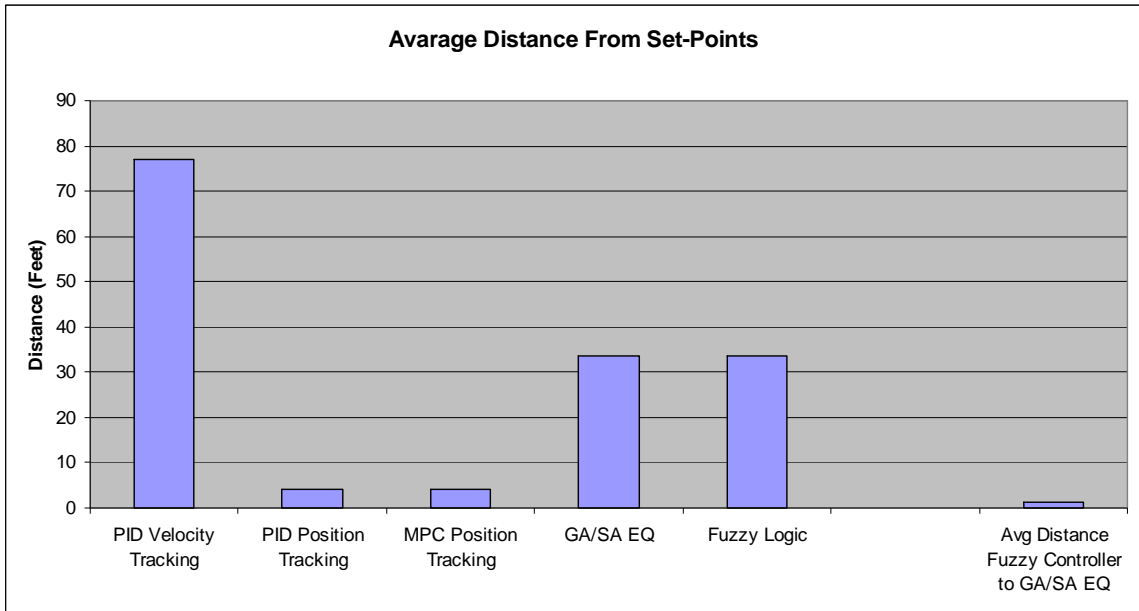


Figure 6.32: Figure-8 Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from The Set-points.

In figure 6.32 all controllers are shown along with their average Euclidian distance from the set-points for the whole maneuver. Note that the average Euclidian distance between the GA/SA derived control equations and the fuzzy logic controller is shown at the far right of the graph. Table 6.2 below shows the average Euclidian distance for each controller over the whole maneuver.

**Table 6.2: Summary of Figure-8 Path Errors in Relation to Set-points.
(All figures are in feet).**

PID Velocity Tracking	PID Position Tracking	MPC Position Tracking	GA/SA EQ	Fuzzy Logic	Avg Distance Fuzzy Controller to GA/SA EQ
76.91881226	4.27307598	4.226641054	33.43756862	33.47859315	1.353576765

6.4.2 U-Turn Flight Path Error Summary

The next three graphs and associated table summarize how well the U-Turn maneuver was executed by each of the controllers by comparing the 3D distance between each controller and the set-points.

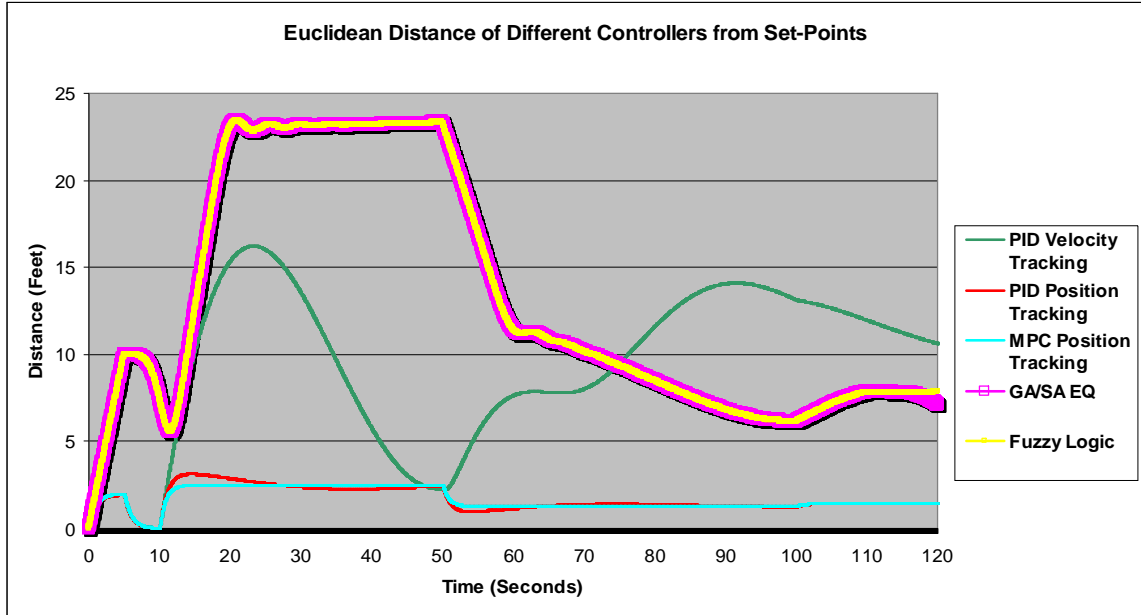


Figure 6.33: U-Turn Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points.

Figure 6.33 shows how well each of the controllers performed in following the set-points. Note that the equation based controller and the fuzzy logic controller's flight paths are very close almost to the end of the maneuver.

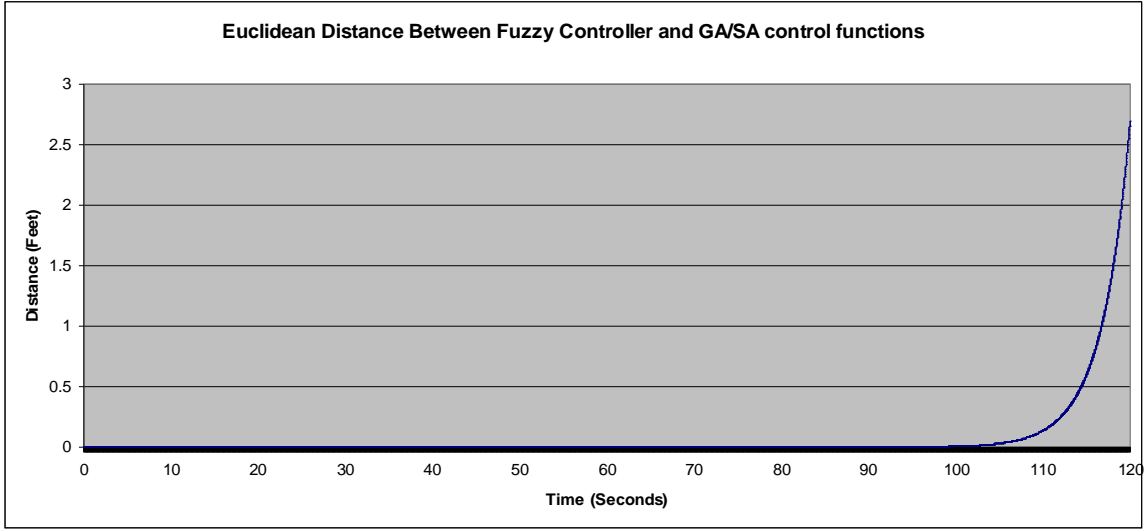


Figure 6.34: U-Turn Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path from That of the Fuzzy Logic Controller Path (Baseline).

In figure 6.34 the GA/SA based control equations manage to follow the path of the fuzzy logic controller very well by showing zero error almost to the end. Note the deviation of approximately 12 feet at the end.

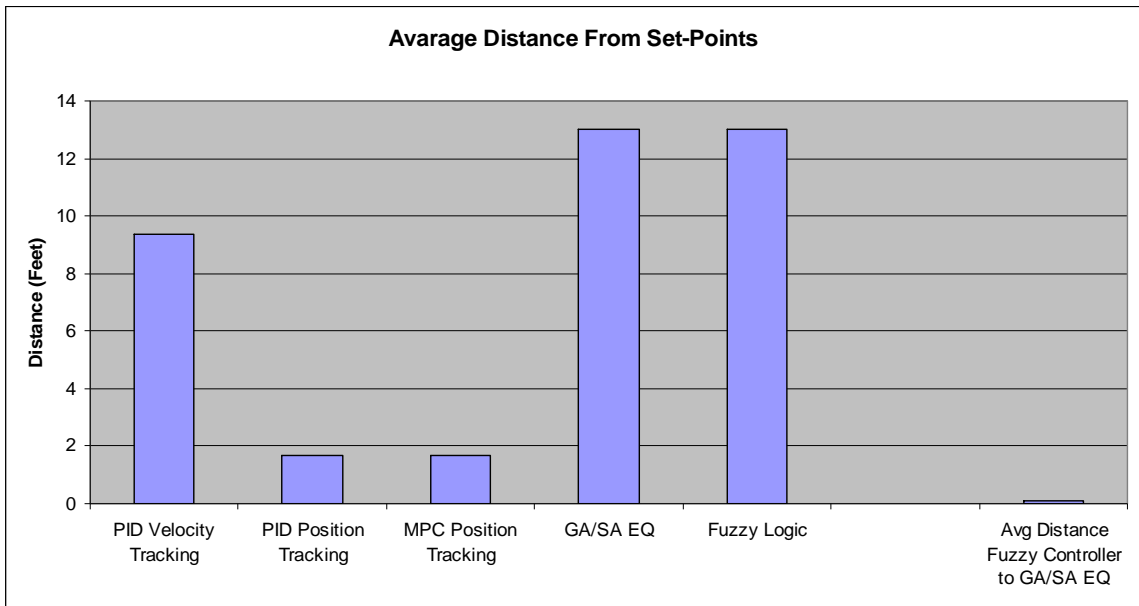


Figure 6.35: U-Turn Maneuver Showing the Average Euclidian Distance between the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-points.

In figure 6.35 all controllers are shown along with their average Euclidian distance from the set-points for the whole maneuver. Note that the average Euclidian distance between the GA/SA derived control equations and the fuzzy logic controller is shown at the far right of the graph. For this maneuver, this distance is minimal which indicates that the control equation based controller was able to duplicate the path of the fuzzy logic controller very well. Table 6.3 below shows the average Euclidian distance for each controller over the whole maneuver.

Table 6.3: Summary of U-Turn Path Errors in Relation to Set-points.
(All figures are in feet).

PID Velocity Tracking	PID Position Tracking	MPC Position Tracking	GA/SA EQ	Fuzzy Logic	Avg Distance Fuzzy Controller to GA/SA EQ
9.358538	1.679473	1.667067	13.00326	13.00936	0.075266856

6.4.3 Spiral-Up Flight Path Error Summary

The next three graphs and associated table summarize how well the Spiral-Up (or Spiral) maneuver was executed by each of the controllers by comparing the 3D distance between each controller and the set-points.

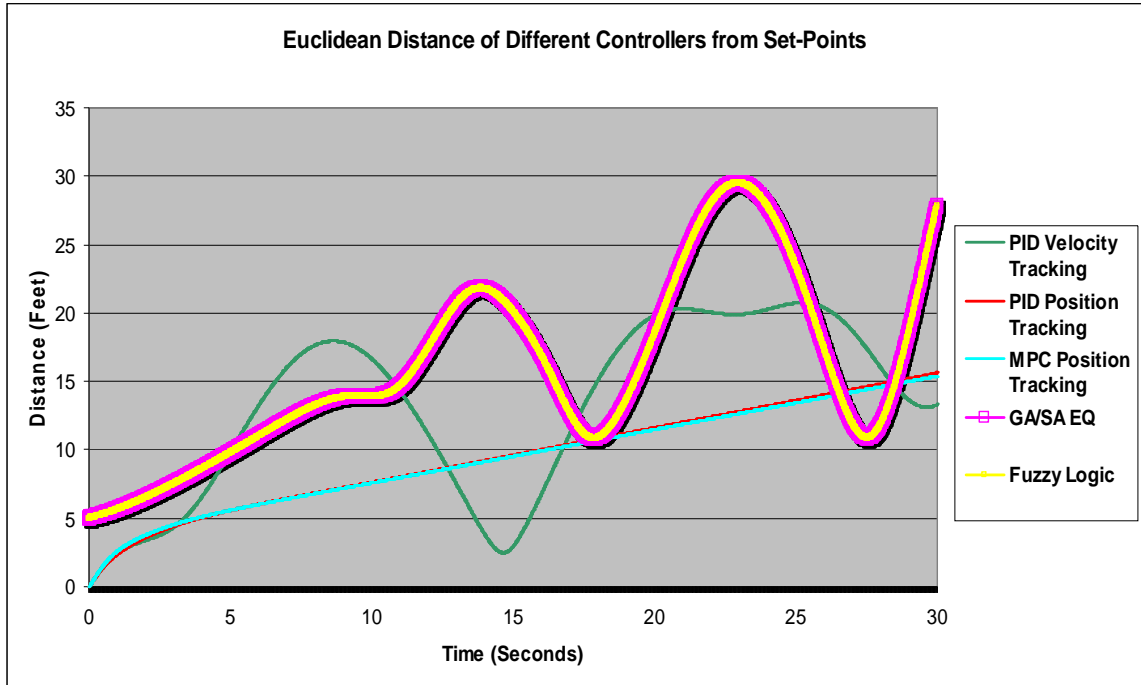


Figure 6.36: Spiral Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from The Set-Points.

Figure 6.36 shows how well each of the controllers performed in following the set-points. Note that the equation based controller and the fuzzy logic controller flight paths are indistinguishable from each other.

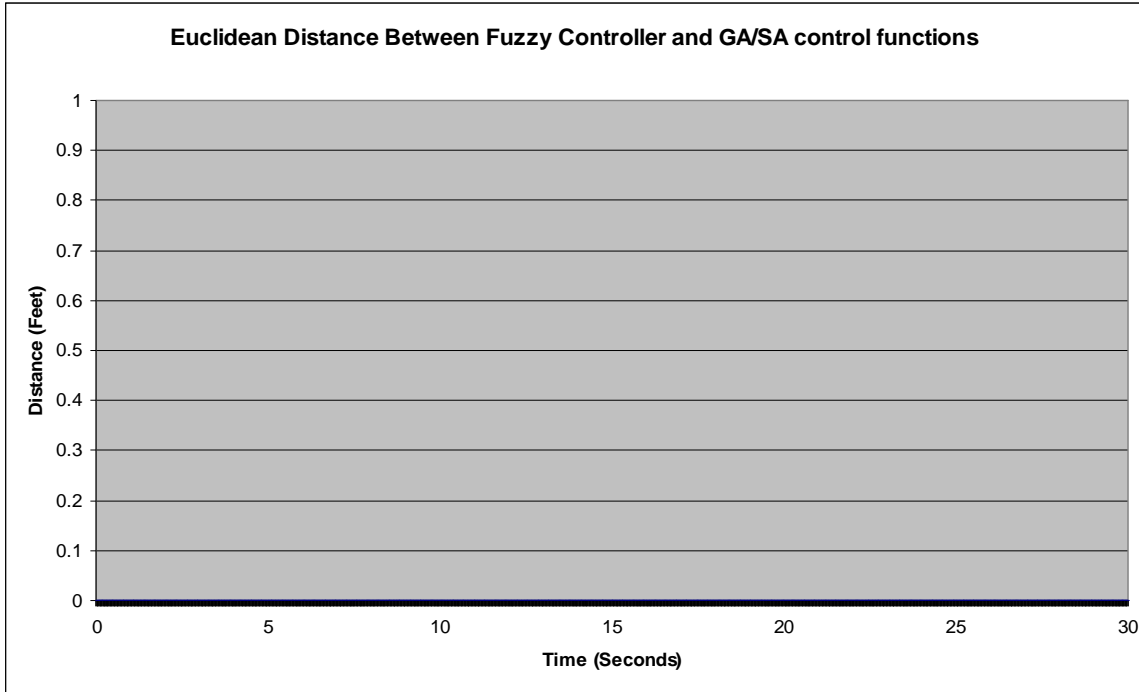


Figure 6.37: Spiral Maneuver Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path from That of the Fuzzy Logic Controller Path (Baseline).

In figure 6.37 the GA/SA based control equations manage to follow the path of the fuzzy logic controller very well by showing zero error all the way to the end. Note that there is no significant deviation throughout the maneuver.

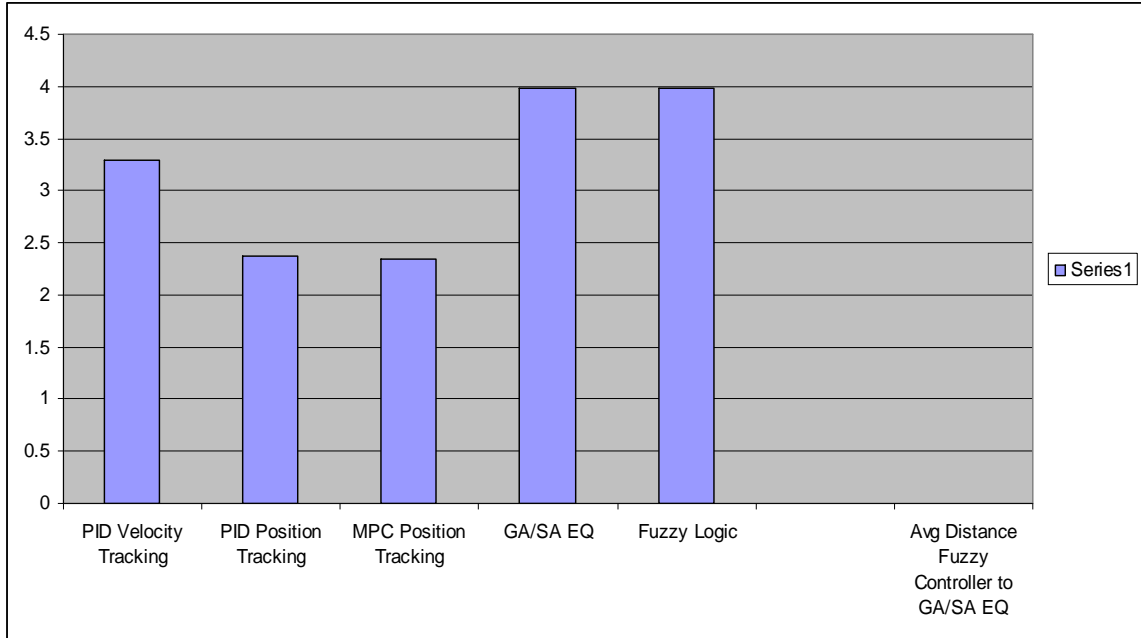


Figure 6.38: Ascending Spiral Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance From the Set-points.

In figure 6.38 all controllers are shown along with their average Euclidian distance from the set-points for the whole maneuver. Note that the average Euclidian distance between the GA/SA derived control equations and the fuzzy logic controller is shown at the far right of the graph. For this maneuver, this distance is 0 and does not show up on the graph which indicates that the control equation based controller was able to duplicate the path of the fuzzy logic controller almost exactly. Table 6.4 below shows the average Euclidian distance for each controller over the whole maneuver.

Table 6.4: Summary of Ascending Spiral Path Errors in Relation to Set-points. (All figures are in feet).

PID Velocity Tracking	PID Position Tracking	MPC Position Tracking	GA/SA EQ	Fuzzy Logic	Avg Distance Fuzzy Controller to GA/SA EQ
3.296512025	2.366273992	2.349485109	3.978934146	3.978934146	0

6.4.4 Variable Figure-8 Flight Path Error Summary

The next three graphs and associated table summarize how well the Variable Figure-8 maneuver was executed by each of the controllers by comparing the 3D distance between each controller and the set-points.

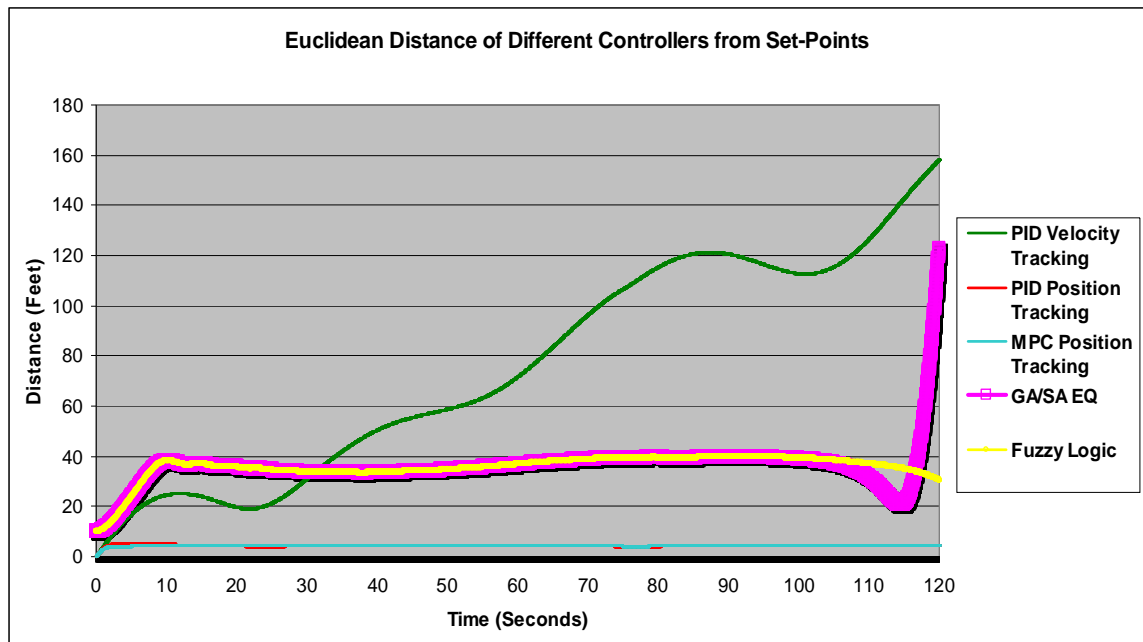


Figure 6.39: Figure-8 Variable Height Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Euclidean Distance from the Set-points.

Figure 6.39 shows how well each of the controllers performed in following the set-points. Note that the GA/SA equation based controller follows the fuzzy logic controller's flight path very well almost till the end. However, there is a significant deviation at the end of the maneuver.

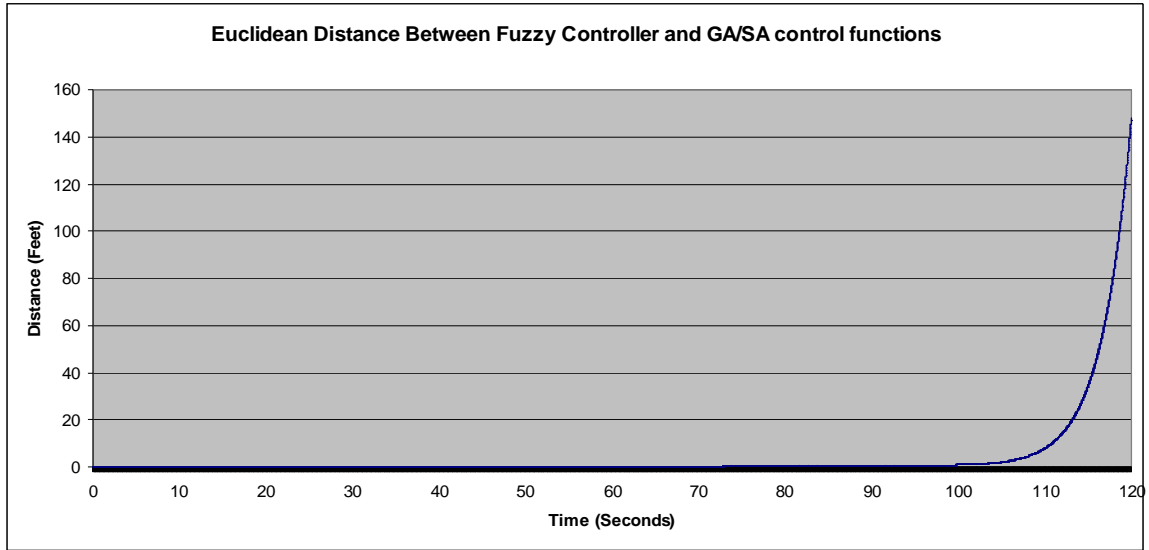


Figure 6.40: Figure-8 Variable Height Showing the Euclidian Distance of the GA/SA Derived Equation Controller Path from that of the Fuzzy Logic Controller Path (Baseline).

In figure 6.40 the GA/SA based control equations manage to follow the path of the fuzzy logic controller very well by showing practically no error almost to the end. However, there is a sizable deviation of approximately 145 feet at the end.

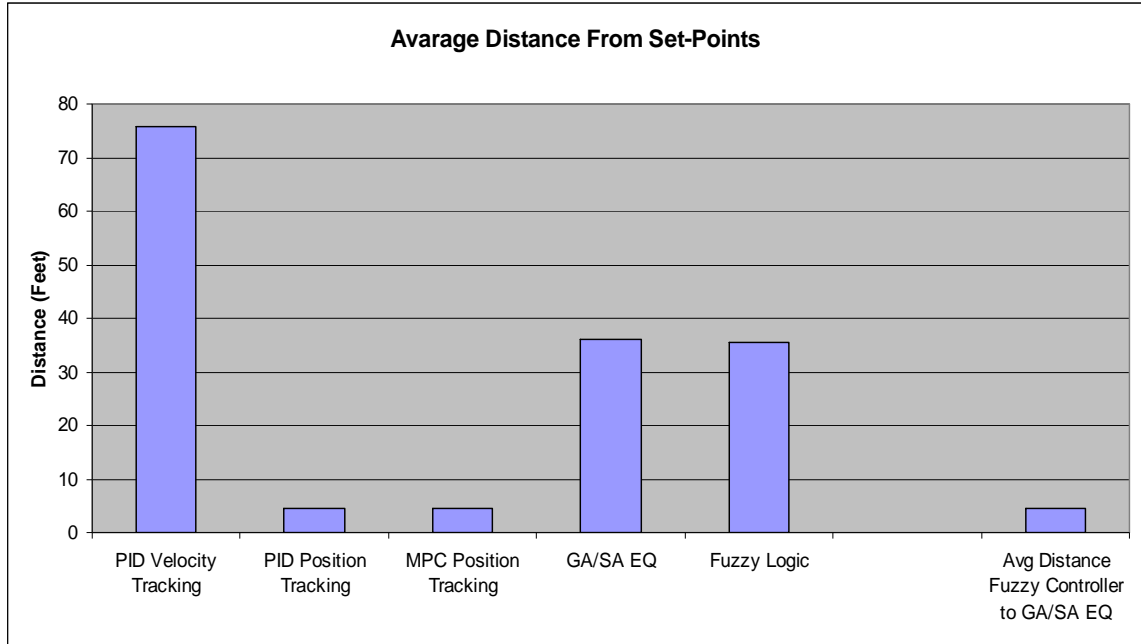


Figure 6.41: Figure-8 Variable Height Maneuver Showing the GA/SA Based Equation Controller and Other Controllers Along With Their Respective Average Euclidean Distance from the Set-points.

In figure 6.41 all controllers are shown along with their average Euclidian distance from the set-points for the whole maneuver. Note that the average Euclidian distance between the GA/SA derived control equations and the fuzzy logic controller is shown at the far right of the graph. For this maneuver, the average distance is small which indicates that the control equation based controller was able to duplicate the path of the fuzzy logic controller reasonably well. Table 6.5 below shows the average Euclidian distance for each controller over the whole maneuver.

Table 6.5: Summary of Figure-8 Variable Height Path Errors in Relation to Set-points. (All figures are in feet).

PID Velocity Tracking	PID Position Tracking	MPC Position Tracking	GA/SA EQ	Fuzzy Logic	Avg Distance Fuzzy Controller to GA/SA EQ
75.69660426	4.4699805	4.43452685	35.989377	35.6338	4.42643287

Table 6.6 summarizes the 3D error for each controller and flight maneuver. For the Figure-8 maneuver, the PID Position Tracking Controller had the lowest average error while the open-loop GA/SA equation was highest at almost twice the error rate. However, in the U-Turn maneuver, the PID Velocity tracking was about the same as the MPC Position Tracking Controller. The open-loop GA/SA control equations did about as well as the closed-loop Fuzzy Logic Controller. The last two maneuvers reflect similar results. The GA/SA based equations were able to duplicate the path of the Fuzzy Logic Controller very well. And since the GA/SA based control equations have no idea what the set-points are, the best they can do is to copy the performance of the Fuzzy Logic Controller, which they did very well for 3 out of 4 maneuvers. The GA/SA derived equation controller exhibited the highest mapping error with respect to the fuzzy logic controller on the Figure-8 maneuver.

Table 6.6: Summary of Path Errors in Relation to Set-points, All Figures are in Feet.

Flight Maneuver	PID Velocity Tracking	PID Position Tracking	MPC Position Tracking	GA/SA equation	Fuzzy logic
<i>Figure-8</i>	76.9188122	4.27307598	4.22664105	33.4375686	33.4785931
<i>U-Turn</i>	9.358538	1.679473	1.667067	13.00452	13.00936
<i>Ascending Spiral</i>	3.29651202	2.366273992	2.34948510	3.97893414	3.97893414
<i>Figure-8, variable height</i>	75.6966042	4.4699805	4.43452685	35.989377	35.6338

6.5 Testing Conclusion

The proposed method was successful in generating formulas that were able to duplicate the flight path of the fuzzy logic controller very well for all of the selected maneuvers. The data obtained demonstrated the following: The algorithm generated functions that were able to statically map the data to an accuracy rate of E-12% or better.

The generated formulas were given control of the virtual helicopter in MATLAB, and the formula based controller successfully flew the VTOL following the path of the fuzzy logic controller very well with an average Euclidian error of less than 1% as shown in table 6.7 below.

Table 6.7: Summary of Euclidian Average Error of the GA/SA Controller with Relation to the Fuzzy Logic Controller. (All figures are in feet).

Flight Maneuver	GA/SA equation	Fuzzy logic	Error %
<i>Figure-8</i>	33.4375686	33.4785931	0.122539498
<i>U-Turn</i>	13.00452	13.00936	0.037203982
<i>Ascending Spiral</i>	3.97893414	3.97893414	0
<i>Figure-8, variable height</i>	35.989377	35.6338	0.997864387

6.6 Conclusion of Research

We have demonstrated a new equation-based VTOL control module that is generated by genetic or simulated annealing based search methods. The generated functions are able to fly a VTOL autonomously using flight time as the sole input. Hence, the control equations are able to fly the VTOL without requiring way-points or positioning data as other controllers do. The GA/SA search algorithm can be used to construct mathematical flight control formulas using datasets collected from a system under observation.

The derived functions model the actions of a source controller be it a human pilot, a PID controller, a fuzzy logic controller, or other automated pilot as they fly a virtual helicopter. The flight paths are non-aggressive maneuvers that include a figure-8, u-turn, ascending spiral, and variable height figure-8. The experimental results demonstrate that this methodology is capable of generating formulas that achieve an accuracy level of

about E-12% or better. This paper also presented a unique method for constructing and generating control formulas.

Since the proposed system does not need set-points to re-create a flight path, it has an advantage over other flight controllers that need this information to successfully complete a mission. Hence, this system may be used as a backup controller should the main flight controller become inoperable due to GPS failure or some other system error. This added layer of redundancy is useful and important especially with the current trend of aviation moving towards more and more unmanned aircraft. The number and price tag of UAVs continues to rise as they are used increasingly in civilian and military applications. Hence, a backup controller that is able to fly the VTOL home in open-loop mode presents an economic and strategic advantage by increasing the chances of recovering aircraft that may be otherwise lost.

Another advantage is the proposed method of using mathematical building blocks to generate formulas. The building blocks may be chosen and adjusted by a researcher which allows for further research and analysis of VTOL control modules to determine which input parameters and flight conditions affect the system. This approach may also have applications in modeling a control module for other autonomous vehicles such as ground rovers and submarines. Applications to planetary rovers may prove to be advantageous as these are especially susceptible to losing communication. The method proposed generates control signal formulas that achieve very good mapping and high accuracy as demonstrated by graphs presented in chapter 5. Since this algorithm can use observed data, control functions can be generated at anytime. Therefore, a VTOL that starts a new mission with new set-points can run this algorithm in the background so that it has a backup controller to take it home should an unexpected condition occur that prevents the main controller from functioning.

6.7 Discussion and Future Work

The method proposed would generate a set of control equations that are capable of flying a VTOL; this method can, potentially, be applied to other vehicle types. Hence, the pilot module is a machine that is capable of learning how to operate other machines without having to know anything about the design or construction of these machines. The proposed algorithm outlines an approach that can be used to develop a machine operator capable of learning maneuvers.

The data shows that the algorithm indeed generates valid functions and that the algorithm performs well. It is important to note that the MATLAB environment did not include wind and temperature factors that may affect the performance of the generated functions. Wind and external factors that affect the stability of the aircraft were assumed to be a part of an external stabilizing module such a gyro-based stabilization system. The latter insures that when there are not inputs to the VTOL it stays stable and does not drift. This should allow the generated functions to run the VTOL as successfully as they did in simulation.

REFERENCES

1. Wikimedia Foundation, Inc. "Unmanned aerial vehicle." [Online] December 2007. <http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle>
2. Wikimedia Foundation, Inc. "VTOL." [Online] December 2007. <<http://en.wikipedia.org/wiki/VTOL>>
3. Wikimedia Foundation, Inc. "Autopilot." [Online] December 2007. <<http://en.wikipedia.org/wiki/Autopilot>>
4. Answers Corporation. "Dead Reckoning." [Online] December 2007. <<http://www.answers.com/topic/dead-reckoning?cat=technology>>
5. Answers Corporation. "GPS." [Online] December 2007. <<http://www.answers.com/GPS+?cat=technology>>
6. Jungkeun Yoon, Mingyan Liu, Brian Noble. "Random Waypoint Considered Harmful Conference on Computer Communications." The 22nd Annual Joint Conference of the IEEE [Computer](#) and [Communications](#) Societies. April 2003. [Online] December 2007. <http://www.ieee-infocom.org/2003/papers/32_03.PDF>
7. Dan Hague, H. T. Kung, Bruce Suter. "Field Experimentation Of Cots-Based Uav Networking." Harvard University. [Online] December 2007. <<http://www.eecs.harvard.edu/~htk/publication/2006-milcom-hague-kung-suter.pdf>>
8. Military Aerospace Technology Online Archives. "Dull, Dirty and Dangerous - Next generation of UAVs hover on the horizon." [Online] December 2007. <<http://www.military-aerospace-technology.com/article.cfm?DocID=152>>
9. C. L. Castillo, W. Moreno, K. P. Valavanis. "Unmanned Helicopter Waypoint Trajectory Tracking Using Model Predictive Control." Processing of the 15th Mediterranean Conference on Control & Automation, July 2007. [Online] December 2007. <<http://med.ee.nd.edu/MED13/papers/T27-018-174.pdf>>

10. GlobalSecurity.org. "Lockheed Martin, Kaman Aerospace Team To Offer Advanced Manned And Unmanned Helicopter Systems Worldwide." [Online] December 2007
<<http://www.globalsecurity.org/military/library/news/2007/03/mil-070308-lockheed-martin01.htm>>
11. Boeing Integrated Defense Systems. "A160 Hummingbird." [Online] December 2007.
<http://www.boeing.com/ids/advanced_systems/hummingbird.html>
12. Bickle, P. Doksum, K. "Mathematical Statistics", 1977.
13. Barnard Microsystems Limited. "The UAV Market." [Online] December 2007.
<http://www.barnardmicrosystems.com/L4E_uav_market.htm>
14. National Defense Industrial Association. "Article: Pentagon Unhappy About Drone Aircraft Reliability." [Online] December 2007.
<http://www.nationaldefensemagazine.org/issues/2003/May/Pentagon_Unhappy.htm>
15. GPS World. "Article: Aerial Advantage: Robust Sensor Fusion for GPS Inertial Measurement Units in Diverse Flight Environments." [Online] December 2007.
<<http://www.gpsworld.com/gpsworld/content/printContentPopup.jsp?id=352808>>
16. Tank Automotive Research, Development, & Engineering Center (TACOM), US Army. Dr. Richard E. McClelland (Director). "TACOM Advanced Planning Briefing for Industry, November 2002." [Online] December 2007.
<<http://www.dtic.mil/ndia/2002apbi/mcclelland.pdf>>
17. Graham R. Drozeski. "Thesis: A Fault-Tolerant Control Architecture for Unmanned Aerial Vehicles, Georgia Institute of Technology, December 2005" [Online] December 2007.
<http://etd.gatech.edu/theses/available/etd-11192005-010343/unrestricted/drozeski_graham_r_200512_phd.pdf>
18. Straight, G. E. (1983), "Masters Thesis: An Open Loop Missile Evasion Algorithm for Fighters." [Online] December 2007.
<<http://stinet.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA136834>>
19. N. Aldawoodi, R. Perez (2003), "Formula Prediction Using Genetic Algorithms", GECCO AI Conference.
20. S. Saripalli, G. S. Sukhatme, J. F. Montgomery, "An Experimental Study of the Autonomous Helicopter Landing Problem", Department of Computer Science, University of Southern California, Los Angeles, California.

21. Q. Wang, T. Aoyama (2001), “A Neural Network Solver for Differential Equations”, Miyazaki University, Japan.
22. Laurence R. Newcome, “Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles.”, American Institute of Aeronautics and Astronautics.
23. University of California, Berkeley. “Article: Ultimate Auto-Pilot by David Pescovitz” [Online] December 2007.
<<http://www.coe.berkeley.edu/labnotes/1003/sengupta.html>>
24. M. Selier, G. Voorsluijs, A.J. de Jong, J.K. Langendoen, C.F. Muller, E.G. IJsselmuiden FlyCam B.V. “FUTURE SURVEILLANCE USING AUTONOMOUS UNMANNED HELICOPTERS, Aerospace Engineering, Delft University of Technology, The Netherlands.” [Online] December 2007.
<<http://administration.ewi.tudelft.nl/live/binaries/fca4cf60-4e90-4110-9ea5-2398d13ce2d9/doc/340-e6aa20.pdf>>
25. Zak Sarris. “SURVEY OF UAV APPLICATIONS IN CIVIL MARKETS (June 2001).” 9th IEEE Mediterranean Conference on Control and Automation (MED 01). June 2001. [Online] December 2007.
<http://med.ee.nd.edu/MED9/Papers/Aerial_vehicles/med01-164.pdf>
- 26 Space News. “Article: Growing Use of UAVs Strains Bandwidth.” [Online] December 2007.
<http://www.space.com/spacenews/archive06/Uav_071706.html>
27. Simon Haykin. “Neural Networks, A Comprehensive Foundation.” Prentice Hall Books. 1999.
28. local positioning systems (LPS) Inc. “Article: Alternative Positioning Technologies.” [Online] December 2007.
<<http://www.indoorlbs.com/id78.html>>
29. Wikimedia Foundation, Inc. “PID Controller.” [Online] December 2007.
<http://en.wikipedia.org/wiki/PID_controller>
30. NASA Aeronautics Research, Timothy H. Cox (NASA), Christopher J. Nagy (NASA), Mark A. Skoog (NASA), Ivan A. Somers (CSM, Inc.) “Civil UAV Capability Assessment, December 2004.” [Online] December 2007.
<http://www.nasa.gov/centers/dryden/pdf/111761main_UAV_Capabilities_Assessment.pdf>
- 31 Yamaha Corp. “Yamaha Autonomous-flight Unmanned Helicopter deployed for observation illegal dumping around Mt. Fuji.” [Online] December 2007.
<<http://www.yamaha-motor.co.jp/global/news/2002/02/06/sky.html>>

32. Extreme GPS. "Startup Rolls Out Wi-Fi Alternative to GPS, June 2005." [Online] December 2007. <<http://www.extremetech.com/article2/0,1697,1830767,00.asp>>
33. Heikki Laitinen (editor), Suvi Ahonen, Sofoklis Kyriazakos, Jaakko Lähteenmäki, Raffaele Menolascino, Seppo Parkkila. "Cellular Location Technology." Information Society Technologies, CELLO Consortium 2001. [Online] December 2007. <<http://www.telecom.ntua.gr/cello/documents/CELLO-WP2-VTT-D03-007-Int.pdf>>
34. Wikimedia Foundation, Inc. "Enhanced 911" [Online] December 2007. <<http://en.wikipedia.org/wiki/E911>>
35. Telematica Instiuit. "Article: PLIM context-aware mobile application framework software." [Online] December 2007. <<http://www.telin.nl/index.cfm?language=en&context=660&id=659>>
36. MoreRFID. "AXCESS and Tyco Fire & Security Integrate Event-Driven RFID Solutions Into Access Control Systems." [Online] December 2007. <http://www.morerfid.com/details.php?subdetail=Report&action=details&report_id=1815&display=RFID>
37. AT&T Laboratories, Cambridge. "The Bat Ultrasonic Location System." [Online] December 2007. <<http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/>>
38. Mark A. Sturza and Farzad Ghazvinian. "White Spaces Engineering Study: CAN COGNITIVE RADIO TECHNOLOGY OPERATING IN THE TV WHITE SPACES COMPLETELY PROTECT LICENSED TV BROADCASTING?" New America Foundation. Working Paper #16 January 2007. [Online] December 2007. <http://www.newamerica.net/files/WorkingPaper16_WhiteSpaceSensing_Sturza.pdf >
39. eXtreme Results International, Inc. "United States Patent 5,510,801 Engelbrecht , et al. April 23, 1996: Location determination system and method using television broadcast signals" [Online] December 2007. <<http://xrint.com/patents/us/5510801>>
40. PatentStorm LLC. "Position location using ghost canceling reference television signals." [Online] <December 2007. <<http://www.patentstorm.us/patents/6522297-description.html>>
41. Whitley, D. Goldberg, D. Cantu, E. Spector, L. Parmee, I. Beyer, H. "GECCO 2000", Proceedings of the Genetic and Evolutionary Computation Conference July 10-12 2002.

42. Davis, L. DeJong , K. Vose, M. Whitley, D. “Evolutionary Algorithms”, Springer 1999.
43. Neuros Co., Ltd. “Mini Turbo Jet/Shaft Engine.” [Online] December 2007.
<http://www.neuros.com/04gov/gov_01.htm>
44. Back, T. “Evolutionary Algorithms in Theory and Practice”, Oxford University Press 1996.
45. Michael, D. “The Simple Genetic Algorithm”, MIT Press 1999
46. Mitchell, T. “Machine Learning”, WBC/McGraw-Hill, 1997.
47. Quagliarella, D. Periaux, J. Poloni, C. Winter, G. “Genetic Algorithms and Evolution Strategies in Engineering and Computer Science”, Wiley and Sons 1998.
48. Koza, J. Banzhaf, W. Chellapila, K. Deb, K. Dorigo, M. Fogel, D. Garzon, M. Goldberg, D. Iba, H. Riolo, R. "Genetic Programming", Proceedings of the Third Annual Genetic Programming Conference July 22-25 1998.
49. Stender, J. Hillerbrand, E. Kingdon, J. "Genetic Algorithms in Optimization, Simulation and Modeling", IOS Press 1994.
50. Davis, L. “Handbook of Genetic Algorithms”, VNB 1991.
51. Whitley, D. Goldberg, D. Cantu, E. Spector, L. Parmee, I. Beyer, H. "GECCO 2000", Proceedings of the Genetic and Evolutionary Computation Conference July 10-12 2002.
52. L. Davis, (1990), Genetic Algorithms and Simulated Annealing, Pitman, London.
53. P. J. M. van Laarhoven and E. H. L. Aarts, (1987), Simulated Annealing Theory and Applications.
54. N. Aldawoodi, R. Perez, Wendy Alvis, Kimon Valavanis (2004), “Developing Automated Helicopter Models Using Simulated Annealing and Genetic Search”, GECCO AI Conference.
55. N. Aldawoodi, R. Perez (2004), “Advanced Formula Prediction using Simulated Annealing”, GECCO AI Conference.
56. Wikimedia Foundation, Inc. “Helicopter.” [Online] December 2007.
<<http://en.wikipedia.org/wiki/Helicopter>>
57. Wikimedia Foundation, Inc. “Focke-Wulf Fw 61.” [Online] December 2007.
<http://en.wikipedia.org/wiki/Focke-Wulf_Fw_61>

58. Wikimedia Foundation, Inc. "Flettner Fl 282." [Online] December 2007.
<http://en.wikipedia.org/wiki/Flettner_Fl_282>
59. Wikimedia Foundation, Inc. "Vought-Sikorsky 300." [Online] December 2007.
<http://en.wikipedia.org/wiki/Vought-Sikorsky_300>
60. Wikimedia Foundation, Inc. "Sikorsky R-4." [Online] December 2007.
<http://en.wikipedia.org/wiki/Sikorsky_R-4>
61. H. Shim, T. J. Koo, F. Homann (2001), "A Comprehensive Study of Control Design for an Autonomous Helicopter Robotics and Intelligent Machines", University of California at Berkeley.
62. V. Gavrillets, B. Mettler and E. Feron, "Nonlinear Model for a Small-Size Acrobatic Helicopter", AIAA Guidance, Navigation, and Control Conference and Exhibit, August, 2001.
63. P. Spanoudakis, N. C. Tsourveloudis, K. P. Valavanis, "Technical Design Specifications for a Prototype Unmanned VTOL Vehicle", (under review, revised twice).
64. I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, A. N. Kostaras, "3-D Evolutionary Algorithm Based Off-line / On-line Path Planner for UAV Navigation", IEEE Transactions on System, Man and Cybernetics, Part B, December 2003.
65. T. Samad, G. Balas, Editors, Software-Enabled Control, IEEE Press / Wiley Interscience, 2003.
66. "Neuro-Fuzzy Approaches to Anticipatory Control" (with Lefteri H. Tsoukalas and Andreas Ikonomopoulos), Chapter 13, *Artificial Intelligence in Industrial Decision Making, Control, and Automation*, (in press), Athens, Greece, 1994.
67. B. Mettler, "Identification Modeling and Characterization of Miniature Rotorcraft." Kluwer Academic Publishers, 2003
68. Wikimedia Foundation, Inc. "Genetic algorithm" [Online] December 2007.
<http://en.wikipedia.org/wiki/Genetic_algorithm>
69. Reason Magazine. "Article: Learning Curve." [Online] December 2007.
<<http://www.reason.com/news/show/30059.html>>
70. ScienceDaily LLC. "Introduction to genetics." [Online] December 2007.
<http://www.sciencedaily.com/articles/i/introduction_to_genetics.htm>
71. Wikimedia Foundation, Inc. "Introduction to genetics." [Online] December 2007.
<http://en.wikipedia.org/wiki/Introduction_to_genetics>

72. Sushil J. Louis, Gregory J. E. Rawlins. "Predicting Convergence Time for Genetic Algorithms." Department of Computer Science, Indiana University.
<<http://www.cs.indiana.edu/pub/techreports/TR370.pdf>>
73. Wikimedia Foundation, Inc. "Sewall Wright." [Online]
<http://en.wikipedia.org/wiki/Sewall_Wright>
74. Martin, W. Spears, W. "Foundations of Genetic Algorithms 6", Morgan Kaufmann publishers 2000.
75. Roy L. Johnston (Professor of Computational Chemistry) [School of Chemistry](#), University of Birmingham. "Applications of Genetic Algorithms in Chemistry" [Online] December 2007.
<<http://www.tc.bham.ac.uk/~roy/Research/ga.html>>
76. Wikimedia Foundation, Inc. "Dynamic programming" [Online] December 2007.
<http://en.wikipedia.org/wiki/Dynamic_programming>
77. Wikimedia Foundation, Inc. "Simulated annealing" [Online] December 2007.
<http://en.wikipedia.org/wiki/Simulated_annealing>
78. Computational Science Education Project. "Mathematical Optimization, Simulated Annealing." [Online] December 2007.
<<http://www.phy.ornl.gov/csep/CSEP/MO/NODE28A.html>>
79. Wikimedia Foundation, Inc. "Coanda effect." [Online] December 2007.
<http://en.wikipedia.org/wiki/Coanda_effect>
80. GlobalSecurity.org. "HELICOPTER FUNDAMENTALS." [Online] December 2007.
<<http://www.globalsecurity.org/military/library/policy/army/accp/al0966/le3.htm>>
81. Aerospaceweb.org. "Maximum Forward Speed." [Online] December 2007.
<<http://www.aerospaceweb.org/design/helicopter/velocity.shtml>>
82. Gyrodyn Helicopter Historical Foundation. "Gyrodyn Helicopters." [Online] December 2007.
<<http://www.gyrodynhelicopters.com/>>
83. Helicopter History Site. "Helicopters." [Online] December 2007.
<<http://www.helis.com/>>
84. Unmanned Aerial Vehicles (UAVs). "Yamaha RMAX." [Online] December 2007.
<http://www.livingroom.org.au/uavblog/archives/yamaha_rmax.php>

85. Tactical Aerospace Group. "Unmanned Aircraft Systems." [Online] December 2007. <<http://www.tacticalaerospacelgroup.com>>
86. Neural Robotics, Inc. "The AutoCopter." [Online] December 2007. <<http://www.neural-robotics.com>>
87. DefenseReview.com. "Unmanned Mini-Helicopter Gets 'Weaponized'." [Online] December 2007. <<http://www.defensereview.com/article846.html>>
88. CNET Networks, Inc. "Boeing robo-copter lifts heavy load." [Online] December 2007. <http://www.news.com/8300-10784_3-7-0.html?keyword=helicopters>
89. Robot Gossip. "Unmanned Helicopter for Everest Rescues." [Online] December 2007. <<http://robotgossip.blogspot.com/2007/02/unmanned-helicopter-for-everest.html>>
90. Science Applications International Corporation (SAIC). "Vigilante Helicopter Vertical Takeoff and Landing (VTOL) Unmanned Aerial Vehicle (UAV)" . [Online] December 2007. <<http://www.saic.com/products/aviation/vigilante/vig.html>>
91. C4ISR Journal. "Kaman, Lockheed to offer unmanned helicopter." [Online] December 2007. <<http://www.isrjournal.com/story.php?F=2632096>>
92. Bartleby.com. "The American Heritage® Dictionary of the English Language: helicopter." [Online] December 2007. <<http://www.bartleby.com/61/8/H0130800.html>>
93. KAMAN Aerospace. "K-MAX Unmanned." [Online] December 2007. <<http://www.kamanaero.com/helicopters/uav.html>>
94. Greg Goebel / In The Public Domain. "US Battlefield UAVs" [Online] December 2007. <http://www.vectorsite.net/twuav_09.html>
95. Wikimedia Foundation, Inc. "Coriolis effect." [Online] December 2007. <http://en.wikipedia.org/wiki/Coriolis_effect>
96. Wikimedia Foundation, Inc. "Subsumption architecture." [Online] December 2007. <http://en.wikipedia.org/wiki/Subsumption_architecture>

97. National Aeronautics and Space Administration (NASA). "Article: Parameter Identification, CIPHER." [Online] December 2007.
<<http://www.nasa.gov/centers/ames/research/technology-onepaggers/paramid.html>>
98. Robert Gilchrist Huenemann, M.S.E.E. "Article: Bandlimited Interpolation." [Online] December 2007.
<<http://home.flash.net/~bobgh/bandlimited.htm>>
99. University Affiliated Research Center. "Article: Comprehensive Identification from FrEQUENCY Responses." [Online] December 2007.
<<http://uarc.ucsc.edu/flight-control/cifer/index.html>>
100. Wikimedia Foundation, Inc. "Machine learning." [Online] December 2007.
<http://en.wikipedia.org/wiki/Machine_learning>
101. Wikimedia Foundation, Inc. "Function (mathematics)." [Online] December 2007.
<http://en.wikipedia.org/wiki/Function_%28mathematics%29>
102. Wikimedia Foundation, Inc. "Euclidean distance." [Online] December 2007.
<http://en.wikipedia.org/wiki/Euclidean_distance#Three-dimensional_distance>
103. Jukka Kohonen, "A brief comparison of simulated annealing and genetic algorithm approaches." Term paper. December 12, 1999, University of Helsinki [Online] March 2008.
<<http://www.cs.helsinki.fi/u/kohonen/papers/gasa.html>>
104. Luisa Franconi, Christopher Jennison, "Comparison of a genetic algorithm and simulated annealing in an application to statistical image reconstruction", Statistics and Computing, 1997, [Online] March 2008.
<<http://portal.acm.org/citation.cfm?id=599274>>
105. Peter Ross and Dave Corne, "Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems", Evolutionary Computing, Volume 993/1995, [Online] March 2008.
<<http://www.springerlink.com/content/y26875216500p045/>>
106. A. D. KING, B.Sc., F.R.I.N., Marconi Electronic Systems Ltd. "Article: Inertial Navigation – Forty Years of Evolution." GEC REVIEW, VOL. 13, NO. 3, 1998. [Online] December 2007.
<http://www.imar-navigation.de/download/inertial_navigation_introduction.pdf>

APPENDICES

Appendix A: Graphs of Control Signals

Set 1: figure 8 in flight, all 4 control signals are shown below, with the x axis represents time. The time is sampled for a total of 201 samples. The y axis shows the actual value of the control signal. Note that this is a MATLAB internal value and has no unit.

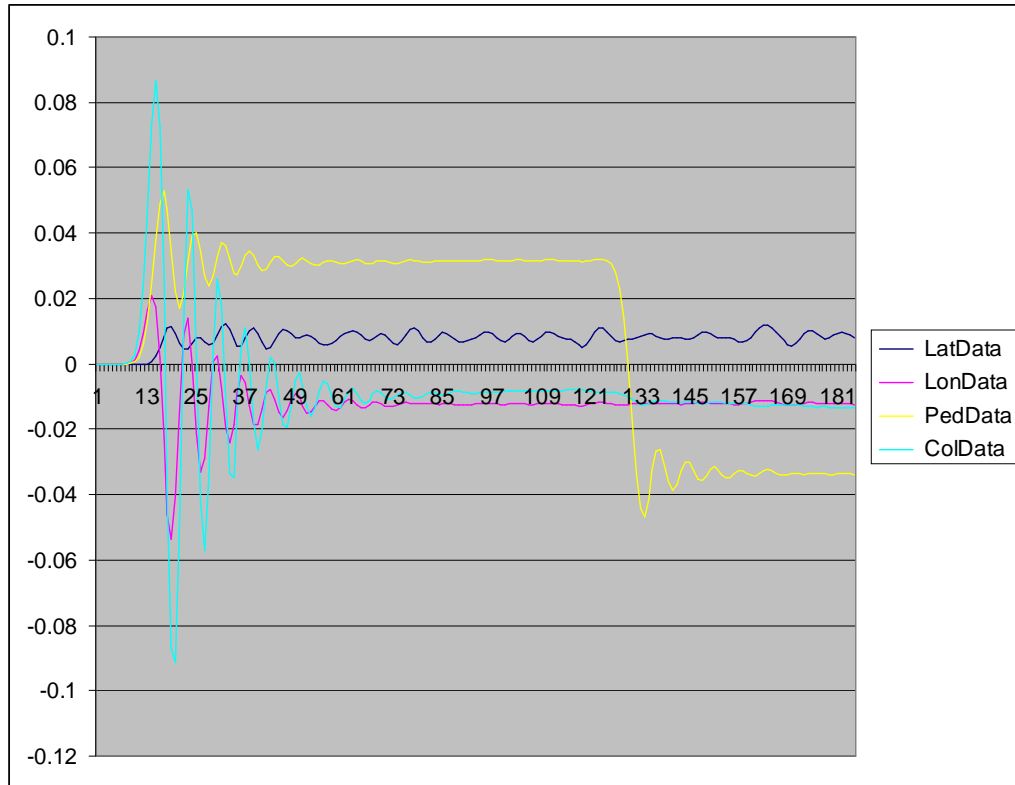


Figure A.1: Figure 8 (In Flight Loop) Showing All 4 Control Signals; Note the y Axis has no Units as These are the Actual Numerical Values Supplied to the VTOL Within the MATLAB Model.

Appendix A: (Continued)

Set 2: U-Turn in flight, all 4 control signals are shown below, with the x axis being the sample number for a total of 201 samples and the y axis showing the actual value.

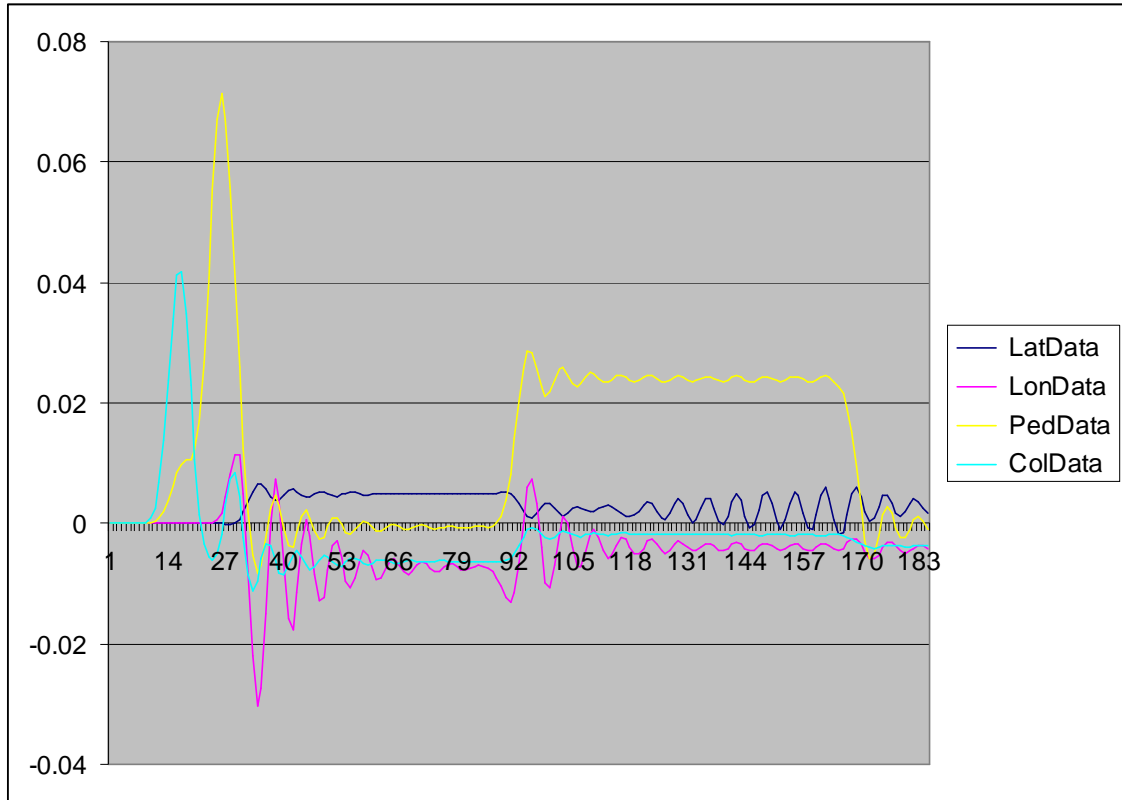


Figure A.2: U-Turn showing All 4 Control Signals; Note the y Axis has no Units as These are the Actual Numerical Values Supplied to the VTOL

Appendix A: (Continued)

Set 3: Ascending Spiral, all 4 control signals are shown below, with the x axis being the sample number for a total of 30 samples and the y axis showing the actual value.

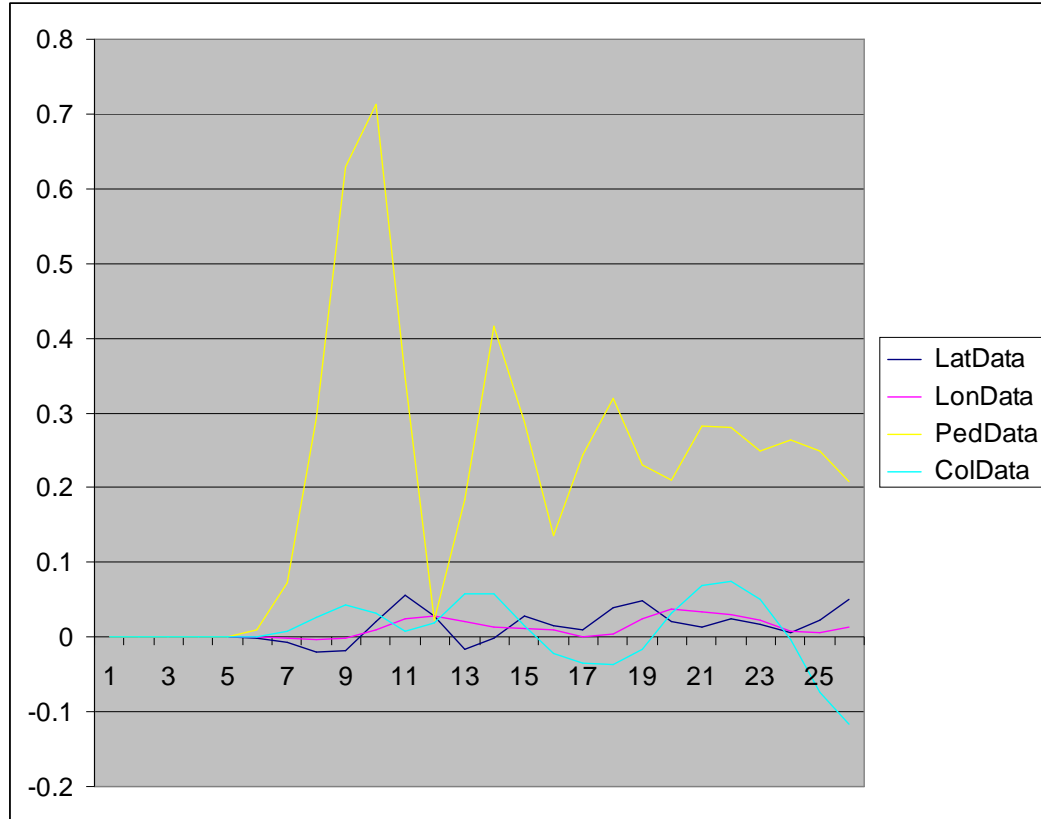


Figure A.3: Ascending Spiral Showing All 4 Control Signals; Note the y Axis has no Units as These are the Actual Numerical Values Supplied to the VTOL

Appendix A: (Continued)

Set 4: Variable height figure 8, all 4 control signals are shown below, with the x axis being the sample number for a total of 30 samples and the y axis showing the actual value.

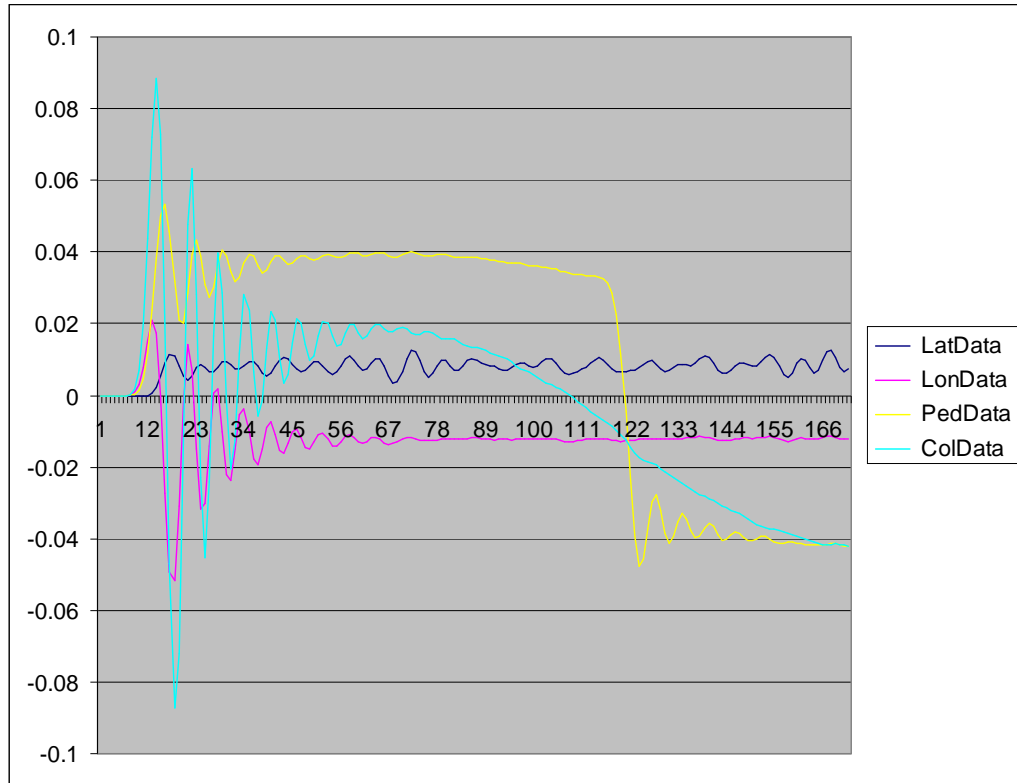


Figure A.4: Variable Height Figure 8 Showing All 4 Control Signals; Note the y Axis has no Units as These are the Actual Numerical Values Supplied to the VTOL

Appendix B: Brief VTOL History

B.1 List of Notable VTOL Development

1. Earliest VTOL discussed in 1928, by Nikola Tesla; Tesla received patents for a machine called the "*Flivver*" or an apparatus for aerial transportation. This is considered one of the earliest VTOL examples.
2. German designers, in World War II, conducted studies on possible VTOL designs and drew up designs for the Heinkel Lerche; however, it was never built.
3. In 1953, Rolls-Royce developed a 'Thrust Measuring Rig' or 'flying bedstead'. This machine led to the development of British VTOL engines that were used in the Short SC.1 (1957); This VTOL used 4 vertical lift engines and used an additional engine for horizontal thrust.
4. Later, scientist explored the idea of using the same engine for both vertical and horizontal thrust by changing the vectoring of the thrust. This effort resulted in the Bristol Siddeley Pegasus engine; this engine used rotating ducts to control the direction of thrust. Also developed at the same time, was the Hawker P.1127 airframe, which was later produced as the Hawker Siddeley Harrier. Note that the supersonic Hawker Siddeley P.1154 effort was discontinued in 1965.
5. The British Harrier aircraft has a STOVL flight mode that allows it to carry a higher fuel or weapon load. The Marine Corps, Italian Navy and the Spanish Navy use an advanced Harrier model called the AV-8 Harrier II. However, the Harrier II is outdated and its planned replacement is a variant of the F-35 Joint Strike Fighter modified to operate in STOVL mode.
6. The National Aeronautics and Space Agency (NASA) has also been involved in VTOL research and has operated VTOL aircraft such as the XV-15 (1977); The Soviet Navy tested a Sikorsky aircraft named the X-Wing, which can take off like a helicopter and then the rotors would become stationary in horizontal flight functioning as wings and producing lift. Additional lift is also provided by the static wings. The US tested a similar aircraft named the Canard Rotor/Wing prototype or Boeing X-50.

Appendix B: (Continued)

7. The French developed an updated version of the Dassault Mirage III in the 1960s; this plane was capable of reaching Mach 1. The Dassault Mirage III - V *Balzac* is able to transition from vertical to horizontal flight; this plane was tested in March of 1966 and was able to reach speeds of Mach 1.3 in level flight.

8. The Soviet Union at the height of the Cold War was working on many VTOL type aircraft such as the Yak-38 Forger which was a modified Yak-36 Freehand experimental plane. Shortly before the collapse of Soviet Union, a supersonic VTOL plane was developed to replace the Yak-38; this plane was called the Yak-141. The Yak-141 is also known as the Yak-41 was later developed into the Yak-43.

9. Germany during the late 1960s and early 1970s was working on three different VTOL planes. The first F-104 was used as a base for researching V/STOL planes. Eventually, two models were completed as the X1 and X2. However, the project was later canceled due funding and political issues at the time. The EWR VJ 101C aircraft managed to operate as a VTOL completing take-offs and landings as well as reaching level speeds of mach 1 and higher. One example is preserved in the Deutsches Museum in Munich, Germany and can be seen till this day. Figure B.1 shows a V/STOL VJ101. The Germans also developed a VFW-Fokker VAK 191B light fighter and the Dornier Do 31E-3 which was designed as a troop transport. These two prototypes are also preserved at the Deutsches Museum branch at Oberschleißheim Airfield.

Appendix B: (Continued)



Figure B.1: A German V/STOL VJ101 "Starfighter." (source: http://content.answers.com/main/content/wp/en-commons/thumb/0/09/180px-Aircraft_VJ101C_top.jpg)

10. The Moller Skycar was designed as a personal air vehicle or (PAV). However it has not been able to transition to level flight; neither has it carried any pilots on board.

11. Since spacecraft operate in environments that have virtually no runways, extraterrestrial missions also require VTOL designs; one interesting example is the LLRV.

12. The US Marines required a more advanced transport to replace their UH-60 Black Hawk helicopter. The V-22 Osprey (Figure B.2) was designed as its possible replacement and is the first production tilt-rotor aircraft in the world. It has one three-bladed prop-rotor, turboprop engine along with a transmission nacelle on each wingtip. The Osprey is classified as a joint service, multimission, military tilt-rotor aircraft that has vertical takeoff and landing (VTOL) capabilities along with short takeoff and landing capability (STOL).

Appendix B: (Continued)



Figure B.2: V-22 Osprey. Picture Shows Marines Jumping. (source: <http://content.answers.com/main/content/wp/en/thumb/a/af/180px-Aircraft.osprey.678pix.jpg>)

13. A new Joint Strike Fighter (JSF) plane being developed is the X-35B, which has STOVL capabilities as it is able to take off in less than 500 feet. It also has supersonic and vertical landing capability. The JSF program was created with the intent of keeping development and production cost down by building three variants of one aircraft that share about 80% of their parts. The demonstrator X-35 aircraft originally flew in 2000; a production version of this plane made its maiden flight on 15 December 2006. An X-35B showing the lift fan is demonstrated in figure B.3.

The three variants are:

- F-35A: This is a conventional takeoff and landing (CTOL) aircraft.
- F-35B: This is a short-takeoff and vertical-landing (STOVL) aircraft.
- F-35C: This is a carrier-based aircraft.

Appendix B: (Continued)



Figure B.3: X-35B Showing Lift Fan (source: http://upload.wikimedia.org/wikipedia/commons/thumb/7/78/F-13_lift_fan.jpg/180px-F-13_lift_fan.jpg)

Appendix C: Run Time Analysis of GA/SA Algorithm

The theory of computer complexity deals with the relative computational effort required to solve computable functions. The term Computable functions, also known as Turing-computable functions, refers to the basic unit of study in the field of computability theory. The advantage of using the theory of Computable functions is in the ability to evaluate and quantify computability without the need for a concrete “model of computation”. Hence, Turing machines and register machines are examples of theoretical operations that can be used to evaluate the complexity of a given algorithm. Church-Turing describes computable functions as functions that can be calculated with a mechanical device provided that there is an unlimited amount of time along with an endless supply of storage space. His thesis concludes that if a function is said to have an algorithm then it is computable.

An abstract computational complexity theory may be modeled on the Blum axioms; the latter means that the task of determining the computational complexity becomes a function problem. This is different from computability theory, which is used to determine if a problem is solvable, independent of the resources needed. More formally, the time complexity of a problem is defined as the number of steps that will be needed to solve a specific instance of the problem expressed as a function of the input size. It is also assumed that the most efficient algorithm available is used to solve the function. Hence the time complexity represents the best case or optimal performance time to solve that specific problem. As such, if an instance of problem A, for example, has n bits (input size) and needs n^2 steps to solve then it is considered to have a time complexity of n^2 . However, the computer language used to implement an algorithm can make a difference in the exact number of steps. In order to avoid needless details, a format known as the Big O notation (also known as order of calculation) is used to describe the time complexity of a specific algorithm. Hence, Big O notation of $O(n^2)$ indicates that a given algorithm will have the same time complexity on most computers.

Appendix C: (Continued)

Complexity can vary, for example, a linear time complexity indicates that as the input grows bigger, the computational time increase linearly. However, the time it takes to find an entry in a dictionary is logarithmic because even if the size of the dictionary doubles (doubling the input), the larger size will only increase the computational time by one step. As soon as the dictionary is opened in the middle, the problem size becomes half of what it was. Generally, different classes of computational time for sufficiently large values of n can be summarized as follows:

$$\log n < n < n \log n < n^2 < n^3 < 2^n$$

In addition, algorithms of different computational complexities are affected differently by an increase in CPU or computational speed as shown in table C.1. For example, linear order algorithms have a linear relation to speed. However, exponential algorithm run times are hardly affected by even computers that are 1000 times faster.

Table C.1: Computational Complexity Classes. (source: http://en.wikipedia.org/wiki/Computational_complexity_theory)

Time complexity function	Size of Largest Problem Instance solvable in 1 Hour		
	With present computer	With computer 100 times faster	With computer 1000 times faster
n	$N1$	$100N1$	$1000N1$
n^2	$N2$	$10 N2$	$31.6 N2$
n^3	$N3$	$4.64 N3$	$10 N3$
2^n	$N4$	$N4+6.64$	$N4+9.97$
3^n	$N5$	$N5+4.19$	$N5+6.29$

Appendix C: (Continued)

Time Complexity Calculation for the GA/SA Algorithm.

Let:

n = number of input rows

T = Annealing Temperature or Number of Generations

The pseudo code for the algorithm becomes:

*for (row = 0 to n; n+s) where n = number of rows and s = number of steps
for (time = 0 to t) where t is 20,000 (or some fixed number), note number of
generations can be substituted for time.
(n/s) * t → for sufficiently large values of n, s is small and t can be treated as a
constant value or c*

Hence, the calculation becomes:

$$n * c = O(n)$$

In other words, the GA/SA algorithm is a first polynomial algorithm. This class of algorithms benefits the most from technological advancements that result in increased computational speed. This can be seen in table C.1 where the effects of faster computers on computing speed are shown.

ABOUT THE AUTHOR

Namir Aldawoodi received a Bachelors degree in Computer Science and a second Bachelors in Computer Engineering from the University of South Florida in 1996, a Masters in Business Administration (MBA) from NOVA Southeastern University in 1999, and, while working on his Ph.D., a Masters of Science in Computer Science and Engineering from the University of South Florida in 2002.

Since 1996, Namir worked as an Engineer at GTE Data Services (GTEDS) which later became Verizon Information Technology. Namir has published three papers relating to his research at USF, they are all published with the Genetic and Evolutionary Computation Conference (GECCO). One paper was published and presented at the 2003 GECCO conference. Two additional papers were published at the 2004 GECCO conference.

Namir's other interests include Artificial Intelligence, Robotics, JAVA programming, Quantum Physics, Classic Music Composition, and Classic Art.