PARALLEL EVOLUTIONARY PROGRAMMING TECHNIQUES FOR STRATEGY OPTIMISATION IN AIR COMBAT SCENARIOS

A thesis submitted in fulfillment of the requirement for
for the degree of Doctor of Philosophy

ISTAS F. NUSYIRWAN
B.Eng, M.Eng

School of Aerospace, Mechanical and Manufacturing Engineering
RMIT University

March 2008

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Istas F. Nusyirwan

( )

Abstract

Air combat between fighter missiles and aircraft can be categorised as a pursuit-evasion problem. One aircraft acts as a pursuer and the other as an evader. Generally, the pursuer will try to capture the evader as quickly as possible and the evader tries to evade capture for as long as possible. For an experienced human pilot, it is trivial to discuss this methodology, but to simulate it, the mathematics involved is very complex and difficult to implement in a computer environment. Classical methods, though very accurate in their analysis, are not suited to solve a complex 6DOF pursuit-evasion problem and they have limitations in representing real-world problems such as discontinuities, discrete, stochastic, chaotic, temporal information or lack of information. In this thesis, Evolutionary Programming (EP) is applied to determine the optimum manoeuvring strategy for an aircraft (evader) to avoid interception by an incoming missile (pursuer). EP is a class of algorithms known as Evolutionary Algorithm (EA). EA has an ability to find an optimal solution in a complex problem which involves discontinuities, discrete, nondifferential parameters and noise. In addition, the methodology was implemented on parallel computer architecture to improve the computing time and expanding the search space. A sensitivity analysis was carried out to determine the best configuration and to understand the effect of parameters, such as number of processors, population size, number of generations, etc., on the results. The effects of sensor and instrument errors were also considered. The method enabled feasible solutions to be found in a relatively short period of time. However, the ability to search for feasible solutions is dependent on various parameters such as initial conditions, aircraft configurations and aerodynamic constraints.

It is concluded that, in general, EP is able to determine feasible maneuvering strategies for an evader to avoid interception with and without instrument errors. The methodology has the potential to be used as a training tool for pilots in air combat or as an intelligent engagement strategy for autonomous systems, such as Unmanned Air Combat Vehicles (UCAV).

Thank you for your love, support and trust in me.

To Mom and Dad, my beloved wife and children.

Acknowledgments

In the realisation of this thesis, numerous persons and organisations have contributed in terms of expertise contribution and financial assistance. Notably, I would like to mention:

- Assoc. Prof. Cees Bil & Dr. George Galanis. As the supervisor and co-supervisor of this project, for the help, guidance and support have permitted the realisation of this thesis.
- Surya, Faisal, Fareez and Firzanah. My wife and children for their patience, understanding and sacrifice.
- Victoria Partnership of Advanced Computing. For providing an excellent computing environment in realising the project.
- the Ministry of Higher Education Malaysia. For providing the much needed financial assistance.
- Universiti Teknologi Malaysia. For allowing me to come to RMIT, Melbourne to further my passion in research on aircraft.
- My parents who encouraged and supported me in pursuing my career.
- All my beloved friends at the Wackett Centre.
- All my fellow Malaysian postgraduates in Melbourne.

Table of Contents

List of Figures

List of Tables

Definitions

Evolutionary Algorithms

Gametes Reproductive cells (sperm and egg) that fuse to form a zygote.

Genes Units of hereditary information. Genes contain the instructions for the production of proteins, which make up the structure of cells and direct their activities.

Genotype The entire genetic identity of an individual, including alleles, or gene forms, that do not show as outward characteristics.

Heredity The handing down of certain traits from parents to their offspring. The process of heredity occurs through the genes.

Phenotype The visible properties of an organism that are produced by the interaction of the genotype and the environment.

Jink A large amplitude periodic manoeuvre.

List of Abbreviations

3D	three dimensions
AAM	air-to-air missile
AC	aerodynamic centre
AOA	angle of attack
AON	angle of the nose/tail
BVR	beyond visual range
C^3	command, control and communications
CFD	computational fluid dynamics
CG or cg	centre of gravity
DOF	degree-of-freedom
EP	Evolutionary Programming
EA	Evolutionary Algorithm
GA	Genetic Algorithm
HUD	head up display
INS	inertial navigation system
IR	infrared
KIAS	knots indicated airspeed
KTAS	knots true airspeed
L/D	lift-to-drag ratio
LOS	line of sight
MIMD	multiple instruction/multiple data

PC	personal computer
PNG/PN	Proportional Navigation System
RAM	Random Access Memory
RF	Radio Frequency
SIMD	single instruction/multiple data
UMA	uniform memory access

List of Notations

Symbol	*Definition*	*Units*
α	Angle of attack, $\alpha = \tan^{-1}\left(\frac{w}{u}\right)$	rad
β	Angle of sideslip, $\beta = \tan^{-1}\left(\frac{v}{u}\right)$	rad
a	Speed of sound	ms^{-1}
a_n	Normal acceleration, positive along the z-axis	g units
a_Y	Lateral acceleration, positive along the y-axis	g units
b	Wing span	m
c	Specific fuel consumption	g/kN·s
$\bar{c}$	Mean aerodynamic chord (MAC)	m
C_D	Aircraft total drag coefficient, $\frac{D}{\frac{1}{2}\rho V^2 S}$	
C_L	Aircraft total lift coefficient, $\frac{L}{\frac{1}{2}\rho V^2 S}$	
C_l	Rolling moment coefficient.	
C_m	Pitchings moment coefficient.	
C_n	Yawing moment coefficient.	
C_{m_α}	Partial derivative of the pitching moment coefficient with respect to the angle-of-attack.	
C_{n_β}	Partial derivative of the yawing moment coefficient with respect to the sideslip angle.	
$C_{l_{\delta l}}$	Partial derivative of the rolling moment coefficient with respect to the roll motivator deflection	
$C_{m_{\delta m}}$	Partial derivative of the pitching moment coefficient with respect to the pitch motivator deflection	
$C_{n_{\delta n}}$	Partial derivative of the yawing moment coefficient with respect to the yaw motivator deflection	

C_{l_p}	Partial derivative (damping) of the rolling moment coefficient with respect to normalised rate of roll.	
C_{mq}	Partial derivate (damping) of the pitching moment coefficient with respect to the normalised rate of pitch	
C_{nr}	Partial derivative (damping) of the yawing moment coefficient with respect to the normalised rate of yaw.	
$C_{m_{\dot{\alpha}}}$	Partial derivative (damping) of the pitching moment coefficient with respect to the normalised rate of change of the angle-of-attack.	
$C_{n_{\dot{\beta}}}$	Partial derivative (damping) of the yawing moment coefficient with respect to the normalised rate of change of the sideslip angle.	
C_X	X force coefficient, $\frac{X}{\frac{1}{2}\rho V^2 S}$	
C_Y	Y force coefficient, $\frac{Y}{\frac{1}{2}\rho V^2 S}$	
C_Z	Z force coefficient, $\frac{Z}{\frac{1}{2}\rho V^2 S}$	
C_{X_α}	Partial derivative of the force coefficients with respect to the angle of attack.	rad^{-1}
C_{Z_α}	Partial derivative of the force coefficients with respect to the angle of attack.	rad^{-1}
C_{X_β}	Partial derivative of the force coefficients with respect to the sideslip angle.	rad^{-1}
C_{Y_β}	Partial derivative of the force coefficients with respect to the sideslip angle.	rad^{-1}
$C_{X_{\delta l}}$	Partial derivative of the force coefficient with respect the roll motivator deflection	rad^{-1}
$C_{Y_{\delta n}}$	Partial derivative of the force coefficient with respect the yaw motivator deflection	rad^{-1}
$C_{Z_{\delta m}}$	Partial derivative of the force coefficient with respect the pitch motivator deflection	rad^{-1}
d_{int}	The interception radius	m
F	The resultant of the airframe aerodynamic forces acting on the vehicle	N

g	Acceleration due to gravity	$\mathrm{m/s^2}$
h	Altitude	m
I_x	Moment of inertia with respect to the body axes about the x axis.	$\mathrm{kgm^2}$
I_y	Moment of inertia with respect to the body axes about the y axis.	$\mathrm{kgm^2}$
I_z	Moment of inertia with respect to the body axes about the z axis.	$\mathrm{kgm^2}$
I_{zx} or I_{xz} or I_{yz}	Product of inertia with respect to the body axes.	$\mathrm{kgm^2}$
I_{zy} or I_{xy} or I_{yx}	Product of inertia with respect to the body axes.	$\mathrm{kgm^2}$
M	Mach number , $M = V/a$	
L	Rolling moment or lift force	$\mathrm{N \cdot m}$ or N
M	Pitching moment	N·m
N	Yawing moment	N·m
$L_{\delta l}$	Partial derivative of the rolling moment with respect to the roll motivator.	$\mathrm{N \cdot m \cdot rad^{-1}}$
$M_{\delta m}$	Partial derivative of the pitching moment with respect to the pitch motivator deflection.	$\mathrm{N \cdot m \cdot rad^{-1}}$
$N_{\delta n}$	Partial derivative of the yawing moment with respect to the yaw motivator deflection.	$\mathrm{N \cdot m \cdot rad^{-1}}$
L_p	Partial derivative of the rolling moment with respect to the rate of roll.	$\mathrm{N \cdot m \cdot srad^{-1}}$
M_q	Partial derivative of the pitching moment with respect to the rate of pitch.	$\mathrm{N \cdot m \cdot s \cdot r^{-1}}$
N_r	Partial derivative of the yawing moment with respect to the rate of yaw.	$\mathrm{N \cdot m \cdot s \cdot r^{-1}}$
M_w	Partial derivative of the pitching moment with respect to the velocity component w.	$\mathrm{N \cdot s}$
P_1	Engine power command based on throttle position	%
P_2	Engine power command to engine	%
P_3	Engine power	%
V	Velocity vector	m/s
m	Mass	kg

p	Rate of roll. For RH axis system with transverse axis (y) positive to the right. Component of the angular velocity Ω along the x-axis.	rad/s
q	Rate of pitch. For RH axis system with transverse axis (y) positive to the right. Component of the angular velocity Ω along the y-axis.	rad/s
r	Rate of yaw. For RH axis system with transverse axis (y) positive to the right. Component of the angular velocity Ω along the z-axis.	rad/s
Re	Reynold's number	
S	Wing area	m^2
u	For a given axis system. Component of vehicle velocity V along the x-axis	m/s
v	For a given axis system. Component of vehicle velocity V along the y-axis	m/s
w	For a given axis system. Component of vehicle velocity V along the z-axis	m/s
x	For a given axis system. Component of vehicle velocity R along x axis	m
y	For a given axis system. Component of vehicle velocity R along y axis	m
z	For a given axis system. Component of vehicle velocity R along z axis	m
ρ	Density	$\frac{\mathrm{kg}}{\mathrm{m}^3}$
n_p	Size of the population	
n_{gen}	Total number of generation	
$\bar{q}$	The free stream dynamic pressure	$\mathrm{N/m}^2$
T	Total instantaneous engine thrust	N
T	Temperature	Kelvin
T	Thrust	N
T_{idle}	Idle thrust	N
T_{max}	Maximum thrust	N
T_{mil}	Military thrust	N

t	Time	sec
T_c	Communication time	
T_f	Time to evaluate one individual	
T_p	Elapsed time for one generation	
x_{cg}	Centre-of-gravity location, in fraction of $\bar{c}$	
γ	Flight path angle	deg
ψ	Heading angle	deg
σ	Bank angle	deg
δ_a	Aileron deflection, positive for left roll	deg
δ_e	Elevator deflection, positive for airplane nose-down control	deg
δ_r	Rudder deflection, positive for left roll	deg
$\mathcal{P}$	Number of Processors used	
$\mathcal{S}$	Number of Slave Processors, P - 1	
τ_T	Engine thrust time constant	s

where,

$$C_{l_p} = \frac{\partial C_l}{\partial \frac{pb}{2V}} \quad C_{l_r} = \frac{\partial C_l}{\partial \frac{rb}{2V}} \quad C_{l_\beta} = \frac{\partial C_l}{\partial \beta} \quad C_{l_{\delta a}} = \frac{\partial C_l}{\partial \delta_a}$$
$$C_{l_{\delta r}} = \frac{\partial C_l}{\partial \delta_r} \quad C_{m_q} = \frac{\partial C_m}{\partial \frac{q\bar{c}}{2V}} \quad C_{n_p} = \frac{\partial C_n}{\partial \frac{pb}{2V}} \quad C_{n_r} = \frac{\partial C_n}{\partial \frac{rb}{2V}}$$
$$C_{n_\beta} = \frac{\partial C_n}{\partial beta} \quad C_{n_{\beta,dyn}} = C_{n_\beta} - \frac{I_Z}{I_X} C_{l_\beta} \sin\alpha \quad C_{n_{\delta_a}} = \frac{\partial C_n}{\partial \delta_a} \quad C_{n_{\delta_r}} = \frac{\partial C_n}{\partial \delta_r}$$
$$C_{X_q} = \frac{\partial C_X}{\partial \frac{q\bar{c}}{2V}} \quad C_{Z_q} = \frac{\partial C_Z}{\partial \frac{q\bar{c}}{2V}} \quad C_{Y_p} = \frac{\partial C_Y}{\partial \frac{pb}{2V}} \quad C_{Y_r} = \frac{\partial C_Y}{\partial \frac{rb}{2V}}$$

List of Publications

Journal Article

1. Istas F. Nusyirwan and C. Bil, *Methods of Aircraft Trajectory Optimisation in Air Combat*, ANZIAM Journal, Vol. 47, pp. C649-C664, 2007.

Refereed Conference

1. I. F. Nusyirwan and C. Bil, *Sensitivity in Trajectory Optimisation for Aircraft in Air Combat*, In Proceeding of Simtect Conference, 9-12 May, Sydney, Australia, 2005.

2. I. F. Nusyirwan and C. Bil, *Methods of Aircraft Trajectory Optimisation in Air Combat*, in proc. of EMAC Conference, 25-28 September , RMIT University, Melbourne, Australia, 2005.

3. I. F. Nusyirwan and C. Bil, *Software Design to Solve Real Time Optimisation of Proportional Navigation Guidance using Genetic Algorithm*, in Proc. of The 1st Malaysian Software Engineering Conference, 12-13 December, Penang, Malaysia, 2005.

4. I. F. Nusyirwan and C. Bil, *Optimal Real Time Evasion Against High Speed Pursuer Using Evolutionary Programming*, in Proc. of the 1st Regional Conference on Vehicle Engineering and Technology, 23-25 May 2006, Kuala Lumpur, 2006.

5. I. F. Nusyirwan and C. Bil, *Three-Dimensional Air Combat: Numerical Solution Using Randomised Trajectory*, in Proc. of Simtect Conference, 29 May - 1 June, Melbourne, Australia, 2006.

6. I. F. Nusyirwan and C. Bil, *Effect of Uncertainties on UCAV Trajectory Optimisation Using Evolutionary Programming*, in Proc. of Intelligence, Decision and Control Conference, 11-14 Feb., Adelaide, Australia, 2007.

7. I. F. Nusyirwan and C. Bil, *Parallel Evolutionary Programming Optimisation for Evader's Trajectory in Pursuit-Evasion Game*, in Proc. of the Twelfth Australian International Aerospace Congress, 19-22 Mar, Melbourne, Australia, 2007.

8. I. F. Nusyirwan and C. Bil, *Optimising Evader's Trajectory in 6DOF Pursuit-Evasion Problem Using Parallel Evolutionary Programming*, in proc. of Simtect Conference, 4-7 June, Brisbane, Australia, 2007.

9. I. F. Nusyirwan and C. Bil, *Using Evolutionary Programming to Find Optimal Evasion Strategy Against an Agile Pursuer*, Oral and Poster Presentation, 22nd International Unmanned Air Vehicle Systems Conference, April 16-18, Bristol, UK, 2007.

Poster Presentation and Oral Presentation

1. I. F. Nusyirwan and C. Bil, *Pursuer-Evasion Strategy Optimisation In Air Combat Scenarios*, Oral Presentation at the Australian JSF Advanced Technology and Innovation Conference, 10-11 July 2007, Melbourne, Australia.

1 Introduction

In an aircraft-versus-missile encounter, the objective of the pilot is to use the best piloting skills and aircraft performance capabilities to out manoeuvre the missile and to avoid interception. Although modern fighter aircrafts are fitted with counter-measures, such as chaff and flares, they are not always effective as missiles are becoming more intelligent in avoiding and ignoring these decoys. The ultimate line of defense for an aircraft against an incoming missile is to adopt a manoeuvring strategy that will avoid the missile from coming close. Since a missile has limited endurance, keeping the missile at a distance that is far enough will eventually cause it to bleed its energy and fall off from the sky. Modern missiles have tracking and homing devices based on radar or infrared sensors. These sensors need to be locked on to the target for guidance. Smart manoeuvring by the evader can cause the missile to lose the target acquisition and make it go into a search mode. Such an air combat can be described as a pursuit-evader problem. A game theory can be applied to solve problems of this nature. The aim is to find a better solution that could give the evader a better chance of being intercepted.

Optimisation is important factor in any decision-making process. The three main components in the optimisation process are the players, the objective function and the information available to the players [53][77]. The game theory falls into the optimisation category. It is a study of strategic or intelligent interactions between players with conflicting objectives. The modern game theory was first explored by John Von Neumann, who was the first to prove the minimax theorem. In 1944, Neumann and Morgenstern published one of their pioneering works on game theory [128], which included the two-person zero-sum game and the notion of cooperative game.

Historically, the optimal game theory has been around since 500AD [129]. Although, it is not fully recognized as an optimal game until 1985, it is mentioned in the Babylonian Talmud to solve marriage contract problems using an appropriately defined game methodology between a man and his three wives. Furthermore, in 1950, John Nash proved the existence of a strategic equilibrium, known as the

Nash Equilibrium for non-cooperative games [84]. From this point on, the research on the game theory started to increase and has led to the emergence of a theory of dynamic games.

Dynamic games are a subset of the game theory. This class of games evolves in, discrete or continuous, time. The origin of such games can be linked to the mixed-strategy in a finite-matrix game [9]. The game is interpreted as the relative frequency of each move, called a pure strategy, by repeating the same game multiple times [9]. In a discrete time dynamic game, each player has a *payoff* or objective function associated with multiple stages. The transition of the game from one stage to the next is governed by differential equations. It requires inputs from all of the players. If the evolution of a game is formulated and solved by differential equations, the dynamic game is known as a differential game.

One of the optimisation problems that is usually linked to differential games is the pursuit-evasion problem between two or more players, such as in air combat with aircraft and missiles. This problem is also called a two-player zero-sum differential game. Zero-sum means one player's gain is another player's loss. The idea of the differential games was pioneered by Isaacs in the 1950s during his tenure as a researcher at the Rand Corporation [59]. Isaacs studied the two-player zero-sum game with players that have conflicting interests. The research in differential games was intensified and the theory of two-player zero-sum differential game was subsequently generalised to the *n*-player non-cooperative game [9]. The development of differential game theory is closely related to the optimal control theory. An optimal control problem can be equated with a differential game with one player. The essence of optimal control theory includes a dynamic programming and the minimum principle.

The minimum principle was introduced by Pontryagin in the 1960s [67]. It states that the optimal trajectories of a system must satisfy the Euler-Lagrange equations. At each point along the trajectory, the optimal control must minimize the Hamiltonian. Such an approach deals with one extreme associated with one state at a time, hence making the optimal control an open-loop problem.

The pursuit-evasion problem in an air combat involves the optimisation of an aircraft manoeuvring strategy. The evader aims to out manoeuvre the pursuer. It can be categorised as a problem of a dynamic multi-player non-cooperative game theory [9]. The game involves multi-player decision making. It is dynamic because the order in which the decisions are made is important and it is non-cooperative because each player involved pursues its own interest which is conflicting with others.

This kind of approach is commonly called “the differential game” because, generally, differential equations are used to formulate the problem. A more appropriate term would be dynamic game [9].

The ability to determine an optimal solution for an air combat engagement is potentially a powerful tool for fighter aircraft development, pilot training, as an in-flight advisory system or as an intelligent system for unmanned air combat vehicles (UCAV). Presently, the search for the optimal solution is done *a priori*, i.e. off-line. However, with the introduction of more powerful and multi-processor computers, it may be possible to do the calculations in real-time by using a central server located remotely and the results can be transmitted back to the aircraft within a relatively short period of time.

Many methods for finding optimal solutions for an air combat maneuvering have been proposed, such as the game theory. However, most of them require the model to be simplified to make them tractable. This simplification would render the method not suitable for determining realistic and practical solutions.

One method that allows the incorporation of a full non-linear model is Evolutionary Algorithms (EAs). The objective of this thesis is to explore the feasibility of using EAs to find a feasible solution for air combat 3D manoeuvring between two players, an aircraft (evader) and a missile (pursuer) with different levels of agility. This is done incorporating a full 6DOF non-linear model for the evader with the presence of noise and disturbances. The pursuer is adequately modeled as a point-mass.

1.1 Modern Air Combat

Modern air combat has gone into a new era, where technology plays a dominant role. Current weaponry such as an air-to-air missile is capable of detecting, tracking and intercepting aircraft beyond visual range. Due to this advancement, the typical close-in “dog fight” scenario is becoming less common. A new type of aircraft that will soon dominate the role of air superiority is the Unmanned Combat Aerial Vehicle (UCAV). UCAVs will play, for example, an important role in first strike scenarios to take out defense installations well into an enemy’s territory without putting pilots or valuable equipment in a dangerous situation. To resolve this dangerous predicaments, new technologies are introduced and performance boundaries are being extended [79].

In contrast to the piloted aircraft, UCAV is an affordable weapon system that expands tactical mission options for revolutionary new air power as an integrated part of a systems solution [27]. The benefits of using UCAVs are numerous. Locating the pilot outside the loop enables a new paradigm in aircraft command and control while maintaining the rationale, judgment, and quality of the performance. In the future, UCAVs are envisioned to be able to be able to be remotely and globally deployed, be safely operated over populated regions, to observe and obey the rules of engagement and make critical decisions depending on the doctrine used. Presently, such a system does not exist due to the complexities involved [27].

Many techniques have been proposed to provide “intelligence” to the aircraft. However, each technique has its own advantages and disadvantages as listed below:

1. **Artificial intelligence system**. This is a rule-based approach derived from human think. Its effectiveness is solely relying on the input given by a human expert. If properly developed and configured, the system is consistent, fast and reliable. But it lacks inspiration to deal with situations beyond its scope of capability.

2. **Differential Games**. This technique uses mathematics to study of conflicts of dynamical system. It is widely applied to solve simple dynamic problems. But, as the problem grows larger, the technique becomes intractable.

3. **Evolutionary algorithms**. This algorithm is based from Darwinian Theory of evolution. It is capable to locate feasible solutions for problems of higher dimensions. But there is no guarantee that it could find feasible solutions at every run.

4. **Fuzzy logic**. A set theory that deals with reasoning. It has similarity to Artificial intelligence system.

5. Neural network. This is an optimisation approach derived from the way the human brain function. It requires a substantial amount of *a priori* knowledge or experience to function properly.

Generally these techniques fall into two distinct types of optimisation:

- Classical optimisation methods - either direct or indirect optimisation, such as differential games.
- "Black-box" optimisation methods - there is no solid theoretical background. These methods include Evolutionary Algorithms, Artificial Intelligence, Fuzzy Logic and Neural Network.

Classical optimisation methods are used to solve differential, continuous, strongly convex, unimodal and many other special problems. However, "blackbox" optimisation methods, such as EAs, seem to excel in areas where discontinuous, non-differentiable, multimodal, noisy and other unconventional response surfaces are involved [5].

1.2 Problem Statement

The objective of this research is to model and simulate a pursuer-evader scenario in close to real time based on a parallel architecture and apply the Evolutionary Programming algorithm to optimise manoeuvring strategies for the evader. The aircraft model will include realistic constraints of an aircraft. In order to do so, the equations of motion will contain non-linearities and are non-differentiable functions.

The search for an optimal trajectory is in general quite complex because it usually consists of many dimensions with potentially significant coupling of the dynamics. The search space should also exhibits discontinuities, which makes it difficult to apply gradient-based techniques. To overcome these obstacles, the use of evolutionary programming as the basis of such a planning system looks ideal. The motivation behind this approach is based on the observation of evolution in nature, which is a very effective natural optimiser. The dynamic strategy planning problem, in the appropriate context, allows the application of the principles of natural selection to simulate the evolution of potential strategies or solutions.

Based on this idea of natural optimisation, the decision space is spawned by many degrees of freedom to effectively search for highly effective strategies. This research is not meant to search for the absolute optimal solution in most cases, but, is looking for near optimal or feasible solutions that will satisfy the given constraints and continuously approach towards obtaining the optimal solution over time. An absolute solution is difficult to find due to the fact that there are uncertainties and the dynamic environment in which the vehicles operate. To develop such a technique to solve this problem is next to impossible. In addition, even to define, let alone find, the absolute optimal solution (pareto-front optimal solution) has yet to be discovered. Furthermore, the computation time required is also a factor as this technique is meant to be a real time process. On the other hand, it is adequate to rely on the ability to rapidly search for feasible solutions in the presence of an uncertain and dynamic environment that would be sufficient for a planning system. This is called a rapid adaptation plan [23].

1.3 Research Aims

The aim of this research is to increase the body of knowledge about the application of Evolutionary Programming to highly time-constrained, dynamic problems in non-linear, discontinuous domains. The inspiration for this research is the search for a near real-time solution to the pursuit-evasion problem that arises from an idealization of air combat scenarios involving a missile launched at a defending aircraft. The research forecasts that future systems might be capable of providing real-time manoeuvre control of a combat aircraft in a manner that allows for the successful defeat of a surface or air launched missile.

1.4 Thesis Outline

The thesis consists of seven chapters that will be presented accordingly. This introductory chapter or Chapter 1 provides an overview of the study. This is followed by Chapter 2 which presents the discussions on current methods and practice in pursuit-evasion problem research. The discussion in this chapter is based on the use of game theory, differential game and calculus of variation in finding optimal solutions for various types of pursuit-evasion problems. In Chapter 3, the application of evolutionary programming in various optimisation problems is discussed further as well as the explanations about the overall approach of Evolutionary Algorithms. The discussion includes the important steps used in the process. Following that, Chapter 4 contains discussions about the parallel implementation of EP. In this chapter, two approaches are presented: the master-slave approach and coarse-grained approach. Next, Chapter 5 comprises discussion about the detailed modelling of the aircraft (evader) and missile (pursuer). The aircraft model is based on the six degree of freedom equations of motion whereas the missile model is based on the three degree of freedom equations of motion (point mass model). Then, Chapter 6 presents the results of several simulations performed at different initial conditions. The results show that the optimal solutions found are highly dependent on the initial condition of each player. Finally, Chapter 7 presents the overall results, suggestions for further improvement that can be applied to the algorithm as well as other possible applications.

2 The Pursuit-Evasion Problem

This chapter discusses the progress of research in finding optimal methods to solve a pursuit-evasion problem between two air vehicles.

2.1 Introduction

Air forces around the world are pushing the technology further to make an unmanned combat aircraft commonly known as Unmanned Combat Air Vehicle to be intelligent and able to perform given tasks. Similarly, as explained in the previous chapter, an intelligent system is a prerequisite to help reduce the pilot's workload in the presence of danger, such as a missile attack. This "intelligence" is achieved by providing the necessary information such as the type of approaching missile, possible launch location, and measures that need to be taken.

In order to realise this concept, several obstacles must be overcome. One of them is the methodology to replace or assist the pilot as the decision maker to make the best decision at that point in time. Methodologies, generally classified as Game Theory, have been developed since the 1700s to solve the classic problem of strategy optimisation for conflict scenarios. In 1713, James Waldegrave was the first to provide a solution to a two person minimax strategy game.

There are two approaches [94] pertaining to air combat modelling: the mathematical model and the behavioral model. The mathematical model can be categorised into two types: game theory and Lanchester theory. Behavioral models are often illustrated in war games. They can be represented by figurines on sand tables. Using these sand tables, tactics are explored by moving miniature figurines or equipment around as depicted in Figure 2.1.

Another important aspect in finding solutions through behavioral models is the emergent behaviour [93]. As extensive modelling became common with the availability of low-cost digital computers, researchers often encountered unexpected results such as the end outcome does not equate to the sum of the parts. Such result could be viewed as the behaviour that is a natural representation of

Figure 2.1: A physical combat simulation using sand table [94].

the real world [93]. By including constraints and assumptions, we may unnecessarily reduce the fidelity of the model that does not represent the actual problem. In complex domains, non-linearities and emergent of properties make modelling difficult and tend to drive fidelity requirements. Air combat is a domain in which the complexity encountered presents a powerful argument for maintaining sophisticated model (beyond that what might be intuitively selected). It is for this reason that this research is based on complex real-time systems than those referenced elsewhere in the literature that has been carried out. For that matter, this proposed research uses a systematic approach that is required to fully understand how to extract the benefit from this combat behaviour.

2.2 Game Theory

In 1944, Von Neumann and Morgenstern [128] developed the game theory in its extensive form. They introduced several approaches of the games such as the strategic normal game, the strategic extensive game, the concept of pure/mixed strategies and coalitional games. They employed the "*maximin*" solution concept to solve simple strategic, zero-sum normal games.

Later on, in 1950, Nash introduced the concept of “Nash Equilibrium” (NE). In the concept, NE will be reached when a player has chosen a strategy and no player can benefit by changing his strategy. At the same time, the other players would have kept their strategies unchanged. The concept can be easily demonstrated for simple problems. But proved to be difficult for complex problems [69].

Isaac [59] cited there are two reasons why the theory of games fails to solve real-life problems. The first one is the increase of difficulty of the problem when there are two conflicting aims: one trying to achieve a maximum result whilst knowing its adversary is doing the same thing. A closed form feedback solution cannot be obtained for real problems [15]. The second reason is the lack of mathematical methods to find answers in real-life problems.

Most mathematicians tend to concentrate on the general theorems but provide very little of usable techniques to obtain practical answers. Breitner [15] stressed that mathematics tend to favour more abstract and general ends. These reasons hinder the use classical techniques to their fullest extend.

As for game theory applied to war scenarios, Isaac [59] outlines several possibility approaches to tackle these problems. One of them is using the discrete matrix games. In principle, any finite and discrete game can be presented in a matrix form. However, the problem is that the dimension can be astronomically large unless the game is kept simple. Hence, this approach, again, reduces the realism of the solution.

Due to the close relation between differential game theory and optimal control theory, the use of optimal control theory to solve differential games is possible [15]. Unfortunately direct methods based on a parameterisation of the state and/or the control variables, which are widely used in optimal control theory, cannot be applied to pursuit-evasion games. Breitner [15] suggests that only dynamic programming methods and indirect methods are appropriate for the numerical solution of pursuit-evasion games. Among these methods are differential dynamic programming, the gradient-restoration algorithm, the min-H method and the multiple shooting method. The problem is solved using the multiple-shooting method [15]. The disadvantage using this method is to find the “starting trajectory”, i.e. an initial guess for the Newton iteration. Even the terminal time, t_f, must be estimated accurately. This is why it is very difficult to predict what might happen in an actual scenario. Furthermore, the prediction has to be done in a short period of time.

The emergence of super computers has sparked an extensive research on this optimisation area [24]. Previously, numerically intensive algorithms were difficult to implement due to the absence of powerful computers. However, developing an intelligent system that can learn, adapt and make decisions in a complex environment still remains a challenge to many researchers.

2.3 Development of Game Theory

With reference to state and control variables and according to Isaac [59] "Players always make their decisions through choosing the values of certain control variables" and these "in turn , govern the values of certain other quantities called state variables". As a result of this, Isaac outlines two types of games [59], i.e. game of kind and game of degree. Game of kind can be easily explained as to whether it is possible for the evader to avoid interception whereas game of degree means, if there is interception, the question is how long can the evader hold on before being intercepted by the pursuer. Apparently, the idea of solving pursuit-evasion games has been on the mind of many researchers for years, including Isaac [59]. In pursuit-evasion games, the pursuer seeks to capture the evader, and the evader seeks to avoid interception. The question can be presented in two forms. The first one is, whether there is a possibility for the pursuer to capture the evader. In this case, the game can be considered as the game of kind, i.e. capture or no-capture. The second is, if capture is possible, the pursuer's ultimate goal is to find the optimal strategy to capture the evader within the shortest possible time. This approach was explored further by [16] by studying complex pursuit-evasion games with state variable inequality constraints.

The study of pursuit-evasion can typically be divided into two approaches, i.e. emphasis on the pursuer's ability to intercept the evader and vice versa. The evader has some restricted optimal behaviours, which is known by the pursuer. This approach was studied by Green [107], Vathsal [124], Imado [57], Menon [80], Shinar [110], Segal [102], Shinar [109], Oshman [92] and Shen [74]. These researchers have been focusing mainly on finding optimal guidance law for missile against mildly maneuvreable evaders. The other approach is emphasis on the evader's ability to avoid interception such as in [82; 12; 104]. For example, Blagodatski and Shevchenko [12; 104] explore the use of simple differential games to solve problems that consist of multiple pursuers and evaders.

By exploring the pursuit-evasion problem further, the use of Evolutionary Algorithms has shed some light on the quest to find optimal solutions, which classical methods were unable to do [6]. To demonstrate this, Fogel [5] has constructed a three dimensional model where EP was used to guide a pursuer towards a moving target.

There have been a few attempts using Evolutionary Algorithm in solving pursuit evasion problems. For example, Tahk [25] has attempted to find optimal evasion strategies against proportional navigation guided missiles in three-dimensions. He used three dimensional three degree of freedom equations of motion to model the vehicles. In the study, a co-evolutionary augmented Lagrangian method was implemented to solve the constrained optimisation problem. The optimisation is performed by transforming the problem into a zero-sum pursuit-evasion game between the parameter vector, x, and the Lagrange multiplier. Both players use separate evolutionary processes to find the optimal solution to get the best payoff. Finding optimal solutions is not easy as shown in this study because it took 10000 generations to find one optimal solution. The search for the final time was also included in the evolutionary process and this has made the approach unattractive for actual use. There was no mention of computing time in the paper.

2.4 Guided Missiles

It should be noted that the capability of guided missiles is important in this thesis because it has been used as the pursuer in this thesis. Knowledge of the capabilities of current generation missiles is essential in implementing accurate optimisation strategies. Basically, guided missiles will receive information on the location of the target and change their flight paths in response to target manoeuvring. Generally, guided missiles are categorised according to their designated mission. This is based on their launching platform and intended target such as:

1. Air-to-Air missile (AAM).
2. Surface-to-Air missile (SAM).
3. Surface-to-Surface missile.

This research deals primarily with air-to-air missiles (AAMs). A typical generic guided missile is show in Figure 2.2. The seeker section consists of a radar that

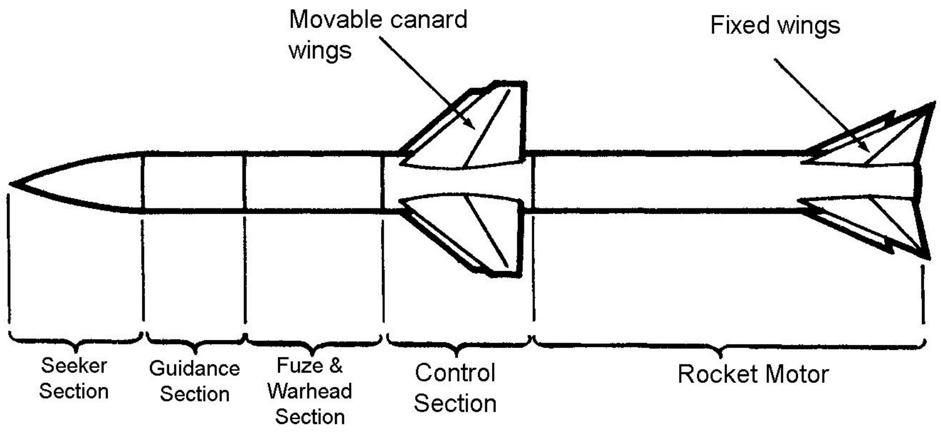


Figure 2.2: A typical guided missile. [103]

scans forward. The information received is then fed into the guidance computer. The lethal capability of the missile is in the fuse and warhead section that stores the explosive charge. The missile is guided by using four fins. Finally, the motor section provides the thrust required to propel the missile. AAMs are generally rocket powered for several reasons, i.e. they provide very high thrust-to-weight ratios, and generate high acceleration and high speed during the short duration of the flight.

In the missile, there are three types of seekers, i.e. active seeker, semi-active seeker and IR seeker. An active seeker uses its own radar to search for the target. A semi-active seeker does not emit any RF signal. However, it receives reflections from radar located at other location to determine the position of the target. An IR seeker is a passive seeker that detects IR signal radiated by the hotspots of the target aircraft. The information received is then fed into the guidance computer. The lethality of the missile is in the fuse and warhead section that stores the explosive charge. The missile is guided by using four fins. Finally, the motor section provides the thrust required to propel the missile. AAMs are generally rocket powered for several reasons, i.e. they provide very high thrust-to-weight ratios, and generate high acceleration and high speeds during the short duration of the flight.

For uses in optimal interception, there are two typical guidance systems, (i) proportional navigation and (ii) line-of-sight command guidance [13]. The missile receives the information on the target either from the reflected radio frequency (RF) signal or the infrared (IR) signal radiated from the target. A complete discussion about missiles and their navigation systems can be found in [112].

The missile guidance is can be divided into three phases [112]: (i) boost or launch, (ii) midcourse and (iii) terminal. The boost phase covers the time the missile leaves the launcher until it runs out of fuel. In this phase, the missile may not be actively guided. Next is the midcourse phase. This phase is usually the longest in terms of both time and distance. In this phase, the guidance system may or may not explicitly require bringing the missile onto the desired course. The guidance system has to ensure it stays on course until the missile enters a zone from which the final phase, i.e. the terminal phase takes over. In the terminal phase, the missile must possess a high accuracy and fast reaction for successful interception. In this phase, the guidance seeker is *locked* onto the target, thus allowing the missile to be guided all the way to the target.

There are many approaches in guiding the missile to the target. The prominent methods are [112] as follows

Command Guidance The guidance instructions come from external sources. No seeker is required for this type of missile.

Beam Rider The missile rides over a beam, typically laser or radar.

Command to Line of Sight (CLOS) In this guidance, the missile is commanded to fly in the Line of Sight (LOS).

Pursuit The missile flies directly toward the target at all times.

Deviated Pursuit The interceptor missile tracks the target and produces the guidance command. The missile heading usually leads the LOS by a fixed angle.

Lead Pursuit A lead pursuit course is achieved by directing its velocity vector at an angle from the target.

Lead Collision A straight-line course is flown by the missile. If the target speed and heading remain constant, the missile will fly a straight-line path to the target.

Pure Collision Flies a straight-line course such that it will collide with the target.

Constant Load Factor A guidance system that maintains a constant-g load factor on the missile that will result in a collision with the target.

Proportional Navigation The missile is flown in such a manner as to change the lead angle at a rate proportional to the angular rate of the line of sight to the target.

Three point The missile is constantly steered to fly between the tracker and the target.

Hyperbolic Guidance The guidance is based on the difference in the time of delay o4zf radio signals transmitted simultaneously from two ground stations, arriving at the missile at different time intervals.

Retransmission guidance Also known as track via missile. With this guidance system, a ground radar tracking system tracks both the target and the missile, and the target tracking beam serves as a target illuminator, while the receiver on the missile detects the reflected illumination.

Typically, most AAMs utilise proportional navigation as their guidance system. In the simulation model developed in this thesis, the pursuer uses a fixed strategy, i.e. a proportional navigation guidance system for interception.

2.4.1 Proportional Navigation Guidance

Proportional Navigation Guidance (PNG) systems are widely applied in AAMs due to their simplicity of implementation and robustness. In PNG systems, a missile is commanded to turn toward the target at a rate proportional to the angular velocity of the line-of-sight (LOS) between the missile and the target. The ratio of the missile turning rate to the angular velocity of the LOS is called the proportional navigation constant N. Typical values for, N are between 2 to 6. If N is greater than 1, the missile will be turning faster than the LOS and subsequently builds a lead angle with respect to LOS. For $N = 1$, the missile turns at the same rate as the LOS, i.e. homing in on the target. A complete discussion about PNG is in Section 5.7.4.

2.4.2 Missile Warhead

The type of warhead is important to intercept an aircraft. It determines the lethality of the missile that is the lethal radius. The bigger the lethal radius, the higher the lethality of the missile. The warheads used in AAMs are, typically [103]:

1. Blast-fragmentation,
2. Incendiary or explosive pellets, or
3. Expanding-rod.

Blast-fragmentation warheads are meant to create damages through the combined effects of the shockwave from the explosion, and high-velocity fragments. Pellet designs are almost similar [103], but some are made of small bomblets that explode or burn on contact.

Explosions at high altitude because of the thin air cannot solely be the reason for aircraft damage. The use of fragments increases the probability of target damage even though they rapidly lose lethal power as the missed distance increases. The expanding-rod system comprises of many short steel rods placed side by side in an annular arrangement around the explosive charge.

The lethal radius of an AAM depends on the type of warhead used. A typical lethal radius for a medium-range AAM is around 10-15m and the fuse detection radius is around 1.5 m for a Mistral type AAM.

2.4.3 Evading a Missile

Developing an algorithm that calculates the optimal control to evade a missile is computationally difficult and time consuming [76]. It requires a thorough understanding of the missile sensory systems, navigation system, dynamics and warhead.

A sensor, such as a missile seeker, generates signals that contain errors. The sources of errors can be centroid wander, blind range, radome boresight error, gimbal limits and have field of view limits. The target aircraft, in view of a seeker, is not a single point but rather a large number of returns from which a centroid is calculated. The intensity from each return can vary significantly between scans leading to centroid wander. A radar has a minimum range and anything beyond it can no longer track the object. Usually this blind range is very small to allow effective evasion, but with the use of electronic jamming, this range can be substantially increased. Another source of error is when the missile rotates. The rotation causes the radome to move with respect to the seeker and the target. The curvature of the radome causes the refraction angle imposed on the seeker to the target line-of-sight to change. To the seeker, this appears as if the position of the target is changing.

The vulnerable areas in exploiting a guidance system of a missile are the guidance laws, homing loop resonance, and the response time constant. The guidance laws point the missile to the target based on the information provided by the sensors. Proportional navigation (PN) is the most widely used guidance system. One way to defeat a missile with PN is to take advantage of the lead compensation in the missile by flying in an oscillatory flight path. This causes the missile to change its lead angle substantially every time the target manoeuvres. Homing loop resonance is also another vulnerability that can be exploited. This vulnerability exists due to the missile response to aircraft movement. It can be modeled as a second order system [76]. This means the system should exhibit overshoot at some frequency input. The evasive tactic would be a periodic manoeuvre at the appropriate frequency to excite resonance in the missile guidance system. However, such a manoeuvre is hard to perform.

With reference to the dynamic aspect of a missile, the vulnerabilities can be; missile stability, turn performance and available energy. Missile stability covers a wide range of subsystems and stability characteristics which define how the missile responds to generated commands as well as external disturbances. Generally, the missile airframe is inherently unstable to enable a quick response. This has caused the design of the control system to be complicated, especially when it involves a wide range of conditions such as Mach number, angle of attack and centre of gravity. One evasive tactic is to get the missile into one of its lesser controllable states that reduces the capability of the missile to accurately steer towards the target. One common point of missile instability is high angle of attack. Thus, one evasive tactic is to drive the missile to experience a high angle of attack by performing a double jink or a large amplitude periodic manoeuvre. In relation to the turn performance aspect of the missile, the turn radius is a function of velocity and its normal acceleration limit. One appropriate evasive manoeuvre is to turn relative to the missile at a high-g load at a short time to go. With respect to energy available, if the missile is launched outside its effective envelope, the best evasive option is to run away and let the missile exhaust its fuel. Another option is weaving in order to drive the missile into a higher drag region to deplete the available energy of the missile faster. Such manoeuvres are best performed if the missile is launched from the fringes of the launch envelope.

The warhead of the missile plays an important part in ensuring its lethality. Since most missiles do not actually hit their target in order to cause damage, the warhead fusing and burst pattern characteristics of a missile can substantially influence the lethal effectiveness of a missile. The fuse is usually a side mounted sensor [76]. It determines the range of the target before deciding the effective distance for detonation. Once the effective range has been reached, the fuse delays the detonation for a period of time to allow the missile to reach the centroid of the target before it detonates the warhead. The lethal radius is a function of the position and orientation relative to the target aircraft. One effective manoeuvre is to position the aircraft strategically in order to minimise the damage due to the blast fragments.

According to Mandt [76], evasive tactics can be classified as follows:

1. Relative position - This is due to the difference in the performance characteristics of the missile and the aircraf. The position of the aircraft with respect to the missile may dictate the type of evasive manoeuvre which the aircraft should perform.

2. Time break - This is due to the missile response time and turn performance limits. The aircraft can perform a high-*g* turn causing a missile to miss. However, it requires timely manoeuvring to ensure a successful evasion.

3. Periodic manoeuvre - The purpose is to force the missile to overshoot the target aircraft by exciting a resonance in the missile's system. Common manoeuvres are weaves, multiple split-S, rolling scissors, flat scissors and barrel rolls.

2.5 Typical Air Combat Manoeuvres

In an air combat, a pilot will perform various manoeuvres depending on its objective, i.e. to pursue or to evade. There are many scenarios in air combat as described by Shaw [103]. The simplest is one-versus-one and the most complex is many-versus-many manoeuvring tactics.

Typical manoeuvres performed during an air combat are pursuit curves (lead, pure, lag), lag displacement rolls, high yo-yo, low yo-yo, lead turn, nose-to-nose and nose-to-tail turns, flat scissors, vertical and oblique turns, rolling scissors and defensive spirals [103]. In an encounter with missiles, finding the optimal evasive manoeuvres for an aircraft has been carried out by many researchers such as [3; 25; 82; 90; 132].

2.6 Aircraft Autonomy in Planning

A manned aircraft and an autonomous aircraft share the same threats and the same evasion strategies. The only difference is the intelligence that controls it. An aircraft is considered autonomous if it has the capability to plan its own actions using the acquired information about its surroundings and environment.

There are several levels of autonomy in an aircraft [95]. The first level is strategic planning. This is a low bandwidth planning in terms of computer processing effort. The tasks that are associated with this strategic planning are path planning, task allocation, search patterns and human mission command.

The second level of autonomy is tactical planning. This level of planning is considered to be medium bandwidth. Among others, the tasks are target observation, path following, communication and cooperation and human monitor interaction.

The third level is dynamics and control. This level of planning is of a high bandwidth. It requires a high degree of processing power. The tasks are state stabilisation, signal tracking, auto piloting and other inner loop controls.

In the pursuit-evasion problem, planning can be of any levels mentioned above as it is dependent on the nature of the problem. If the problem is about traveling from point A to point B with adversaries along the way [123; 134; 65], the planning will start from the first level. The planning involves the identification of static and dynamic threats such as tanks, mobile SAMs, radar sites and other obstacles i.e. buildings and mountains.

If the problem is between two players, the level of path planning will be at level two or three. In the literature, most optimisation algorithms cover only the first and second level. Optimal paths are created by these algorithms that allow the aircraft auto-pilot system to follow the best path as close as possible. For example, an optimal path found may consist of arcs and sharp turns. Due to this, an aircraft will definitely have to compensate with a finite turning radius to negotiate tight turns and arcs. There are several examples from the literature such as Kabamba [65] used a nonlinear programming technique to solve a path planning problem for unmanned aircraft in the presence of a radar-guided surface-to-air missile. A similar study was also conducted by [51] using calculus of variation.

Similar to this research, Price [97] studied singular perturbation theory to find an optimal intercept trajectory for an F-8 aircraft in real time. Instead of having the system built-in into the aircraft, he used flight director needles to display the commands to the pilot. The pilot's task was to keep both horizontal and vertical needles centred to remain on the optimal intercept trajectory.

In another study, a Fuzzy Inference System (FIS) was employed by [119] as a decision making tool to assist the pilot in air combat. The system accurately predicted the adversary's future move in the game. However, the mathematical model is simplified and it only used a 3DOF model. In this system, seven types of predefined manoeuvres were considered in the fuzzy system. They were represented by the maximum longitudinal, lateral and roll angle. These various combinations determine the type of manoeuvre performed by the aircraft. For example, a hard left turn is represented by rolling of -90° and performing a $9g$ turn.

In essence, the need to intelligently find optimal or feasible paths has led in the development of the solution for pursuit-evasion problem at higher dimensions.

2.7 Pursuit-Evasion Problem

Pursuit-evasion games can be any game that has two or more opposing parties. It can be as simple as a *rock-scissor-paper* game or as complex as an aircraft-vs-missile duel. Pursuit-evasion problems have been a very popular research topic for the past 60 years. Many researchers have been working on this problem albeit with simplified models.

The pursuit-evasion problem is still an open problem [30] because in the terminal phase the guidance solution is difficult to estimate due to imperfect information about the dynamics of the target and their intended evasive strategy. In such cases, attempts to use optimal control formulations are usually sub-optimal because the pursuer's command is always bounded. Thus, the state of the system cannot be fully and exactly observed. Furthermore, the target's intended evasion strategy is unknown [30].

In general, the study of pursuit-evasion in an air combat scenarios is basically duels between N missiles and M aircraft or N aircraft and M aircraft. To simulate a real case scenario, air combat manoeuvres are performed in three dimensions in which the aircraft is modeled in six degree of freedom equations of motion.

In most military environments, the pursuit-evasion game usually involves human pilots. Simulation performed are based on the pilot's cognitive skills and rely on their experience. Younger pilots tend to fly their aircraft based on the skills they have learned from the military flying school. Their judgements are easily reasoned and described.

But as they become more experienced, they tend to develop a unique set of skills that are purely based on intuition and judgement that they themselves cannot explain except saying “its from my experience”.

To described the pursuit-evasion problem in a mathematical form is even harder. Basically, the pursuer wants to intercept and capture the evader. The evader wants to avoid interception. A trajectory that could help the evader to avoid interception is accepted and classified in term of classical optimisation as being only sub-optimal.

Apparently, the actual pursuit-evasion problem is highly non-linear, full of noise, unpredictable atmospheric conditions and the manoeuvres of the vehicle are coupled. It is common to simplify a model in order to make the analysis mathematically tractable. Typical simplifications are:

- Linearising the model,
- Using a point-mass model,
- Restricting the manoeuvres into a two-dimensional common plane,
- Assuming factors such as constant velocity and constant weight.

Research on optimisation in pursuit-evasion problems between two aircraft has been done by many researchers such as [60; 50; 38; 61; 121; 55; 110; 117; 89; 7; 105]. Most these studies concentrate on the application of differential games (i.e. classical optimisation approach) in solving these simplified models. Besides the differential games, other suggested approaches are knowledge-based expert systems [78], heuristic value-driven system [70], discrete dynamic programming and influence diagram [127]. The problem of optimising trajectories of a missile was studied by [106] and [122].

Recently, missile guidance systems have largely employed on proportional navigation systems. Presently, modern guidance laws have been been using linearised kinematical models and linear quadratic optimal control formulations [122].

The drawback of these techniques are that they are only useful for non-maneuvering targets. Modern adversaries tend to use a much more robust control system. They may be highly maneuverable and highly unpredictable in their trajectory. The use of conventional navigation systems is useless against these intelligent targets. A feasibility study by [106] have shown that it is not feasible to use classical guidance techniques and estimation methods to “hit-to-kill” against highly maneuvering targets. One suggested solution is to view the problem using the differential game formulation that was made famous by Isaacs [59].

In another study, Turetsky [122] proposed the use of dynamic games to solve such scenarios. He said that intercepting a highly maneuverable target has to be formulated as a zero-sum pursuit-evasion game, in which, due to the unpredictable nature of the evader, an optimal control formulation is inappropriate. His analysis compared two guidance laws, i.e. linear quadratic differential game (LQDG) and norm differential games (NDG). To study this scenario, he used the following criteria or assumptions:-

- Simplified models are used;
- The engagement takes place in a two-dimensional plane;
- Both missiles have constant speeds and constrained lateral accelerations;
- The trajectories of both missiles can be linearised along the initial line of sight;
- The dynamics of both missiles are expressed by the first-order transfer function.

In another study, Green [107] analysed a horizontal pursuit-evasion game between a coasting pursuer with a final velocity constraint and a manoeuvring evader with constant speed. The proposed method was given an improvement by enlarging the capture radius in comparison with an identical evader guided by a traditional Proportional Navigation (PN) guidance system. This method was optimal against a non-optimal evader manoeuvre. The effects of gravity, altitude variations and other effects were neglected in the study. It is also assumed that the costate vector was continuous.

There was also a study by Horie [55] who used realistic dynamics of the aircraft and solved the pursuit-evasion in three-dimension numerically. He used the method of semi-direct collocation with nonlinear programming to solve the problem. The pursuer is superior to the evader. A three degrees of freedom (3DOF) aircraft model with realistic aerodynamic coefficients was employed. The effect of mass variation was neglected due to the short period of time of the duel. The time and the distance are normalised for ease of analysis and to avoid reducing the accuracy of the variables. This was done by keeping them in a similar order of magnitude. The predicted of an initial guess for a nonlinear programming solver is performed using a genetic algorithm. The results showed that the initial evasion strategy had a rapid change of direction (i.e. performing a hard turn) followed by vertical manoeuvring (i.e. diving).

In relation to this study, a Sequential Quadratic Programming (SQP) has been applied to solve pursuit-evasion problems by Ong [91]. 3D and 2D cases were studied using a generic air-to-air missile as the pursuer and an F-4 fighter aircraft as the evader. The pursuer is allowed to manoeuvre in three-dimensional space but the evader is restricted to the horizontal and vertical plane only. The various manoeuvres were represented by pull-ups, dives and turns.

However, these conditions restricted the study from being able to find the optimal solution for the evader in a complete three-dimensional space. This limitation is understood due to the difficulty to derive an appropriate mathematical solution for an SQP procedure. In this procedure, optimal manoeuvres were found to be hard turns, maximum-*g* pull-ups and maximum-*g* dives.

Apart from the studies mentioned earlier, another type is the motion planning using an incremental road map building algorithm within real time path planning that has been explored by [45]. The constraints in this study are both the fixed and moving obstacles.

Unlike [45], Campbell et. al. [19] successfully used several algorithms for real-time path planning and target tracking. The algorithms studied were square root sigma point filter, square root set-membership filter, robust nonlinear model predictive control, and streamline path planner. This study was extended into the validation of the model through flight tests. In addition, a simple pursuit problem was also studied. However, the model used in the study was a small and low performance 6DOF propeller-driven UAV.

Another approach to the study is the use of planar engagement that has been adequate for such an analysis and was demonstrated by Adler [1] who said if the trajectory linearisation is valid, the three-dimensional equations can be decoupled into two identical planar sets. In practice, many missiles have a cruciform configuration and two identical guidance operating in perpendicular planes [122]. The theorem requires the pursuer to have a sufficient manoeuvrable advantage in order to have a guaranteed capture.

Meanwhile, according to [106], one of the reasons why differential games had not been applied to missiles was the difficulty of implementing a realistic environment of noise-corrupted signals. Poor results have been obtained during the implementation of the optimal pursuer strategy of the perfect information games used in the pursuer's guidance law and using a typical estimator.

In order for a successful interception or evasion to take place, the players must implement optimal manoeuvring. Optimal manoeuvring as mentioned by Horie [55], as *a flight-path optimisation problem that can be treated as a one-sided optimisation problem (optimal control problem) or a two-sided optimisation problem*.

The one-sided optimisation considers only one player and this has been successfully applied in many applications [55]. Furthermore , this could be minimising or maximising the cost function of an aircraft, such as the minimum-time to climb problem.

In a duel, the one-sided optimisation approach would not be useful. This is because the optimisation will not take into account the manoeuvring of the other player. Thus the best optimisation for this particular type of game is two-sided or, as [33] labeled it "*saddle point problems*". The analysis must consider the optimal manoeuvres of both players, and thus a saddle-point solution is seek. This type of game obviously falls into the type of a zero-sum two-person differential game category.

A variety of zero-sum two-person differential games using two aircrafts have been solved analytically by using simplified dynamics. It is obvious that such a solution cannot be implemented for actual use. Guelman [50] solved this problem that uses a pursuer's but not the evader's dynamic characteristics. In general, to do an analytical study on such a problem using the actual aircraft data is very difficult. The answer is to solve the problems numerically. According to Horie [55], only a few studies have been done using this actual aircraft data because the optimisation of a realistic air combat scenario in which both aircrafts manoeuvre optimally is very challenging to solve, even if it is done numerically.

In another study, Tuomas [33] used nonlinear programming to study pursuit-evasion games of degree between a missile and an aircraft in the terminal phase. The capture is guaranteed but the time of capture has yet to be identified. In this approach, a maxmin problem is decomposed into subproblems and solved iteratively.

Jarmark [61] studied an air combat manoeuvring problem in a horizontal plane with steady turns and using realistic dynamic and engine data. He used a modified differential dynamic programming (DDP) method for solving general optimal open-loop differential games. The approach used fixed final time as a game parameter. The study found that a good throttle control and the ability to accelerate and decelerate are important for making a hard turn.

Most research on the pursuit-evasion problem between two air vehicles utilises point-mass equations of motion for reasons of simplicity. The author has not yet found any research that is close to this scope of study that uses six degree of freedom equations of motion to find an optimal solution for the evader using any optimisation method. In addition, there is a difficulty of trying to model the problem into an appropriate form to reduce the high-dimensionality involved or "curse of dimensionality" [40].

This is especially true if the problem is finding optimal control solutions for an aircraft in which the control is represented by the deflection angle of the aileron (δ_a), elevator (δ_e), rudder (δ_r) and throttle setting (δ_T). If each variable has p_a, p_e, p_r and p_T possibilities and at every δt seconds the variables change, and the game is played for t_f seconds. The number of possible combination of solutions is

$$p_a^{\frac{t_f}{\delta t}} \cdot p_b^{\frac{t_f}{\delta t}} \cdot p_c^{\frac{t_f}{\delta t}} \cdot p_d^{\frac{t_f}{\delta t}}. \tag{2.1}$$

The complexity level is O(n^m) where n is the number of possible control surface combinations and $m > 1$ as the number of changes of control surface deflections. For example, the number of possible control surface combinations, n, is 10,000 and for a 40 second encounter, m is 40.

It can be astronomically high even for a moderate range of control surfaces deflections. Thus, solving the problem through classical methods can be prohibitive. A new method that is able to find the optimal solution in a noisy environment which is full of uncertainties as well as being robust, reliable and fast is required.

The need for fast intelligent systems on board an aircraft is crucial as the technology becomes more sophisticated and cheaper. The workload of a pilot seems to increase as new aircrafts used in the service are equipped with new and complicated weapon systems. A pilot may not be able to think of everything during a tensed situation and the help of an intelligent agent would undoubtedly be very beneficial during an air combat.

A pilot in an air combat can be in either one of these two positions: (1) at the advantage position to shoot or (2) at the disadvantage, where the other pilot has the opportunity to shoot. Most research concentrates on how to achieve the former condition (1), but very few are studying the latter, such as [82], which explores the evasion methods from these missiles. For example, Moore [82] employed genetic programming algorithms to find the optimal trajectory of an F-16C aircraft against missiles in extended two dimensions.

The assumption that an evader is already inside the capture zone of the pursuer is considered by many in the literature [33; 108]. Being in the zone, guarantees that the evader will be captured. Apart from that, linearisation of the equation of motion of the vehicle is widely practised. However, one question arises; how about optimising the evader outside the capture zone? Is it possible for one to optimise the trajectory of the evader before it enters the capture zone? Such questions are difficult to answer because of the nature of the problem is highly nonlinear and non-differentiable. Furthermore, it is not known if a much better solution can be found by initially behaving in an inferior way. It is similar to a game of chess. Sometimes, one needs to sacrifice a piece to get a better overall position. Unfortunately, most optimisation techniques refuse to follow such path. Perhaps, this is due to its inability to see the "other side of the hill". In an air combat scenario, this could be translated in to a manoeuvre that looks like a sacrifial move, but, being at a correct position, having good timing, using the correct manoeuvre could drive the aircraft toward its objective. For example, the evader could slow down to allow the missile to come close and then bank away at the last moment.

Long computing time is one of the major problems posed by many of these optimisation techniques. Even if the technique is able to find an optimal solution, the computing time can be impractically long which renders it useless for real time applications.

As the problem becomes more complex, it takes a longer time to reach the optimal solution. The computing time is rarely mentioned while searching for the optimal solution in the literature. Thus, a definitive comparison between algorithms cannot be made on the aspect of speed in their effort to find the optimal solution.

Another aspect of optimisation algorithms is that most of these classical approaches require the problem to be modeled into a form that is understood by the optimisation method. In doing so, many assumptions have to be made so that the problem is tractable, i.e. differentiable, no singularity, the initial guess values must be correct so that the optimisation algorithm would converge into the right solution [58; 24]. This will definitely reduce the level of realism of the problem.

Instead of trying to redefine the pursuit-evasion problem into a form recognised by the optimisation algorithm, the idea is to apply the optimisation algorithm to the pursuit-evasion problem in its natural state-space form without any simplification and modification. In fact, there is one algorithm found to be suitable i.e. the Evolutionary Algorithm (EA) for this purpose. Chapter 3 explains in greater detail about the EA and Evolutionary Programming (EP). This approach can be considered as a behavioral search [94] but with an agent model of a higher complexity.

The pursuit-evasion problem described in this thesis falls into the category of "games of kind". The optimisation algorithm strives to find a trajectory could guide the evader from being intercepted or otherwise. A feasible solution is the one that will save the evader from interception.

2.8 Evolutionary Algorithms in Pursuit-Evasion Problems

Evolution is an optimisation process [77; 63]. The selection strategically removes suboptimal solutions, also known as individuals or phenotypes. A comprehensive review on evolution itself is available from [63]. This leads to the idea of using computers to emulate evolution. The idea is to generate machine learning through simulated evolution [41]. Amongst earlier attempts to apply evolutionary theory in practical engineering problems appeared in the areas of statistical process control, machine learning, and function optimisation [63].

Evolutionary algorithms (EAs) have been recognised by the research community as one of the new paradigms to approach or resolve the optimisation problem. The classical optimisation methods have difficulty trying to solve this problem because it is either trapped in the local minima or maxima.

Recognising this fact, the idea of using EAs in pursuing an optimal trajectory for a pursuit-evasion problem has begun to flourish in the research community. Some problems are best solved using running simulations, such as pursuit-evasion problems. The problems can be more complicated in stochastic environments, i.e. when the objectives involve uncertainties or noisy value due to the presence of random variables. To make the problem even more complicated, the problem could have multiple objectives, such as "black-box" objective functions that could cause classical numerical optimisation approach to be computationally expensive [36]. This optimisation approach is also known as "simulation optimisation" [36].

A study by Mulgund [83] used GA to optimise the initial launch of a beyond visual range missile in a many-vs-many engagement. Although point-mass models were used, the study did not cover the optimisation of a “close-range” duel. The study mainly concentrated on the optimal formation of the attacking units.

The robustness of Evolutionary Algorithms has provided the lead for the research community to find an optimal solution in a dynamic environment. Finding an optimal solution in a dynamic environment is not easy, but if one could find a solution that could follow the optimal solution closely is already good enough [4]. He found that the use of self-adaptation was found to be effective on some functions, but detrimental to others. These effects validated the “no-free-lunch theorem” [131]. Thus, a deeper understanding of the problem at hand is essential during the development of an EA algorithm.

Another example is by [72], where the author used a genetic algorithm and a simulated annealing algorithm to search for optimal evasive manoeuvres in the vertical plane. The evader is an aircraft and the pursuer is a missile. The author used genetic algorithms to search for global optimal solution and the fine tuning of the search space (local search) were performed using simulated annealing.

Algorithms based on evolutionary principles are: Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary Strategies (ES) and Evolutionary Programming (EP). All these fall into one main algorithm type called Evolutionary Algorithms (EAs). The metaphor underlying EAs is an approach that is based on natural selection. The idea is borrowed from Darwin’s principle of “survival of the fittest” [28]. Fogel [42] conducted an indepth analysis of evolutionary computation and its prospect as a new algorithm for artificial intelligence.

A genetically-based machine for learning for air combat manoeuvre has been studied by [113; 114; 115]. The learning classifier system is a Q-learning based learning technique that uses pairs of state/action encoded by GA to generate optimal trajectory or manoeuvre. For a period of 30 seconds, through a simulated game, the GA searches the optimal manoeuvre for either one-player or both-player. The state/action pair technique executed is similar to the expert system where many IF-THEN-ELSE conditions are employed. Similarly with this thesis, the author uses a nonlinear 6DOF aircraft model to perform the simulation to study post-stall technology (X-31 experimental aircraft) in an air-combat. The other aircraft is an F-18. Several ingenious manoeuvres have been performed by the X-31 aircraft to achieve an upper hand against the other aircraft. Some of the manoeuvres found by GA are: the Herbst Manoeuvre and the PST (Post Stall Technology) Hammerhead turn. The Herbst Manoeuvre causes the aircraft to quickly reverse through a combination of high angle-of-attack and rolling. In the PST Hammerhead turn, the aircraft reverses its direction by performing a manoeuvre resembling a backflip.

On the other hand, the current application of Evolutionary Algorithms (EAs) in pursuit-evasion problems is limited in terms of its role. In many studies, EAs are utilised to find good initial guess values for use in other optimisation algorithm such as nonlinear programming as examined by Horie [55]. He solved the optimal solution in pursuit-evasion problem by utilising conventional collocation with nonlinear programming. A genetic algorithm is used to provide an approximate solution (i.e. initial guess) for the nonlinear programming solver. Others have also attempted to use the Evolutionary Algorithm in simplified pursuit-evasion problems [71; 7; 32].

The dynamic nature of a pursuit-evasion problem has put Evolutionary Algorithm in the spotlight due to its ability to locate a close-to-optimal or feasible solutions in a relatively short period of time. With the advent of fast computing technologies, such as parallel computing architectures, the use of Evolutionary Algorithms becomes even more promising. This can be done by performing the search from a remote position using a very powerful computer and sending the results back to the aircraft in the manner that is as close to real-time. Thus, the aircraft can be kept lightweight, simple and inexpensive.

The search for optimal mutation rates, crossover rates and many other parameter optimisation has been carried out in many studies using standard objective functions. For example, Yong [73] introduced fast-GA with exclusion-based selection operators and Fourier series auxiliary functions to improve the performance of a standard GA for multi-model problems.

A more general pursuit-evasion problem between agents such as robots has been studied by [46; 22; 88]. Fukunaga [46] applied Genetic Programming (GP) in planar pursuit-evasion game by co-evolving the players' strategies. The performance of the GP algorithm was improved by using increment evolution. A relationship between parameters was observed while trying to improve the performance of the algorithm. This may be due to the large number of parameters involved and the stochastic behaviour displayed. The pursuit-evasion between a team of pursuers and a team of evaders are also studied. For example, Nitschke [88] utilised co-evolution of genetic algorithms to find the optimal pursuit-evasion between a cooperative pursuer and a non-cooperative evader. The agents used were robots. It was observed that the tuning of parameters for optimal performance is still a problem in itself [46; 88].

A point-mass pursuit-evasion problem using genetic programming (GP) was solved by Moore and Garcia [82]. The evader was a generic jet fighter aircraft and the pursuer was an air-to-air missile. In their approach, each optimal solution found by GP was only optimised for a specific type of pursuer. A fixed set of possible manoeuvres were used in the coding schemes and in the solution space.

This thesis aims to find feasible solutions for the evader in a pursuit-evasion problem through Evolutionary Programming (EP) algorithm. The pursuit-evasion problem has to be in three-dimensions, using either a six degree of freedom model (6DOF) or 3 degree of freedom model (3DOF). The aircraft data has been obtained from actual flight-test and wind-tunnel observations. The EP algorithm must be able to find the optimal control history of the evader to be as close as possible to real time. As far as the author knows, there is no such research has been attempted that use a full 6DOF model for the evader and either 6DOF or 3DOF model for the pursuer using EP.

2.9 Parallel Evolutionary Algorithms

EAs are inherently well suited to be parallelised. The advantages of using parallel EAs are numerous [21]. Two important aims in utilising these parallel EAs are speed and quality [21]. Firstly, the aim is to reduce the execution time so that the feasible solution can be obtained faster. Secondly, the aim is to increase the quality of the feasible solution, i.e. within the same computing time, more solutions can be evaluated by distributing them to several processors which translates them for a broader search space. For that matter, the execution time has two components: the time used for computation and the time used to communicate information among processors. There are four models commonly used in parallel EAs as outlined by Cheang [24]:

Master-slave. This is also known as a global model. In this model, a master process assigns individuals to different slave processors for fitness evaluations.

Multiple-demes or Coarse-grained [21]. In this model, a complete EA cycles are performed by several demes. Individuals are interchanging periodically.

Cellular model. [64] In this model, each individual is assigned to a cell on a multi-dimensional grid. Each cell can only interact with its adjacent neighbours.

Cooperative co-evolution model [96] In this model, an individual is divided into sub-components. Each sub-component evolves into an isolated subpopulation. A complete individual is obtained by assembling representatives (i.e. best individual of sub-components) from each of the sub-population.

For the purpose of this research, only two models are used. They are the master-slave and coarse-grained (multiple-demes) models.

2.10 Summary

This chapter details the various approaches adopted in solving the pursuit-evasion problem. The first approach was called the classical approach. The classical approach requires the problem to be defined in the form of differentiable equations. If the problem is differentiable, the saddle point can be searched.

The second approach is to use numerical modelling such as a nonlinear programming technique to search for an optimal point [99; 33]. One of the ways is to decompose the problem into ordinary optimal control problems which are solved using discretisation and nonlinear programming techniques. Another methods are to discretise the game and transform it into a bilevel programming problem and solve using a first order feasible direction method.

A new optimisation approach has been suggested, i.e. Evolutionary Programming. This can be used to overcome the limitations of the classical techniques. The following Chapter 3 will describe the Evolutionary Programming in greater depth. EP can be further optimised by executing it using parallel computers. Thus, in this thesis, the EP will be developed as a Parallel EP which is explained further in Chapter 4.

3 Evolutionary Computation

3.1 Introduction

Evolutionary Computation (EC) is a field of study that is inspired by natural evolution and adaptation [6]. The development of EC was initiated to solve complex real world problems through robust and efficient computational systems. In contrast to EC, traditional computational methods are difficult to implement for such problems because their objective function is usually mathematically undifferentiable, highly nonlinear, noisy, or has many singular surfaces of various types. However, EC is able to approach the problem in its natural form. Thus, EC is suitable to find optimal or close to optimal solutions for complex real-world problems. The way EC approaches the problem is by using a population-based approach that relies on the power of random variation and selection. Algorithms that are based on evolutionary principles are called Evolutionary Algorithms (EAs). Commonly found EAs are Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary Strategies (ESs) and Evolutionary Programming (EP).

The key element in EAs is that the solution of the problem is represented in a population of individuals. An individual (a solution) represents the genotypes of the solution space. The quality of the solution is improved by altering the genotypes (genetic structure) of the individuals. The improvement is also called evolution (deriving a better solution). Such an approach is so unique because EAs create a virtual world for the population to breed, interact and die under certain rules. To do so, the four algorithms used in EA share similar properties [24] and they fall into this category:

1. EAs approach collectively the learning process among individuals. Each individual represents a search point in the solution space.

2. The offsprings of individuals are generated stochastically, in the case of Evolutionary Programming, through mutation.

3. An individual is evaluated at each life cycle. So each individual will have

a fitness value that represents their ability to survive. The selection will be based on these fitness values.

In essence, evolutionary computation can be seen as a "generate and simulate" approach in which multiple candidate solutions are generated and tested simultaneously. The propagation of the candidate solutions is moderated by the selection pressure and the method of "intelligent" reproduction that would govern the distribution of the solution throughout the population.

In a dynamic environment, it is expected that the optimisation cycle is repeated at every adaptation cycle. This is true when the current executed plan experiences new threats or changes in the environment. Although the feasible solution is searched through a "perfect and virtual" experience in a simulated world, it is hoped that it will perform well and similar success rate will be obtained when exercised in the real environment. As an optimisation solver, the problem at hand must be presented in a framework amenable to derive a solution by the "generate and simulate" process. In doing so, the evaluation of the solutions require an objective function that could measure the performance and sort them accordingly.

In many real world problems, near-optimal or feasible solutions are usually sufficient to resolve the problem. For example, finding an absolute evasion solution for 6DOF model in a given time span can be very difficult and time consuming. On the other hand, it is adequate to find a feasible solution that could satisfy all the constraints and still guide the evader from being intercepted. Even the first feasible solution found during the optimisation cycle is adequate if the objective function is robust and suitable enough to be extended to the real world.

3.2 Evolutionary Programming Algorithm

The Evolutionary Programming (EP) method was pioneered by L.J. Fogel in the early sixties. Optimising continuous parameters using evolutionary programming has many similarities with evolution strategies explicating how mutations are normally distributed. According to Fogel [43], evolutionary programming can be categorised into five variants:

- Standard EP is characterised by the absence of any self-adaptation mechanism.

- Continuous standard EP is where a newly created individual is immediately evaluated and inserted into the population.
- Meta-EP incorporates variances into the genotype in order to allow for their self-adaptation.
- Continuous meta-EP is similar with Continuous standard EP where a newly create individual is immediately evaluated and inserted into the population.
- Rmeta-EP is similar to meta-EP but it also incorporating standard covariances.

In EP, an initially random population of trial solutions is created. Mutations are then applied to each individual to create new individuals. Mutations vary in the severity of their effect on the behaviour of the individuals. The new individuals are then compared in a "tournament" to select which should survive to form the new population.

EP [5] is a class of EAs for simulating evolution. This is done by utilising the concepts of Darwinian evolution theory to iteratively generate increasingly better solutions in a dynamically changing environment. This process differs from Artificial Intelligence which requires human experts to define the set of rules. On the contrary, EP finds the solution by evolving sets of solutions which exhibit optimal behavior with regard to the environment of the problem and obtaining the desired payoff.

EP is also similar to Evolutionary Strategies, ES, although the two approaches were developed independently. In EP, selection is by comparison with a randomly chosen set of other individuals whereas ES typically uses deterministic selection in which the worst individuals are purged from the population. Both rely on a multi-agent stochastic search algorithm that can be used to find the optimal solution of the given functions. The functions are usually non-linear and consisting of many local sub-optimal solutions. In this case, the conventional gradient based search algorithms are unsuitable because the solutions can be trapped in a local sub-optimum or the computation of the gradient itself is very difficult to obtain [100].

Mathematically, the algorithm is, as described by [6]:

1. *Initialisation*: An initial population of μ individuals is generated and the generation count, g, is set to 1. Each individual in the population is represented as a pair of real or integer valued vectors, $(x_i, \sigma_i), i = 1, 2, \ldots, \mu$.

The x_i's represent the individual input vector and σ_i's represent the associated standard deviations. Each individual is evaluated using the objective function, f.

2. *Mutation*: Each parent $(x_i, \sigma_i), i = 1, 2, \ldots, \mu$ produces a single intermediate offspring (x_i', σ_i') according to

$$\sigma_i' = \sigma_i(j) exp(\tau' N(0, 1) + \tau N_j(0, 1)) \tag{3.1}$$

$$x_i' = x_i + \sigma_i'(j) N_j(0, 1) \tag{3.2}$$

where $N(0, 1)$ is a normally distributed random number with zero mean and unit variance. $N_j(0, 1)$ is similar with $N(0, 1)$ but is regenerated for every j. τ and τ' are generally being set to $\frac{1}{(\sqrt{s\sqrt{n}})}$ and $\frac{1}{(2\sqrt{n})}$, respectively. [100] uses lower threshold level of 10^{-4} of the standard deviation.

3. *Objective function value*: The objective function value of the offsprings (individuals) is computed as $f(x_i', i = 1, 2, \ldots, \mu)$. They are combined with the parent with size μ to create the total population of 2μ.

4. *Tournament*: Every individual in the population is compared with q randomly selected opponents. For each comparison, the individual that a lower or equal fitness value will receive a win.

5. *Selection*: The better half (size μ) of the population (size 2μ) is selected to be the parent for the next generation.

6. *Termination*: If the generation count, g reaches g_{max}, the maximum number of generations, the evolution is terminated. Else, g is incremented by one and the algorithm loops back to step 2.

Basically, in its standard form, the basic evolutionary program utilises four main components i.e. initialisation, variation, evaluation or scoring and selection.

Initialisation means creating a set of possible solutions stochastically. In order to apply EP successfully, an intelligent problem-specific solution must be devised [63].

Variation provides the means for moving the solutions within the search space. This could prevent solutions from being trapped in local optima. Evaluation will measure the fitness of each solution and gives each solution its own weight or fitness value. This fitness value is used as the basis for the selection of better solutions.

The selection process probabilistically culls suboptimal solutions from the population. This provides an efficient method for searching the topography. However, some suboptimal solutions are kept in the population in hoping that they will be able to explore the solution space and locate the optimal solution.

The population size may range over a broadly distributed set. In general, it should be larger than one solution. Each of these solutions is evaluated using a fitness function. After the creation of the population of initial solutions, each of the parent members is mutated to produce an offspring. However, recombination is not utilised in EP.

Later on, the fitness is assessed for all offspring solutions with (i) the best solutions are retained to become the parents for the next generation also known as elitism; (ii) some of the best solutions are statistically retained (through tournament); (iii) the solutions are selected using proportional-based selection.

Finally, the process is terminated when a specified number of iterations has been achieved, or a sufficient convergence criteria is reached.

3.3 Representation

Representations of the underlying problem are important for every search and optimisation evolutionary algorithms. They represent the solutions of the underlying problem. In most engineering problems, a solution is a real-value vector used to unlock the given problem.

Depending on the problem at hand, a solution can be a time- or frequency-dependent functional variation for control system problems. The solution could also be a strategy for games. Thus, the knowledge of the underlying problem is important in devising the representation of its solution [5]. This is because it affects the efficiency and complexity of the search algorithm. Obviously there is no strict guideline on how a solution of a problem should be represented, but rather it is more by the experience of the algorithm designers.

There are generally two approaches to represent information inside individuals in a population. In the EA community, two well known approaches are the Pitt approach and the Michigan approach.

The Pitt approach is from the work of De Jong and his students from the University of Pittsburgh [29]. In this approach, each individual in the population is encoded with all parameters of a possible solution.

On the other hand, the Michigan approach defines how the individual represents the parameter. In this approach, each member of the population representing a single or a subset of parameters and the entire population forming a complete solution of the problem.

In this thesis, the Pitt approach is used to code parameters values in individuals. For the pursuit-evasion problem, the evader's control time history (i.e. the solution) is coded inside each individual solution representing the entire rule set so that each generation consists of a population of possible solutions. The development of the optimisation process is discussed in Chapter 5.

3.4 Evaluation of Solutions

The fitness of an individual or a solution is measured through a function commonly known as the objective function. The derivation of the objective function is highly dependent on the problem at hand. In a pursuit-evasion problem, the objective function is evaluated based on the simulated pursuit-evasion responses. A proper air combat simulation model has to be built to evaluate the solutions.

In the related literature, issues due to the difficulty in developing a 6DOF model are solved by using a simpler 3DOF model, such in [61], [98], [38], [57] and [58]. The use of a full 6DOF model in such a problem is still in its early stage.

In EP, the outcome of the game forms the basis of the fitness value. Even the infeasible solutions can be regarded as useful. The fitness value has to be properly scaled and valued. For example, if all solutions are infeasible at the early generation, they are not scrapped but the one that gives the maximum capture time and is less likely to violate the constraints will be given a better fitness value. The better solutions will be selected and mutated accordingly to form the offsprings.

3.5 Mutation

The mutation provides the diversity of the solution in EP. In the case of standard Evolutionary Programming as derived from the work of [10], the Gaussian mutation operator $m_{\{\beta_1,\ldots,\beta_n,\gamma_1,\ldots,\gamma_n\}} : I^\gamma \rightarrow I$ is an asexual operator, such that again only the reduced from $m'_{\{\beta_1,\ldots,\beta_n,\gamma_1,\ldots,\gamma_n\}} : I \to I, m'(\vec{x})$ is used. The operator works with a standard deviation that is obtained from each component x_i of the object variable vector as the square root of a linear transformation of the fitness value $\Phi(\vec{x})$, that is $(\forall i \in \{1, \ldots, n\})$:

$$x_i' = x_i + \sqrt{\beta_i \cdot \Phi(\vec{x}) + \gamma_i \cdot N_i(0,1)} \tag{3.3}$$

The proportionality constants β_i and offset values γ_i are $2n$ independent parameters that must be tuned. Usually, the value of β_i and γ_i is set to one and zero, respectively. Thus equation 3.3 becomes

$$x_i' = x_i + \sqrt{\Phi(\vec{x})} \cdot N_i(0,1) \tag{3.4}$$

A more intelligent approach is to make the EP self-adapting which is known as the meta-EP. This is achieved by mutating the variance as follows; mutation $m_{\{\zeta\}}$ applied to an individual $\vec{a} = (\vec{x}, \vec{v})$ produces $(\vec{x}^1, \vec{v}^1)$:

$$x_i' = x_i + \sqrt{v_i} \cdot N_i(0,1) v_i' = v_i + \sqrt{\zeta v_i} \cdot N_i(0,1) \tag{3.5}$$

where ζ denotes an exogenous parameter to ensure that v_i tends to remain positive [10] and $\forall i \in \{1, \ldots, n\}$.

The benefit of using various mutation strategies such as Gaussian, Cauchy, Lévy and others has been studied in the past. However, they were found to perform better at certain types of problems but did not fare well at others. For example, in conventional evolutionary programming, the Gaussian mutation strategy is widely used, but it does not perform well for multi-modal functions.

3.6 Recombination

As a note, Evolutionary Programming does not use any kind of recombination operator but relies on the power of mutation alone. Fogel argues that the role of a crossover is often overemphasized by claiming that it is inappropriate to define evolution by the mechanisms of genetic change rather than their phenotypic effects. It is supported by Atmar [44] who claimed that crossover should be no more than a second- or third-order component of evolution. The author claims that the benefits of using crossover are not clear.

3.7 Selection

The offspring population is directly derived from the parent. So, their sizes are identical, i.e. $\gamma = \mu$. A probability selection method is used on the union of parents and offsprings to reduce the total number of individual from 2μ to μ. This selection method is also called $(\mu + \mu)-$ selection.

The selection is controlled by an additional parameter $q \in \mathbb{N}(q \geq 1)$. q denotes the tournament size. For each individual $\vec{a}_k(k \in \{1, \ldots, 2\mu\})$, q individuals are chosen at random from the parent population, $P(t)$ and the offspring population $P'(t)$ or $P(t) \cup P'(t)$, and compared to $\vec{a}(k)$ with respect to the fitness values.

The number of individuals in q that are worse than $\vec{a}(k)$ are counted and scored, $w_q \in \{0, \ldots, q\}$. The 2μ individuals are then ranked according to their score values, w_i, and μ individuals with the highest score are selected to form the next population. The calculation of w_i is as suggested by [10].

In this selection process, there is a difference between stochastic and deterministic selection [63]. The use of deterministic selection guarantees that the best individual in the population will always get selected. Stochastic selection is employed to provide "noise". The motive is to improve their "robustness" by decreasing the probability of converging to a sub-optimal solution. The balance between exploration and exploitation is necessary to have a good level of "greediness" so that the selection pressure is able to locate optimal solutions and at the same time diversified the selection space to avoid being trapped in sub-optimal peak. The search for the optimal setting between exploration and exploitation is by trial and error. This may result in the loss of feasible solutions, but the advantage should not be ignored.

In another study, Jianjun [56] suggested an approach to overcome limitations such as premature convergence, stagnation, loss of diversity, lack of reliablity and efficiency and oversimplified of the objective function experienced by conventional evolutionary algorithm.

During the selection, when the population fitness increases, it will become harder to escape from the existing search vector. The new approach is called Hierarchical Fair Competition (MFC) model. The model is achieved by maintaining individuals in a hierarchically organised fitness levels. The evolution is kept running at all fitness levels. This transforms the EA from a conventional convergent evolutionary computation model into a sustainable search framework by culturing and maintaining building blocks of various fitness scales.

The thesis employs both the elitist selection and the tournament selection mechanism in selecting individuals for the next population for comparison purposes. It can be specified in the configuration file prior to the simulation.

3.8 Applying to Pursuit-Evasion Problem

Literature related to the current approaches in solving pursuit-evasion problems has been described in Chapter 2. Simplifications were necessary as to make the problem mathematically tractable to suit the techniques. But it comes with a price, and that is, it reduces the level of realism. Thus, a new optimisation paradigm is required to fill the gap between numerical simulation and reality. Evolutionary computations appear to be able to fill in the gap due to its ability to solve complex real-world problems [24] and one of these examples is solving the optimisation in power systems planning and operation [52].

In an air combat, the evading pilot would have to know the position of the adversary and the type of weapons they are carrying. By assuming the pilot knows the exact location where the adversary fired its missile and the type of missile it uses, the pilot then will steer the aircraft for evasive manoeuvre by moving the control stick and the throttle. If the application of the control stick and the throttle is done in a timely and correct manner, successful evasion will be achieved. The timely manner means that the timing of the manoeuvre must be perfect whereas the correct manner means the type of manoeuvres such as hard turn or climb must be correct. If these criteria are not followed, the evader will be intercepted. Thus, the ability to locate the evasive window of opportunity is crucial in such a scenario. To achieve this, the pilot must know the tricks and skills to perform the evasive manoeuvres [103]. The types of manoeuvres have been discussed in Sec. 2.5. However, such tricks and skills are not easily mimicked in a computer environment. The computer must have an inherent ability to perform the manoeuvres correctly.

As mentioned earlier, many researchers use 3DOF model to determine optimal trajectories in a pursuit-evasion problem. But their solutions are limited to the terminal phase (the end game) and with the assumption that the capture is guaranteed. There is hardly any literature that talked about exploiting the pursuer's weaknesses to the fullest for a successful evasion. By reducing the equations of motion to a simplified form, a lot of the important information is lost. This loss of information could result in an incorrect conclusions.

In other research, very few researchers are actually applying a 6DOF model with actual aerodynamic data to solve optimal trajectory in an air combat scenario. Similarly, the relationship between the medium phase and the terminal phase of the game has also not been studied. Perhaps, this is due to the difficulty arising from the mathematical analysis.

In order to successfully use the EP to solve an air combat problem, the idea is to represent the solution as a series of deflections of the aileron, the elevator, the rudder and the throttle settings for a fixed period of time. The solution is evaluated in a game of an air combat that has taken place in a three-dimensional space. An optimal solution is the solution that helps the evader from being intercepted. The solution is the one that has the highest fitness value. If the solution caused the aircraft to violate constraints or causes it to be intercepted, then would be considered sub-optimal. A penalty is given that will reduce its fitness value. If the game is not over yet, the search for the next optimal solution is repeated for another spawn time.

In its general form, the optimisation approach described in this thesis can be interpreted as a *black box* optimisation problem. One may argue that the approach lacks a tangible mathematical approach and does not have a thorough theoretical background. But the potential is considerable [21].

Apart from computational time, if the procedure is properly designed and described, there are no restrictions on the type of inputs that can be fed into the black box. From the optimiser's perspective, EP sends a set of parameters into the black box, and in return, it will receive a scalar performance measure, i.e. the fitness value.

From a pursuit-evasion perspective, this means that the objective function can be based on the full nonlinear flight response characteristics of the players. These include realistic representations of uncertainty, discontinuities, disturbances and noise of sensors and the environment.

The main computational results of this thesis focus on the ability of the evolutionary programming method to find optimal solutions without any prior knowledge. This is achieved by performing evolutionary search for control parameters through numerous flight simulations which will then be used to assess the performance of possible solutions.

3.9 EAs in Uncertain Environments

EAs is definitely an uncertain optimisation algorithm. Even if the mathematical models are very close to real, the high dependence on random number generator itself guarantees the presence of uncertainties in its ability to search for optimal regions. This section was written primarily based on the work of [62]. As outlined by [62], there are four classes of uncertainties affecting the performance of EAs such as:

Noise Noise exists during the evaluation of the fitness function. They may come from different sources. In an air combat game, the noise may come from sensory noise, measurement errors and estimators. Mathematically, a noisy fitness function is represented as:

$$F(X) = \int_{-\infty}^{\infty} [f(X) + z]\, p(z) dz = f(X), z \sim N(0, \sigma^2) \qquad (3.6)$$

where X is state or design variables, $f(X)$ is the fitness function, z is the additive noise, which commonly assumed to be normally distributed with zero mean and variance σ^2. It has been observed that there is no significant difference between a Gaussian and a non-Gaussian noise distribution [62].

Robustness The state variables are subjected to perturbations after the optimal solution is found. Thus, it is important to have a high confidence that the optimal solution will still work. Monte Carlo simulation is commonly applied to estimate the robustness. Commonly, robustness is viewed from two aspects [62]:

1. The optimal solution is insensitive to small variations of the design or control variables (e.g. the aileron, the elevator, the rudder and the throttle setting); and
2. The optimal solution is insensitive to small variations of the environmental and states variables.

Fitness Approximation An approximation of the fitness function is usually used when the original fitness function is very expensive to evaluate. For example, evaluating a solution on a CFD[1] solver which may take hours to converge. Thus, using an approximate fitness function may save time and cost.

[1]Computational Fluid Dynamic

Approximation is typically not required if the fitness function is relatively fast to evaluate.

Time-variant Fitness Function The fitness function is highly dependent on time because the optimal solution moves along with time. This is particularly true for optimising trajectory in an air combat. In an air combat, the initial state variables and control variables will determine the optimal solution. Thus, as the game goes on, the optimal solution has to be computed for a defined spawn-time, see Section 5.1.

3.10 Summary

This chapter explains the methodology of Evolutionary Programming algorithm. This algorithm forms the general outline of the optimisation algorithm discussed in Chapter 5.

The next chapter explores further the algorithm by explaining the application of parallel Evolutionary Programming algorithm to solve pursuit-evasion problem. It is important to find the correct representation of the solution to the problem so that the solution could be easily evaluated and always ready to be applied to the actual problem.

4 Parallel Computing

4.1 Introduction

Evolutionary Programming (EP) or any evolutionary algorithm requires hundreds or thousands of function evaluations of solutions. Each evaluation may take seconds or days to be resolved, depending on the cost of the evaluation. Fortunately, the evaluation of a solution in EP is independent of each other, making it possible to distribute the computing load among multiple processors. This approach will speedup the optimisation process [63]. Parallel Evolutionary Algorithms (EAs) are considered by some as “embarrassingly parallel” programs [20]. But, although their mechanics are simple, the underlying algorithms are complex and controlled by many parameters that affect the quality and efficiency of their search. Generally, the design of parallel EP involves choices such as using one population or multiple populations. The choice of the size of the population(s) must be carefully determined in order to get the best speedup and the quality of the solution.

4.2 Parallel Systems

A parallel computing system is a computer that has more than one processor and it is used for parallel processing. In the past, each processor of a multiprocessing system always came in its own processor packaging and design. In recent developments, multicore processors are introduced. It contains multiple processors in a single box. Although, there are many different kinds of parallel computers. They are distinguished by the kind of interconnection between processors (known as “processing elements” or PEs) and the way the memory is distributed and shared. On a historical note, the application of parallel computing on evolutionary algorithms was probably first used by Bethke [11].

Parallel architectures are synonymous with Flynn’s taxonomy. It is one of the most accepted taxonomies of parallel architectures, classifies parallel as well as serial computers based on the following criteria:

- All processors execute the same instructions at the same time (single instruction/multiple data – SIMD) or
- Each processor executes different instructions (multiple instruction/multiple data – MIMD).

These processors are further classified into two main categories based on memory access times which are:

- Uniform Memory Access (UMA), in which access times to all parts of the memory are equal, or
- Non-Uniform Memory Access (NUMA), in which in which access times to all parts of the memory are not equal. Distributed memory parallel computers also have multiple processors, but each of the processors can only access its own local memory; no global memory address space exists across them.

Another common classification is based on the number of processors a system has. If the number of processors comes in thousands, the system is known as massively parallel. A PC based parallel system is generally called a small scale system. The parallel systems are also divided into asymmetric and symmetric multiprocessors. It depends on the similarity of the processors. One group of processors may have different privileges of running programs than the others.

Besides this parallel system, there are many different types of parallel architectures: ring architecture, hypercubes architecture, fat trees architecture, systolic arrays architecture and many others.

4.3 Performance vs. cost

If a system of *n* parallel processors is less efficient than one *n*-times-faster processor, the parallel system is often more cost effective to build. Parallel computation is used for tasks which require large amounts of computations, require long processing times, and can be divided into many independent subtasks. In recent years, most high performance 'computing systems, also known as supercomputers, have parallel architectures.

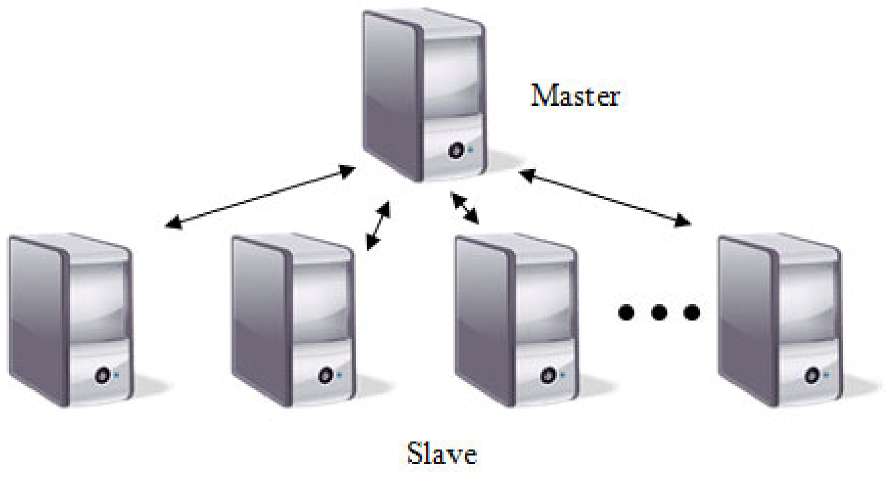


Figure 4.1: Master-Slave Model.

4.4 Master-Slave Model

In this configuration of parallel processors, the master node executes the EP operations, i.e. selection and mutation, and the slave nodes evaluate the fitness of the solutions and send back the results to the master node. Master-slave EAs are widely used for several reasons [20] such as the following:

1. The explored search space is similar to serial EA;
2. They are easy to implement; and
3. The use of this model can result in significant performance improvement.

The level of communication in this approach is high. It happens when the master node sends the solutions to the slaves, and the slaves return the fitness values to the master node. Thus, the speedup is limited to the serial part of the processing such as the communication and the tasks solely performed by the master slave.

Grefenstette [48] was one of the earliest to use master-slave parallel EA. His works is on the performance of parallel GAs (genetic algorithms). He proposed four prototypes of GAs. The first three were variants of a master-slave scheme, and the fourth was a multiple-demes parallel GA. He also suggested to combine both schemes to create a hybrid parallel GA.

The key performance indicator in this scheme is to see if the master-slave model could improve the performance, i.e. the computing time and the quality of the solutions. In the earlier works [35], even though the master-slave approach was speculated to be able to give many benefits, studies have shown that some implementations have stumbled upon a scalability problem. The problem occurred when more processors were used causing the efficiency of the algorithm decrease. This is mainly due to the communication costs and the serial component of the process.

In this thesis, the master-slave scheme is applied. The master node initially creates individuals and then sends them to the slave nodes for evaluation. The slave nodes evaluate the individual. Once all individuals are evaluated, the slave nodes send the fitness values back to the master node. The master node will then do the mutation, and selection.

4.4.1 Execution Time

The execution time of the master node is important in this approach. The master node performs solution generations, mutations, evaluations and selections. The slave nodes only perform the evaluations.

In the process, firstly, the master node divides the population evenly among slave nodes and send them to the slaves in time T_c. Then, the fraction of the population is evaluated using the time $\frac{nT_f}{P}$, where T_f is the time required to evaluate an individual, n is the size of the population, and $\mathcal{P}$ is the number of processors. Finally, the slaves will evaluate the given solutions and return the fitness values back to the master node. There is a delay time T_c for all slaves and master node before all fitness values are back to the master node before it can be processed further. According to [20], the elapsed time for one generation of the parallel GA (or EP) may be estimated as

$$T_p = \mathcal{P}T_c + \frac{nT_f}{\mathcal{P}} \tag{4.1}$$

In Eq. 4.1, the computation time decreases if the number of slaves increases, but with the expense of increasing communication time. This situation creates the needs to use an optimal number of processors to minimise the execution time. The optimal solution can be found by setting $\frac{\partial T_p}{\partial \mathcal{P}}$ and solve for $\mathcal{P}$ to get

$$\mathcal{P}^* = \sqrt{\frac{nT_f f}{T_c}} \tag{4.2}$$

which translate to the optimal number of slaves to $\mathcal{S}^* = \mathcal{P}^* - 1$.

Equation 4.2 can be deduced further if the T_c is a function of the amount of information (x), the inverse of network bandwidth (B) and the latency of communications (L).

Since each slave receives $n/\mathcal{P}$ number of individuals, the time required to send them to the slaves is

$$T_{send} = B\frac{nl_i}{\mathcal{P}} + L \tag{4.3}$$

and the time to send them back to the master node is

$$T_{recv} = B\frac{nl_f}{\mathcal{P}} + L \tag{4.4}$$

where l_i and l_f are the length of the individual and the fitness values, respectively. Thus the optimal number of processors becomes

$$\mathcal{P}^* = \sqrt{\frac{n(T_f + B(l_f - l_i))}{L}} \tag{4.5}$$

4.5 Coarse-grained Model

This is the most natural EP model. This approach is also known as an "island" model. An "island" model comprises of several islands in which there are a number of centralised EPs running in parallel on different processors. For properly modelling the technique, several questions must be answered [63]:

- How many islands will be used?
- How are they interconnected?
- Which individuals would migrate?
- How often do migrations occur?
- Should each island be running identical EP?
- How big is the population?

These questions have resulted in many research papers and theses such as [20; 130; 26]. Overall, these island models can significantly improve the problem-solving capabilities of EAs [63]. However, to achieve this improvement, the cost is extremely high. There are many factors that need to be understood in order for the approach to be effective for particular applications [63].

This approach of improvement is also known as “coarse-grained” or “distributed” EAs, simply because the communication to computation ratio is low [20]. Often, this approach is implemented on distributed-memory computers. The simplest implementation of this approach is to run the EA on each deme without any communication among them. However, an isolated deme may produce a lower fitness value than the ones achieved by a single large population [49]. Another attempt is to start the migration after the demes are completely converged [14].

Another type of implementation is when the demes are fully connected. In this configuration, the demes communicate between each other by migrating solutions amongst them. This is known as fully connected demes. The parameters that control the migration are (1) what type of solutions are to be migrated - just the optimal solutions or statistically selected solutions (2) and the migration rate.

The upper bound of the frequency of migration is by exchanging solutions at every generation [20]. This represents the most intensive communication, where the demes are fully connected. In this situation, the migration rate is at the maximum and communication occurs at every level during the generations. The difference from master-slave model is that the coarse-grained model has at most one migration event in a generation. It may occur before or after selection and recombination.

The migrants and the individuals that they replace can be chosen in many ways. The migrants can be the best candidate at home or chosen randomly by using a specified probabilistic distribution. The individuals they replaced can be the worst in the population or chosen randomly. The convergence of the solution depends on the migrant policy employed in this process.

The thesis considers minimal migration between demes and the migration only occurs with the adjacent deme and it is one way. The migration occurs at every generation. The best individuals are copied to adjacent deme. The migrant will replace the worst individual in the deme. The effect of migration rate to the quality of the solution is studied.

4.6 Amdahl's Law, Parallel Speedup and Efficiency

The discussion on parallel computing is incomplete without mentioning the *Amdahl's Law*. The *speedup* of a parallel program is defined as the ratio between the time taken to complete a given task on one processor to the time taken to complete the same task on multiple processors [54]. Mathematically, let $T(N)$ be the time required to complete the task on N processors. The speedup $S(N)$ is the ratio

$$S(N) = \frac{T(1)}{T(N)} \tag{4.6}$$

If the execution time has two portion, i.e. the serial time portion T_s and the parallel time, T_p, the speedup can be calculated as:

$$S(N) = \frac{T(1)}{T(N)} = \frac{T_s + T_p}{T_s + T_p(N)} \tag{4.7}$$

Eq. 4.7 is known as the *Amdahl's Law*.

Finally, the efficiency of a parallel program is calculated as

$$E(N) = \frac{S(N)}{N} \tag{4.8}$$

4.7 Summary

This chapter explains the concept of parallel EP. Two models are considered, i.e. master-slave and coarse-grained. There are several issues arising from using parallel EP notably the speedup and the efficiency. However, since EP is inherently ready for parallel implementation, the benefit of implementing parallel EP is numerous such as expanding the search space and improving the time to search for feasible solutions. The next chapter explains the implementation of the equations of motion of an aircraft and its integration into EP.

5 Implementation of an Air Combat Problem

5.1 Introduction

In this chapter, a realistic air combat has been enacted using Evolutionary Programming. This research focuses on the case of an air combat between two players. The assumption is that one player is superior to the other. The superior player acts as the pursuer and the second player is the evader. The different roles form the fitness function or the objective function required by EP. Generally, the pursuer aims to intercept the evader. The pursuer can be a jet fighter aircraft considering an interception when its gun takes a shooting position of the evader, or it can be a missile in which an interception is considered when the separation distance of the vehicles is less than the lethal distance of the missile. In this situation, a feasible solution is defined as a solution that increases the probability of survival for the evading player. Furthermore, the objective is to avoid interception. A bad solution is defined as a solution that has a high probability of causing the evader to be intercepted.

The study aims to find feasible solutions using two vehicle models, i.e. three degree of freedom (3DOF) and six degree of freedom (6DOF) which are as close as possible to a real time use. The 3DOF formulation is also known as a point-mass formulation because it only involves translation and does not take into consideration the aircraft attitude. The fact is that the 6DOF formulation is more complex because it involves both translation and attitudes. In this chapter, both approaches are discussed.

The search for a feasible control solution is a continual process of generating new control solutions over a period time by adapting previously computed control solutions for future motion. This concept is illustrated in Figure 5.1. The concept is started by defining a spawn point. A spawn point is defined as the time where the planner updates the output trajectory. During the course of a mission, a planner will continually compute an updated trajectory which starts from the next spawn point (point 3) until a predetermined planning time horizon (point 4 in the figure). The computation starts (point 2) somewhere between the previous spawn point (point 1) and the next spawn point (point 3). The time between point 2 and point 3 in Figure 5.1 is the computation time required by the planner. An update of the feasible control solutions is sent to the controller of the vehicle for execution every time the vehicle reaches a spawn point. The time difference between spawn points specifies the maximum time available to plan for the next feasible trajectory. It is important to note that the shorter the time planning requires faster algorithms. This is the niche for Evolutionary Programming algorithm to show its capability. Since traditional approaches require high computing power for nonlinear problems, EP has a good potential to find feasible solution in a shorter period of time. This concept is similar to that of Model Predictive Control (MPC) [18].

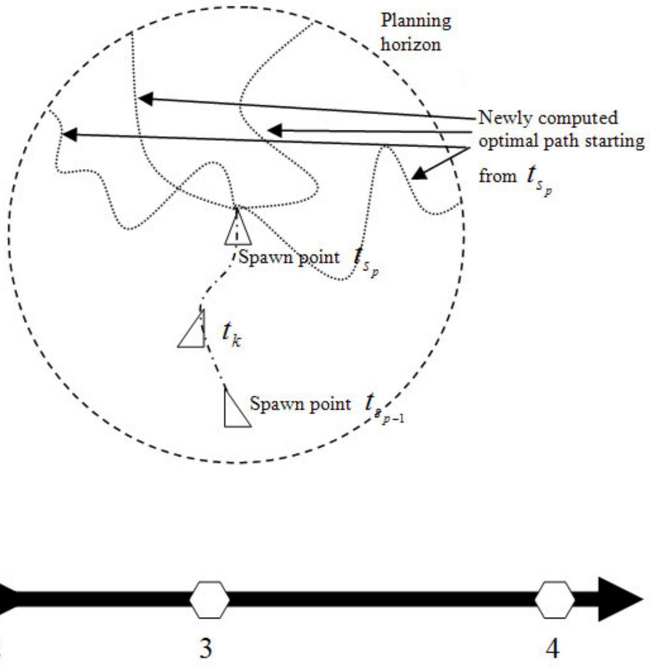

Planning horizon
Newly computed optimal path starting from t_{s_p}
Spawn point t_{s_p}
t_k
Spawn point $t_{s_{p-1}}$
1
2
3
4

Figure 5.1: Time horizon planning of optimal pursuit-evasion problem.

The difficulty of the problem hinges on how one defines the objective function and the constraints on the input variables. In order to find a feasible control solution for the evader against a very agile pursuer, it would be done by putting them in a simulated game over a predetermined time period. To simulate a realistic air combat scenario, the actual aerodynamic of the aircraft and stability characteristics are employed. The information obtained from this simulated combat is combined with the aircraft general equations of motion and constraints. At the end, the mathematical model forms the foundation of the objective function.

The search for the feasible solution is done by running each solution in self-play simulated games. Contrary to the approach used by [126] and [33], the feasible solution caters for both the inner-loop and the outer-loop control. The inner-loop control ensures that the vehicle does not violate any constraint during the whole period of the game and the outer-loop control ensures that the vehicle's trajectory is feasible for evasion. Building the correct mathematical model is not easy. As emphasised in Fishman's monograph [39], there is no guarantee that the time and effort devoted to the modelling will return useful results and satisfactory benefits. This is rather an on-going exercise in which the researcher will always improve the mathematical model to achieve a better result. However, relying too much on the method and not enough on the ingenuity could hinder from achieving a higher goal. Thus, a proper balance between the two is important to achieve a higher probability of success. The second point in the monograph explains the tendency of the researcher to treat the problem as the best representation of reality. A good understanding of other methods used to solve the same problem is important as a benchmark against the researcher's own method of choice. The third concern is using the mathematical model beyond its capability.

There are two conflicting objectives in simulating a real system as mentioned by [101], i.e. realism and simplicity. Realism is important so that the model is representing the closest possible actual problem and at the same time the mathematics must be tractable for implementation on a computer. The most important aspect in modelling of a problem is its high correlation between the prediction by the model and what would actually happen with the real system [101]. One way of ensuring this is to test and establish control over the solution. As for a pursuit-evasion problem between two air vehicles, the validation can be either through other validated simulation models or feedback from experienced fighter pilots. Due to the nature of the problem, both options are classified materials. They require security clearance from the military.

Since the problem is complex, and most analytical solutions have to be simplified just to make them tractable, the only alternative is through the use of simulated game [85]. The benefits of running simulations are numerous [85] such as (i) it makes it possible to study the effects of certain informational, organisational, and environmental changes by altering the inputs and observing the outputs, (ii) the experience of designing a computer simulation model can be more valuable than the actual simulation itself, (iii) simulation of a complex system can yield valuable insight into identifying important variables and how these variables interact, and (iv) it can be used to experiment with new situations, for example new aircraft designs, and (v) a simulation is safer and less costly than the real experiment.

The remaining sections describe the development of the mathematical model of the objective function used in EP.

5.2 Assumptions

The assumptions in the development of the simulation models are as follows:

1. A flat-Earth axis system is used.
2. External disturbances such as wind gust and turbulence are currently ignored.
3. The atmosphere follows the International Standard Atmosphere.
4. The evader knows very well about the pursuer's systems and their capability.
5. The vehicles have a rigid-body.
6. The missile (the pursuer) has a negligible response lag.
7. There are no above ground obstacles, such as hills and mountains, but the ground is modeled as a constraint.

5.3 Aerodynamic Forces and Moments

The aircraft equations of motion are derived from Newton's second law of motion. It states that the summation of all external forces acting on a body must be equal to the time rate of change of its momentum. Also, the sum of the external moments acting on a body must be equal to the time rate of change of its moment of momentum. The 3DOF equations of motion are described in 5.6.3 and the 6DOF equations of motion are described in 5.7.3.

The forces and moments acting on an aircraft are typically defined in terms of dimensionless aerodynamic coefficients. In three-dimensional manoeuvres, the coefficients are functions of the two aerodynamic angle, namely angle-of-attack and sideslip angle, the Mach number and the Reynolds number. The Reynolds number and the dynamic pressure can be estimated at any altitude if the Mach number is known, together with the knowledge of the local atmosphere. Cases that involve the deflection of control surfaces such as the aileron, the elevator and the rudder will change the coefficients. However, there are other factors that change the coefficients such as the landing gear, external tanks and external weapons which are not considered in this thesis.

The aerodynamic forces acting on the aircraft are divided into components of X, Y, Z parallel to the body axes. The forces are given by

$$\begin{aligned} X &= q_0 S C_X \\ Y &= q_0 S C_Y \\ Z &= q_0 S C_Z \end{aligned} \tag{5.1}$$

where $q_0 = \frac{1}{2}\rho V^2$.

The force coefficients are derived from the various aerodynamic forces and moments acting on the airframe. The coefficients are derived based on the body axes coordinate system and they are non-dimensional.

In addition, the aerodynamic moments are calculated by assuming that the centre of the act is at the centre of the body mass. There are three moments acting on the centre of mass of the aircraft, i.e. the rolling moment, L, the pitching moment M and the yawing moment N. The moments are defined as

$$\begin{aligned} L &= q_0 S C_l \\ M &= q_0 S C_m \\ N &= q_0 S C_n \end{aligned} \tag{5.2}$$

5.4 Aerodynamic Coefficients

The dependence of an aerodynamic coefficient can be defined as

$$C_{()} = C_{()}(\alpha, \beta, M, H, \delta_s, T) \tag{5.3}$$

where subscript () represents X, Y, Z for forces and L, M, N for moments and δ_s represents the control surface deflection.

Eq. 5.3 implies a complicated functional dependence has to be modelled as a lookup-table using a computer model. In this model, some coefficients are static and they are measured from a stationary model in a wind tunnel. For highly agile fighter aircraft, the aerodynamic effects during manoeuvring are also important. These effects require a different equation model of the aerodynamic forces and moments. For the purpose of this study, the aircraft model in this thesis uses coefficients extracted from [47].

5.5 Component Buildup

The aircraft model was build based from the work of Stevens [118]. The aircraft aerodynamic and stability coefficients are employed. These coefficients are used by the equations of motion to calculate the aircraft dynamics at every time step. Mostly, they are built as a function of the angle of attack, the sideslip angle, the Mach number and altitude.

For this thesis, a small database prepared by Gilbert and Nguyen [47] for a low speed F-16 model data which was developed at the NASA Langley Dryden and Langley Research Center was used. An example of a three-dimensional plot is shown in Figure 5.2. The aerodynamic look-up table data are discrete in which aircraft models require data as arbitrary values of the independent variables. The look-up is performed by using an interpolation algorithm that comes with the data. The interpolation algorithm is included in Appendix A.

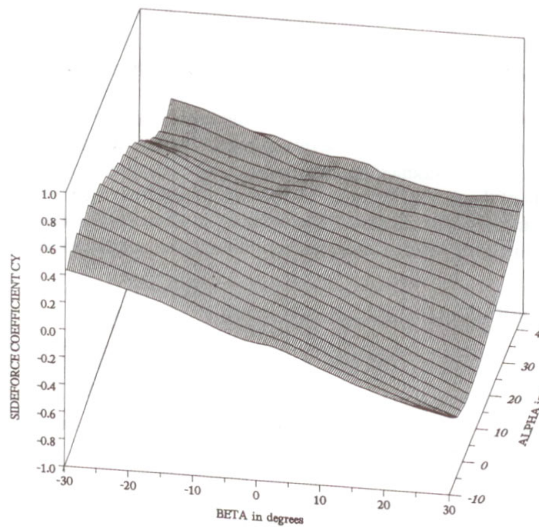


Figure 5.2: Sideforce coefficient of the F-16 model [118].

5.6 Three Degree of Freedom Simulation Model

In this section, a simulation using three degree of freedom (3DOF) or a point-mass model, is developed to describe the aircraft trajectories. The trajectories are represented through six state variables; angle-of-attack (α), bank angle (θ), velocity (v), east-ward position (x), north-ward position (y) and altitude (h). The flight path angle (γ) and heading angle(ψ) are used to control the aircraft.

The 3DOF model forms the basis of the pursuer's dynamics. The pursuer's kinematics is assumed to behave closely with its real-world counterparts when equipped with a thrust model and the known proportional navigation model. However, due to the limited information about the pursuer (i.e. the missile), an estimation has to be made.

5.6.1 Introduction

The development of an EP algorithm was firstly initiated for a three degree of freedom pursuit-evasion optimisation problem. The reason is because the 3DOF model is simpler to build as it uses a preliminary platform in simulating a pursuit-evasion game before taking a step further into 6DOF simulation. For 3DOF problems, both players are presented in the form of point-mass models. Due to the limited information on the players, the aerodynamic and relevant coefficients are assumed to be constant. The simplified coefficients are derived from [86] and [133].

5.6.2 Vector Rotation

Military aircrafts perform extreme manoeuvres such as half-loops or full-loops. In terms of modelling these manoeuvres mathematically, the use of quaternion is natural because Euler rotation suffers from singularity problem [118]. This occurs when the flight path angle, $\theta = \pm\pi/2$. A quaternion is used to resolve this problem. The quaternion is used to rotate a Euclidean vector similar to the Euler rotation formula but in a much simpler form. In quaternion, the vector part defines the rotation axis, and the scalar part defines the angle of rotation.

The rotation axis is specified by its direction cosines in the reference frame. It is typical to use unity norm constraint on the quaternion. If the direction angles of the axis are α, β and γ, and a measure of the rotation angle is δ, the rotation quaternion is written as:

$$q = \begin{bmatrix} cos\,\delta \\ cos\,\alpha\,sin\,\delta \\ cos\,\beta\,sin\,\delta \\ cos\,\gamma\,sin\,\delta \end{bmatrix} = \begin{bmatrix} cos\,\delta \\ sin\,\delta\,\mathbf{n}^r \end{bmatrix} \tag{5.4}$$

n is a unit vector along the rotation axis,

$$\mathbf{n} = [cos\,\alpha\,cos\,\beta\,cos\,\gamma]^T \tag{5.5}$$

which,

$$\text{norm}(q) = cos^2\delta + sin^2\delta(cos^2\alpha + cos^2\beta + cos^2\gamma) = 1 \tag{5.6}$$

This equation guarantees that there is a unique quaternion for each value of δ in the range $\pm 180°$ which covers all possible rotations [118]. In Euler angles, three-dimensional coordinate rotations are built up as a sequence of plane rotations. However, the rotations are not commutative. A standard aircraft practice is to describe the aircraft orientation by the z, y, z right-handed rotation sequence as required by the transformation equation to get from a reference system based on flat-Earth coordinate system into alignment with an aircraft body-fixed frame. Basically, from the reference frame, the sequence rotation is

1. Right-handed rotation about the z-axis (i.e. positive yaw,ψ);
2. Right-handed rotation about the new y-axis (i.e. positive pitch,θ); and
3. Right-handed rotation about the new x-axis (i.e. position roll,ϕ).

Thus, the rotation matrices can be written as

$$\mathbf{u}^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}^r \tag{5.7}$$

In quaternion, the complete transformation matrix is

$$C_{b/r} = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_3 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{bmatrix} \tag{5.8}$$

For the yaw, pitch, roll sequence, the quaternion formulation is

$$\mathbf{u}^b = q_{roll}^{-1} q_{pitch}^{-1} q_{yaw}^{-1} \mathbf{u}^r q_{yaw} q_{pitch} q_{roll} \tag{5.9}$$

where

$$q_{yaw} = \begin{bmatrix} cos(\psi/2) \\ 0 \\ 0 \\ sin(\psi/2) \end{bmatrix} \quad q_{pitch} = \begin{bmatrix} cos(\theta/2) \\ 0 \\ sin(\theta/2) \\ 0 \end{bmatrix} \quad q_{roll} = \begin{bmatrix} cos(\phi/2) \\ sin(\phi/2) \\ 0 \\ 0 \end{bmatrix}$$

Using quaternion multiplication, these transformation gives

$$\begin{aligned} q_0 &= \pm(cos(\phi/2)cos(\theta/2)cos(\psi/2) + sin(\phi/2)sin(\theta/2)sin(\psi/2)) \\ q_1 &= \pm(sin(\phi/2)cos(\theta/2)cos(\psi/2) - cos(\phi/2)sin(\theta/2)sin(\psi/2)) \\ q_2 &= \pm(cos(\phi/2)sin(\theta/2)cos(\psi/2) + sin(\phi/2)cos(\theta/2)sin(\psi/2)) \\ q_3 &= \pm(cos(\phi/2)cos(\theta/2)sin(\psi/2) - sin(\phi/2)sin(\theta/2)cos(\psi/2)) \end{aligned} \tag{5.10}$$

which represents the element of $q_{b/r}$.

5.6.3 Equations of Motion

The mathematical model developed for the 3DOF model is based on the model described by Miele [81]. In the 3DOF model, the trajectories are considered short range and the velocities are far smaller than the Earth rotation velocity. Thus, the general dynamics equation is reduced to the following:

$$\mathbf{T} + \mathbf{A} + m\mathbf{g} = m\mathbf{a} = m\frac{d\mathbf{V}}{dt} \tag{5.11}$$

where **T** is the thrust vector, **A** is the aerodynamic force vector acting on the aircraft. m is the mass, **g** is the acceleration of the gravity, **a** is the acceleration of the aircraft with respect to the Earth, **V** is the velocity and t is the time. The velocity, **V** is described as:-

$$\mathbf{V} = \frac{d\mathbf{EO}}{dt} \tag{5.12}$$

where **EO** is the position vector of the aircraft in relation to the location of the earth.

Then, using Newton's second law, the force equation becomes:

$$m\frac{d\mathbf{V}}{dt} = \mathbf{F} \tag{5.13}$$

5.6.4 Equations over Flat-Earth

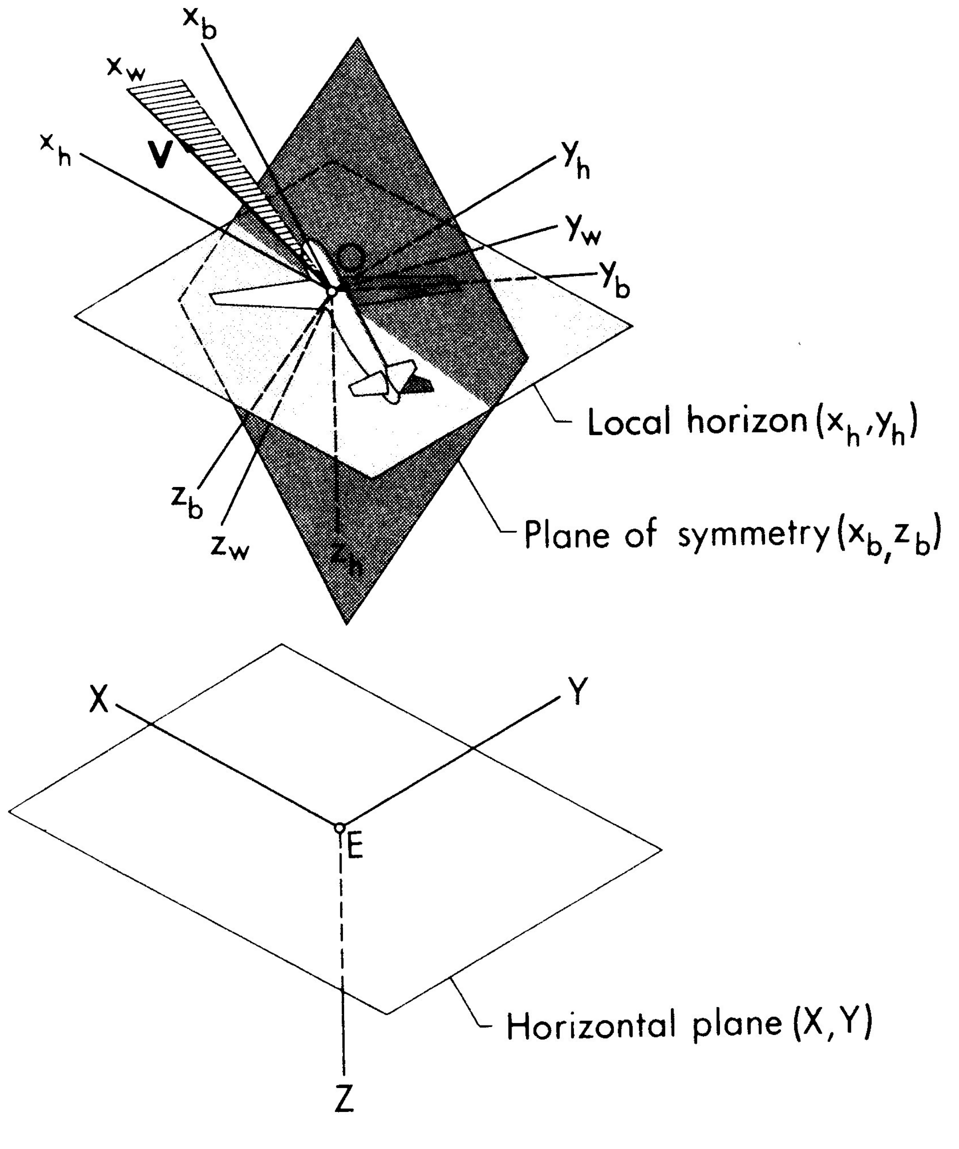


Figure 5.3: Coordinate systems for flight over a flat Earth. [81]

The coordinate systems used for a flight over flat Earth are given in Figure 5.3.

The ground axes system is $EXYZ$, the local horizon is $Ox_hy_hz_h$, the wind axis is $Ox_wy_wz_w$ and the body axis system is $Ox_by_bz_b$.

As for the atmosphere, a specific set of data representing the atmospheric average condition is adopted. This set of data is widely known as the International Standard Atmosphere [31]. For the sake of simplicity, this atmospheric model is used in the simulation.

The set of equations of motion is provided as the following derived from the work of Miele [81]:

$$\frac{dv}{dt} = \frac{1}{m}(T\cos\epsilon - D) - g\sin\gamma \tag{5.14}$$

$$\frac{d\gamma}{dt} = \frac{1}{mv}(T\sin\epsilon + L)\cos\phi - \frac{g}{v}\cos\gamma \tag{5.15}$$

$$\frac{d\psi}{dt} = \frac{1}{mv\cos\gamma}(T\sin\alpha + L)\sin\phi \tag{5.16}$$

$$\frac{dx}{dt} = v\cos\gamma\cos\psi \tag{5.17}$$

$$\frac{dy}{dt} = v\cos\gamma\sin\psi \tag{5.18}$$

$$\frac{dh}{dt} = v\sin\gamma \tag{5.19}$$

The lift force, L, and the drag force, D, are defined as:

$$L = \frac{1}{2}\rho(h)v^2S \cdot C_L(\alpha) \tag{5.20}$$

$$D = \frac{1}{2}\rho(h)v^2S \cdot C_D(\alpha) \tag{5.21}$$

where ρ is the air density, v is the velocity of the aircraft, S is the reference wing planform area, C_D is the drag coefficient which is a function of angle of attack, α, and C_L is the lift coefficient which is a function of angle of attack. The definitions of vector and angle variables of the forces and moments are shown in Figure 5.4 and 5.5.

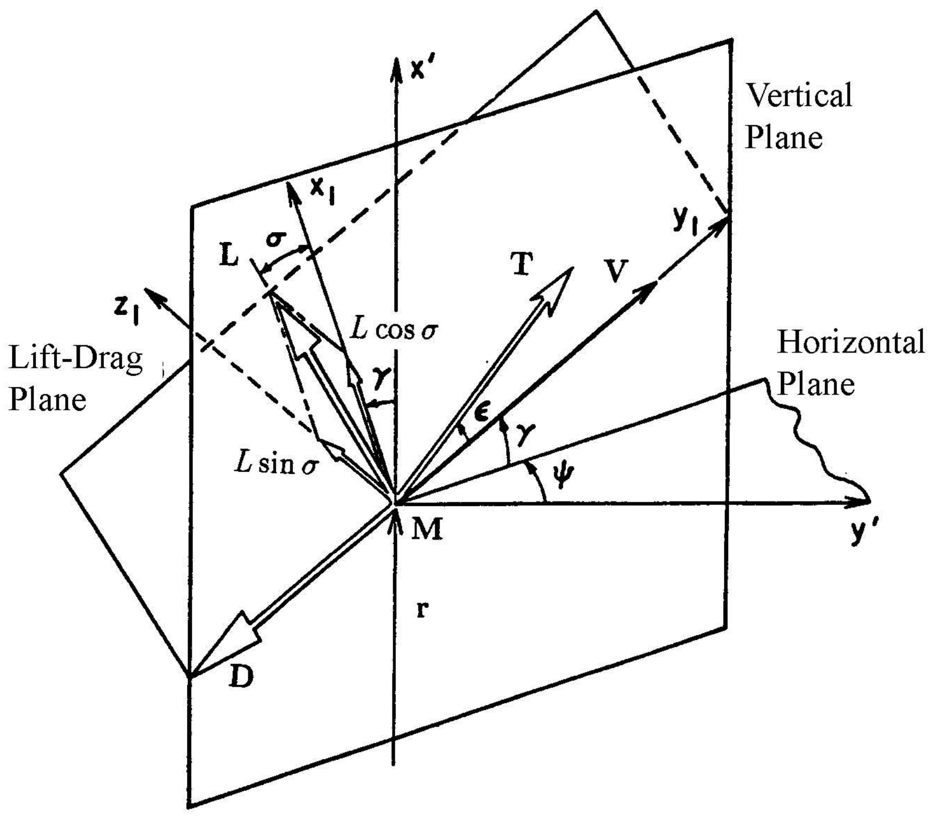


Figure 5.4: Aerodynamic Forces in Three Dimensions. [125]

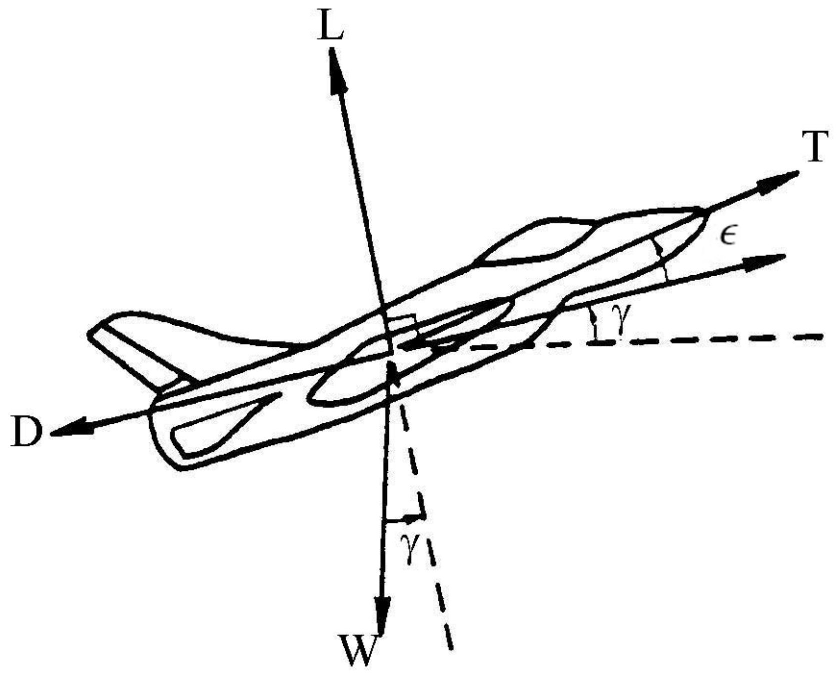


Figure 5.5: Aerodynamic and Propulsive Forces [125].

5.6.5 Aircraft Performance Calculation

A trajectory of the evader must not violate predefined performance limits in order to be considered feasible. For the 3DOF model, the performance limits are maximum angle of attack limit, maximum turning rate, structural load limit, stall limit, dynamic pressure limit, ground proximity limit, maximum ceiling limit and fuel limit. The players must not exceed these limits at all time.

5.6.5.1 Thrust

For a prescribed throttle setting, the thrust is a function of the altitude and Mach number, in which their effect have been considered in the simulation. The thrust is simplified by a function of altitude given in Eq. 5.22 derived from [125]

$$T_{alt} = T_{SL} \frac{\rho_{alt}}{\rho_{SL}} \tag{5.22}$$

Thrust is produced by burning fuel in the engine, thus the mass of the aircraft will vary with time. The equation for mass flow rate is [125]

$$\frac{dm}{dt} = -\frac{c}{g} T \tag{5.23}$$

where T is the thrust magnitude, c is the specific fuel consumption and g is the gravitational acceleration. In this equation, the specific fuel consumption is assumed to be constant due to data unavailability for the specified engine.

With regards to a rocket engine, the thrust is calculated from Eq. 5.24 from Ref. [120]

$$F = I_{sp} \cdot \dot{m} \cdot g_0 \tag{5.24}$$

where F is the thrust produced, I_{sp} is the specific impulse in seconds, $\dot{m}$ is the mass flow rate and g_0 is the sea-level gravitational acceleration. The problem in this calculation is the lack of information on the mass flow rate and specific impulse of the missiles of interest. Understandably, the information is classified due to the fact that the missiles are still in military service. One has to assume the values for the sake of the realisation of the simulation. The typical impulse value for a solid propellant is between 200-300 seconds, see Ref. [120].

5.6.5.2 Load Factor and Turning Performance

The maximum and minimum load factors determine the region where the aircraft could manoeuvre safely. At a high speed, the maximum load factor is based on the structural limitations of the aircraft and the limitations of the person handling the aircraft. For this reason, a typical load factor can be as high as 8 for a jet fighter [75]. For an unmanned aircraft, this value can be higher since there is no human pilot involved. One of the adverse effects on humans at high load factors is the draining of blood from the head to the lower part of the body. This will cause the pilot to 'black-out' or experience loss of vision or consciousness.

Another factor to be considered is the maximum load factor and the available thrust will determine the maximum turning rate and the minimum turning radius to be executed by the aircraft at a particular condition. For a fighter in an air combat, it is necessary for survival that it optimises the fastest turn or the minimum turning radius at a prescribed thrust. The situation becomes more complicated when the lift coefficient is close to the stalling point and the load factor is high. The thrust available has to work harder to overcome the high drag that the aircraft produces. An optimal trade-off has to be identified within a very short period of time.

The load factor for three-dimensional turn is given by Eq. 5.25 [125].

$$n = \frac{\cos\gamma}{\cos\phi} \tag{5.25}$$

For a three dimensional turn, the calculation of a bank angle is based on Eq. 5.25. In terms of simulating the aircraft to perform various modes of manoeuvres, the simulation software must be able to limit the maximum turning rate and the minimum turning radius that corresponds to the maximum load factor.

The turning rate is calculated using Eq. 5.26 provided the constraints on the load factor, lift coefficient and thrust available are satisfied. The turn considers the case of turning at low altitude where the structural constraints, n_s, is enforced.

$$\dot{\psi} = \frac{q\sqrt{n_s^2 - 1}}{aM} \tag{5.26}$$

At a higher altitude, the maximum load factor, n_{max}, as calculated by Eq. 5.27, is lower than n_s due to the smaller value of the ambient pressure, p and the structural constraint may no longer applies.

$$n_{max} = \frac{kpS}{2W}(C_{L_{max}}M^2)_{max} \tag{5.27}$$

where k is the ratio of specific heats, p is the ambient pressure, and M is the Mach number.

5.6.6 The Pursuer

The pursuer is modelled as a point-mass model described in 5.6.3. The guidance system used is a Proportional Navigation Guidance system with navigation constant, N, as described in Section 5.7.4. The parameters of the pursuer are described in Appendix A.2.

The movement of the pursuer is limited by the amount of fuel it has and the maximum load factor. The maximum load factor determines the minimum turning radius and the maximum turning rate of the pursuer. Besides that, the missile aerodynamics are available from [87].

5.7 Six Degree of Freedom Simulation Model

5.7.1 Introduction

The 6DOF simulation is based on the source code provided by Stevens [118] and Zipfel [133]. The code provides the maximum fidelity, but requires a significant amount of information, such as a full aerodynamics and thrust tables, a complete flight control system design, mass parameters which include moment of inertia and, if applicable, sensors and guidance logic.

However, these control variables are restricted to the aileron, elevator and rudder deflections, and throttle settings. These control variables are coded as required by EP. The coded control variables is called *a solution* in EP. Each solution is evaluated, mutated and recombined and stopped when a predefined number of generation is reached. The best solution is the solution that gives the maximum value of the objective function, $J(x)$.

5.7.2 Trajectory

The act of flying involves a six degree of freedom motion, inclusive of translations and rotations. Translations are in x, y and z axes and rotation rotates the aircraft in those axes. For a fixed wing aircraft, these movements are controlled by control surfaces, i.e. aileron, rudder, elevator, slat, flap, and spoiler, and by varying the engine thrust. Trajectories are defined as the path taken by the aircraft. A trajectory can be mathematically defined as

$$T(t) = f(\delta_a(t), \delta_e(t), \delta_r(t), \delta_\tau(t), x_i(t)) \tag{5.28}$$

where,

$\delta_a(t)$ is the aileron deflection angle.
$\delta_e(t)$ is the elevator deflection angle.
$\delta_r(t)$ is the rudder deflection angle.
$\delta_\tau(t)$ is the throttle setting.
$x_i(t)$ is the initial states.

The velocity vector, $v(t)$, is defined as

$$v(t) = f(thrust(t), C_L(t), C_D(t), M(t)) \tag{5.29}$$

which means that the velocity vector is a function of thrust, aerodynamic derivatives of the aircraft and ambient atmospheric conditions.

The thrust is a function of the throttle setting and ambient atmospheric conditions. On the other hand, the aerodynamic coefficients are functions of Mach number, aircraft attitude and the position of the control surfaces.

In this project, a trajectory for a n second game is defined. It consists of a series of randomly selected values of δ_a, δ_e, δ_r and δ_τ at every second up to t_f seconds.

5.7.3 Nonlinear Equations of Motion

This section discusses the set up of the aircraft model so that it could be used as a fitness function to test solutions discussed in the previous sections.

The aerodynamic forces and moments discussed in 5.3 and 5.4 are combined with the vector equations of motion to design the aircraft model. This model uses the EP algorithm to find a feasible solution for the evader through simulation.

As the complete aircraft data are insufficient and not available in the literature, the study uses an F-16 model from Stevens [118] and Nguyen [47]. Basically, the data was from the wind tunnel data taken during the NASA's Langley wind tunnel tests on a subscale model of an F-16 aircraft. The range of speeds covered by the data is up to Mach 0.6. The stability and aerodynamic force and moment coefficients together with the aircraft equations of motion are used to obtain the 6DOF aircraft model for simulation and analytical purposes.

The following discussion is derived from [133]. We have considered the derivation of aircraft motion using the aerodynamic force and moment models that are combined with the vector equations of motion.

As given by [118], the state vector for the body-axes equation is

$$X = [p_N \;\; p_E \;\; p_D \;\; \phi \;\; \theta \;\; \psi \;\; U \;\; V \;\; W \;\; P \;\; Q \;\; R] \tag{5.30}$$

Meanwhile the typical control vector is

$$U = [\delta_t \;\; \delta_e \;\; \delta_a \;\; \delta_r] \tag{5.31}$$

this control vector ignores the presence of wind. If the presence of wind is considered the control vector is

$$U = [\delta_t \;\; \delta_e \;\; \delta_a \;\; \delta_r \;\; W_N \;\; W_E \;\; W_D] \tag{5.32}$$

A complete set of equations of motion is displayed in Table 5.1.

5.7.3.1 Engine Model

The engine model is derived from NASA's F-16 afterburning turbofan engine model [118]. The model has the thrust response of a first-order lag. The lag time constant is a function of the actual engine power level (POW) and the commanded power (CPOW). A function PDOT calculates the time constant, whose value is the rate of change of power. The state variable X_{13} represents the actual power level.

Furthermore, the function TGEAR (throttle gearing) links the commanded power level with the throttle position. Subsequently the function THRUST calculates the variation of engine thrust which takes power level, altitude and Mach number as inputs.

5.7.3.2 Sign Convention of the Control Surface Deflections

The sign convention of the control surface deflections used in the mathematical model of the aircraft is given in Table 5.2 [118].

Table 5.1: The Flat-Earth,Body-Axes 6-DOF Equations [118].

Force Equations

$$\dot{U} = RV - QW - g_D sin\phi + (X_A + X_T)/m$$
$$\dot{V} = -RU + PW + g_D sin\phi cos\theta + (Y_A + Y_T)/m$$
$$\dot{W} = QU - PV + g_D sin\phi cos\theta + (Z_A + Z_T)/m$$

Kinematic Equations

$$\dot{\phi} = P + tan\theta(Qsin\phi + Rcos\phi)$$
$$\dot{\theta} = -RU + PW + g_D sin\phi cos\theta + (Y_A + Y_T)/m$$
$$\dot{\psi} = QU - PV + g_D sin\phi cos\theta + (Z_A + Z_T)/m$$

Moment Equations

$$\Gamma\dot{P} = J_{xz}[J_x - J_y + J_z]\,PQ - [J_z(J_z - J_y) + J_{xz}^2]\,QR + J_z\ell + J_{xz}n$$
$$J_y\,\dot{Q} = (J_z - J_x)\,PR - J_{xz}(P^2 - R^2) + m$$
$$\Gamma\dot{R} = [(J_x - J_y)J_x + J_{xz}^2]PQ - J_{xz}[J_x - J_y + J_z]QR + J_{xz}\ell + J_x n$$
where $\Gamma = J_xJ_z - J_{xz}^2$

Navigation Equations

$$\dot{p}_N = Uc\theta c\psi + V(-c\phi s\psi + s\phi s\theta c\psi) + W(s\phi s\psi + c\phi s\theta c\psi)$$
$$\dot{p}_E = Uc\theta c\psi + V(c\phi s\psi + s\phi s\theta c\psi) + W(-s\phi s\psi + c\phi s\theta c\psi)$$
$$\dot{h} = Us\theta - Vs\phi c\theta - Wc\phi c\theta$$
where $s\theta = \sin\theta$,$s\phi = \sin\phi$, $s\psi = \sin\psi$,$c\theta = \cos\theta$,$c\phi = \cos\phi$, $c\psi = \cos\psi$

Table 5.2: Aircraft Control-Surface Sign Conventions.

Control Surface	Deflection	Sense	Primary Effect
Aileron	right-wing trailing edge down	positive	negative rolling moment
Elevator	trailing-edge down	positive	negative pitching moment
Rudder	trailing-edge left	positive	negative yawing moment

5.7.3.3 States and Controls

In this mathematical model, thirteen states of the vehicle are calculated at every time step

1. north position (x),
2. east position (y),
3. altitude (h or $-z$),
4. roll angle (ϕ),
5. pitch angle (θ),
6. yaw angle (ψ),
7. total velocity (V),
8. angle of attack (α),
9. angle of side-slip (β),
10. roll rate (P),
11. pitch rate (Q),
12. yaw rate (R), and
13. engine power.

As mentioned before, the control inputs are the aileron deflection, the elevator deflection, the rudder deflection and the thrust setting. The thrust acts positively along the body x-axis. A positive thrust will increase the acceleration along the x-axis. A positive aileron deflection gives a negative roll. A positive elevator will give a negative pitch and a positive rudder will a negative yaw as given in 5.2.

5.7.3.4 Constraints

There are several constraints implemented on the objective function to prevent the evader from stalling, crashing to the ground and exceeding the structural load limit. Thus, the feasible solution must not violate any of these constraints.

In this implementation, the angle of attack is limited from -10° to 40°. This is to prevent the evader from experiencing stall and exceeding the capability of the aircraft.

Similarly, a maximum structural load factor is also set on the pursuer. The value can be varied from 20 to 80. This value will determine the maneuverability of the pursuer based from Eq. 5.26 and 5.27.

However, the maximum structural load factor is set to 9 for the evader. The evader maximum turning rate and minimum turning radius are determined by the maximum load factor the aircraft could sustain. If the evader is unmanned, the maximum structural load factor can be higher than 9. This will definitely provide an added advantage to the evader in term of an increase in the maximum turning rate and a smaller minimum turning radius.

The most important constraint is that at any time during the pursuit-evasion encounter, the distance between the evader and the pursuer must not be less than the given interception radius. Once the evader is inside the interception radius where detonations will disable the aircraft, capture is assumed and the game is over. This assumption is made because in modern air warfare, it is very unusual for aircraft to be in a dogfight. Most of the time, they simply use standoff weapons, such as missiles.

5.7.3.5 Actuators

The actuators of the control surfaces are modeled as first-order lags with a gain (K) as well as limits on deflection and rates. The thrust has a unity gain and a rate limit of ± 10,000 lbs. The elevator has a gain of $\frac{1}{0.0495}$ and a rate limit of ±60 deg/s. The aileron gain is $\frac{1}{0.0495}$ with a rate limit of ±80 deg/s. The rudder gain is $\frac{1}{0.0495}$ with a rate limit of ±120 deg/s. They are shown in Table 5.3.

5.7.3.6 Numerical Solution of the State Equation

An aircraft state equations of motion is highly nonlinear. The information of the aircraft aerodynamic and stability parameters are largely derived from experiments. As it is not possible to use analytical solutions, numerical methods are the only viable ways to compute an aircraft trajectory.

Generally, the state vector of a physical system will move in a smooth, continuous manner in the n-dimensional state-space. This is because of the way state variables describe the energy stored in a physical system. Energy in the state variables cannot have any instantaneous change, as it requires a gradual change of energy level. Therefore, derivatives of the state variables do exist. Taylor series expansion can be used to predict this motion.

A numerical evaluation of a continuous trajectory implies that, given the initial condition $X(t_0)$ and control input $U(t_0)$, the discrete sequential values of the states is calculated as

$$X(t_0 + kT), k = 1, 2, ... \tag{5.33}$$

that satisfies the state equations

$$\dot{(X)}(t) = F(X(t), U(t)) \tag{5.34}$$

This is called the *initial value problem*, and the time-step T is usually fixed.

There are many numerical methods to solve the initial-value problem, one of them is the Runge-Kutta (RK) method. For practical reasons, the fastest time to change the control surface angles is set to one second. The control surfaces can be retained at a fix position as long as required.

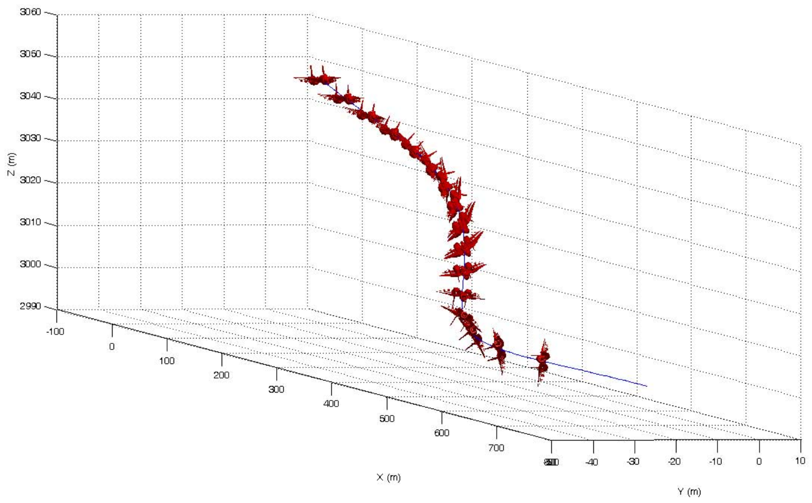


Figure 5.6: 6DOF simulation with aileron, elevator, rudder and throttle as control variables calculated with 4th order Runge-Kutta method.

5.7.3.7 Modelling the Pursuer

To simplify the modelling of the pursuer, a point-mass model is employed as discussed in Section 5.6. The states of the evader at any time step are fed into the pursuer navigation function. The integration time step for the pursuer is similar to the evader, i.e. 0.01 second.

Due to the limited information available, the pursuer's aerodynamic and performance data are simplified. In most cases, the data are set to be constant. These include the maximum lift coefficient $C_{L_{max}}$, the zero-lift drag coefficient C_{D_0} and C_{L_α}.

5.7.4 Proportional Navigation Guidance

A Proportional Navigation Guidance (PNG) system guides the pursuer to intercept the evader. In this guidance system, the pursuer is commanded to turn at a rate proportional to the angular velocity of the line of sight (LOS) as given in Figure 5.7. A line-of-sight is defined as an imaginary line from the pursuer to the evader.

The direction of the LOS is calculated by a device called "a seeker". A seeker is basically a radar or a sensor that tracks the evader; and the output is the angular velocity of the LOS with respect to the inertial space as measured by rate gyros mounted on the seeker. (LOS) as given in Figure 5.7. The equations of the PNG system are given in [13].

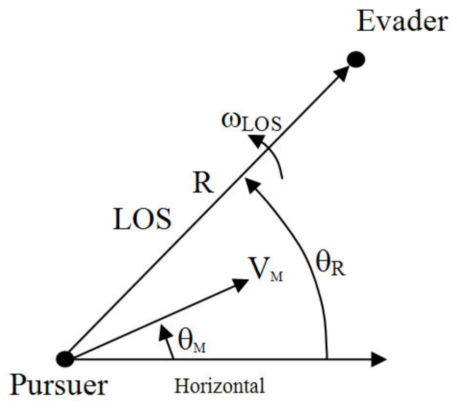


Figure 5.7: Angular velocity of the LOS for proportional navigation. [112]

The magnitude of the angular velocity of the LOS is determined by the components of pursuer and evader velocity perpendicular to the LOS. By referring Figure 5.7, the component of pursuer velocity perpendicular to the LOS is

$$V_{M(\perp LOS)} = V_M sin(\Theta_R - \Theta_T) \tag{5.35}$$

which generates a positive LOS rotation. The magnitude of the angular velocity is the difference between the components of the pursuer and the evader velocity perpendicular to the LOS divided by the distance between the evader and the pursuer *R*. The component of evader velocity perpendicular to the LOS is

$$V_{T(\perp LOS)} = V_T sin(\Theta_R - \Theta_M) \tag{5.36}$$

which generates a negative LOS rotation. If the perpendicular component of the pursuer and the evader velocity are equal and constant, the LOS angular rate is zero and the pursuer will be on a collision course with the evader. Generally, the angular velocity of the LOS is

$$\omega_{LOS} = \frac{V_M sin(\Theta_R - \Theta_M) - V_T sin(\Theta_R - \Theta_M)}{R}(\text{rad/sec}) \tag{5.37}$$

However, there is no closed-form solution for a general equation that describes the pursuit a highly nonlinear manoeuvring target [111]. In this thesis, the Pure Proportional Navigation [111] guidance system is used.

5.7.5 The Representation of the Solution in EP

In finding the feasible solution using EP, the control solution must be coded in a way that the EP will understand and is able to manipulate effectively. From a pilot perspective, to control an aircraft, the pilot will control at least four control variables, i.e. the aileron, the elevator, the rudder and the throttle, as in Figure 5.8. The first three are used to control the position and attitude of the aircraft, and the last one is used to control the velocity. This is achieved by moving the control stick and the throttle.

To simplify the analysis due to limited available data, these four control variables are employed in the simulation model. The output of the simulations will determine the quality of each of the solutions.

The analysis starts by producing a population of possible solutions. In each population, there are *n* strategies or solutions. A strategy is actually an instruction for the aircraft to change its aileron, elevator and rudder deflection angles, and

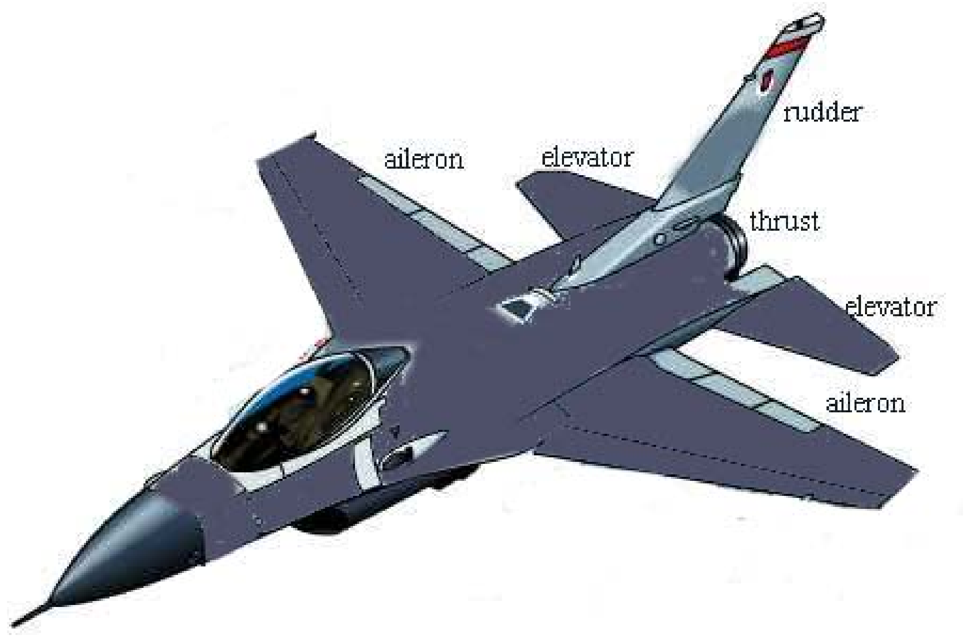


Figure 5.8: Control surfaces of an F-16 aircraft.

throttle setting at every second.

For example, at $t = 0$ s, the aircraft is instructed to deflect δ_a degrees of aileron, δ_e degrees of elevator angle, δ_r degrees of rudder angle, and δ_t percent of throttle setting. To represent a strategy in a computer program, the aileron angle deflection (δ_a), elevator angle deflection (δ_e), rudder angle deflection (δ_r) and the throttle setting (δ_t) have to be coded. For a typical jet aircraft, the deflection limit, rate limit and the time constant are given in Table 5.3.

Table 5.3: Control Surface Actuator Limits and Constants.

	Deflection Limit	Rate Limit	Time Constant
Aileron	$\pm 21.5^0$	$80^0/s$	0.0495s lag
Elevator	$\pm 25.0^0$	$60^0/s$	0.0495s lag
Rudder	$\pm 30.0^0$	$120^0/s$	0.0495s lag

These codes are made possible by determining the maximum permissible range for the control variables. A discrete angle interval of 4.78° is used for aileron and 5.56° for elevator and 6.67° for rudder. Throttle setting range is between 0 and 1 with a discrete interval of 0.1. In total, 10000 possible combinations can be generated. In a tabular form, Table 5.4 shows the coding of heading angle change, flight path angle change and throttle setting.

Table 5.4: Encoding the control surface deflections and throttle setting.

ID	Aileron Deflection Angle (δ_a), deg	Elevator Deflection Angle (δ_e), deg	Rudder Deflection Angle (δ_r), deg	Throttle Setting
1	-21.5	-25	-30	0.0
2	-21.5	-25	-30	0.1
3	-21.5	-25	-30	0.2
⋮	⋮	⋮	⋮	⋮
10000	21.5	25	30	1.0

08094056090900205560094210974105998067040594905185...
08084004910963000744009980429301850047900045206432

Figure 5.9: An example of coded path for 20 second spawn time.

Instead of directly using the angles, the strategy uses the values of IDs as shown in Table 5.4. A series of numbers valued between 0001 and 10000 are randomly constructed such as shown in Figure 5.9 with $m \in \mathbb{N}$ five-digit integers were ordered in series. The first value is 08094, which means to deflect the aileron to 7.78°, the elevator to $-10.0°$, the rudder to 5.0° and set the throttle to 33%. Next is 05609 which means deflecting the aileron to 1.11°, the elevator to 3.33°, the rudder to $-5.0°$ and the throttle to 89%. By skipping to the last figure in the line of Figure 5.9, 06432 means to deflect the aileron to 3.33°, the elevator to $-1.11°$, the rudder to $-1.67°$ and the throttle to 11%. The whole process is called the trajectory/path of the aircraft or a strategy of a 20 second spawn time.

$n \in \mathbb{N}$ strategies are generated to represent a population. The number of population used in this study is $n_{gen} \in \mathbb{N}$.

5.8 The Objective Function

The overall performance of the game, J, is measured in Eq. 5.38. It fulfills the criteria that at all time no constraint is violated and the distance between the evader and the pursuer is never less than *interception distance*, d_{int}.

$$J = \left[\left(\sqrt{(x_e(t) - x_p(t))^2 + (y_e(t) - y_p(t))^2 + (z_e(t) - z_p(t))^2}\right)_{min} - d_{int}\right] \cdot p_g \cdot t_f \quad (5.38)$$

where subscript e stands for the evader, subscript p stands for the pursuer, (x, y, z) is the vehicle's position relative to inertial frame, t_f is the final time and p_g is the penalty or the reward given. The fitness value J represents the distance between the pursuer and evader during the simulation period.

The aim of the optimisation is to find a feasible solution that maximises J. A scalar penalty is given to a solution if it violates any of the constraints or gets the evader captured. The penalty will result in the fitness value of the solution to be lowered. A solution will receive a reward and a high fitness value if no constraint is violated and it does not cause the evader to be intercepted or if the pursuer crashes during the game. A solution, T_e^*, is feasible for the evader if

$$J(T_p, T_e, t, X) \leqslant J(T_p, T_e^*, t, X) \tag{5.39}$$

where T_p is the strategy of the pursuer, T_e is the strategy of the evader, for all t and X are the states of the players.

For realistic modelling, several constraints have to be considered. For the evader, the constraints are the permissible range of the angle of attack. The range is between $-10°$ and $40°$.

$$-10° \leqslant \alpha_e \leqslant 40° \tag{5.40}$$

The altitude must not be less than 500 m (Eq. 5.41) with a flight path angle not less than $-30°$ (Eq. 5.42) from the horizon to avoid from hitting the ground and to allow the evader to recover its altitude.

$$h_e \geqslant h_{min} \tag{5.41}$$

$$\gamma \geqslant -30° \tag{5.42}$$

5.9 Integrating the Simulation Model Into EP

The developed simulation model discussed in the previous sections resembles the time-based simulation model with predefined control inputs between two players. There is no intelligence in the simulation model. The model simply simulates a game between two players with a given set of control inputs. The outcome of the game matters. If the set of control inputs give a good outcome, the control inputs (i.e. the solution) will be selected.

The intelligence is due to the way EP selects and manipulates solutions. In order to find feasible solution, EP uses the developed 6DOF mathematical model to find the feasible solution through a game. Although, the method sounds trivial, the way EP manipulates solutions could somehow find the feasible solution in a highly multi-dimensional space.

Initially, EP will generate a set of solutions randomly. The solutions are used by the evader to avoid capture. The players start at a predetermined position and all players know the state of the other player at the current time but not in the future.

Next, EP will evaluate each of the solutions through a game. The evader will use the given solution during the game and the pursuer will use conventional navigation guidance system to get to the evader. After a predetermined time such as 40 seconds, the game is stopped. The time frame is chosen not to be too long in the effort to reduce the computation time, but at the same time allows reasonable time to find a feasible solution. If the evader is captured during the game, or if the evader exceeds one of the constraints, the solution is considered bad. But, if the evader was able to out-manoeuvre the pursuer and passes all of the constraints, the solution will be deemed good or feasible.

The feasible and infeasible solutions (parent population,) are categorised, sorted and mutated before inserted into the child population as explained in Chapter 3. The child population is then being evaluated to find their 'fitness'. Ultimately, the selection is performed using a tournament or an elitism approach between the child and parent population. The n solutions that have the highest fitness values will be selected to be in the next population.

EP is stopped after reaching a predetermined number of generations. The solution that has the highest fitness value is selected as the feasible solution. The next question is, how fast is the algorithm. If a single processor is not fast enough, could a multiple-processor achieved the same result with less computing time?

This is achieved by coding the simulation using FORTRAN 90 programming language. The random number generator used to generate the first population of strategies is RANDLIB90 [17].

To improve the execution time, in some cases it is useful to store the solution information in computer memory to avoid duplication which leads to waste of computer resources due to unnecessary re-evaluations [34]. But, by doing so, a substantial size of memory has to be allocated and an algorithm to compare newly created solutions with the previously generated ones has to be developed. Such an approach will undoubtedly increase the computing time and, ironically, make the evaluation more expensive. If the dimension of the solution is high, the probability of producing duplication will be small leading to notion that it is unnecessary to remember the already generated solutions. This proposition discussed above will hopefully lead to a faster execution of the algorithm.

5.10 Parallel Implementation

The use of parallel computing is to improve the computing time and the quality of the solutions. The benchmark is usually from the results gathered by running the simulation on a single processor. As discussed in Chapter 4, there are two approaches used to improve the solution's fitness and the computation time, (i) the master-slave approach and (ii) the coarse-grained approach. The difficulty level of implementing both approaches is high since each approach has different styles of coding. Thus, two sets of codes are developed specifically for the 3DOF and 6DOF models.

The communication among processes is performed using MPI library [8; 116]. Historically, MPI which stands for Message Passing Interface is a standard that was designed by a group of researchers from academia and industry. The interface is designed to function over a wide variety of computers. There are several successful implementations of MPI. The initial implementation of MPI 1.x standard was called MPICH designed by Argonne National Laboratory (ANL) and Mississippi State University. The current release is MPICH2 which is based on the MPI 2.1 standard.

Another MPI implementation is OpenMPI. Originally, OpenMPI was a combination of several MPI implementations such as FT-MPI, LA-MPI, LAM/MPI and PACX-MPI.

This project uses the MPICH2 library in FORTRAN to communicate between processes. MPICH2 is readily available for Intel FORTRAN compilers, thus it is a natural candidate for use in this project. MPICH2 is able to run on both Linux and Windows operating systems.

The MPI functions used are MPI PACKED, MPI UNPACKED, MPI ISEND, MPI RECV and MPI WTIME. MPI PACKED is used to pack several variables of different types into a single unit before being sent to another processor using

MPI ISEND. On the other hand, MPI RECV is used to receive data from other processes. Then, the communication is tracked using MPI WAIT. At the same time, the processing time at various levels is tracked using MPI WTIME. All of these functions are implemented using the FORTRAN programming language.

5.11 Overall Implementation

The mathematical model and EP are coded in FORTRAN programming language. The compilers are Intel Fortran Compiler and IBM XL Fortran Compiler with parallel capability. The parallel library used is MPI. The hardware is two Linux clusters operated by the Victorian Partnership of Advanced Computing (VPAC). This was made possible through an e-research grant awarded by VPAC (EPANRM153).

The optimisation process is illustrated in the flow charts given below. The master-slave model flowchart is given in Figure 5.10. The coarse-grained model flowchart used in 6DOF model is given in Figure 5.11. The simulation of the air combat scenario over a period of *n* seconds is illustrated in Figure 5.12.

5.12 Summary

In this chapter, the development of the objective function is discussed. Two aircraft models are used, i.e. three degree of freedom and six degree of freedom models. The three degree of freedom model uses simplified aerodynamic data and does not consider the aircraft attitude motion. The EP implementations are paralleled using two approaches, i.e. master-slave and coarse-grained. More info to complete the use of these two models. The next chapter will present the results of the numerical simulations based on the models used.

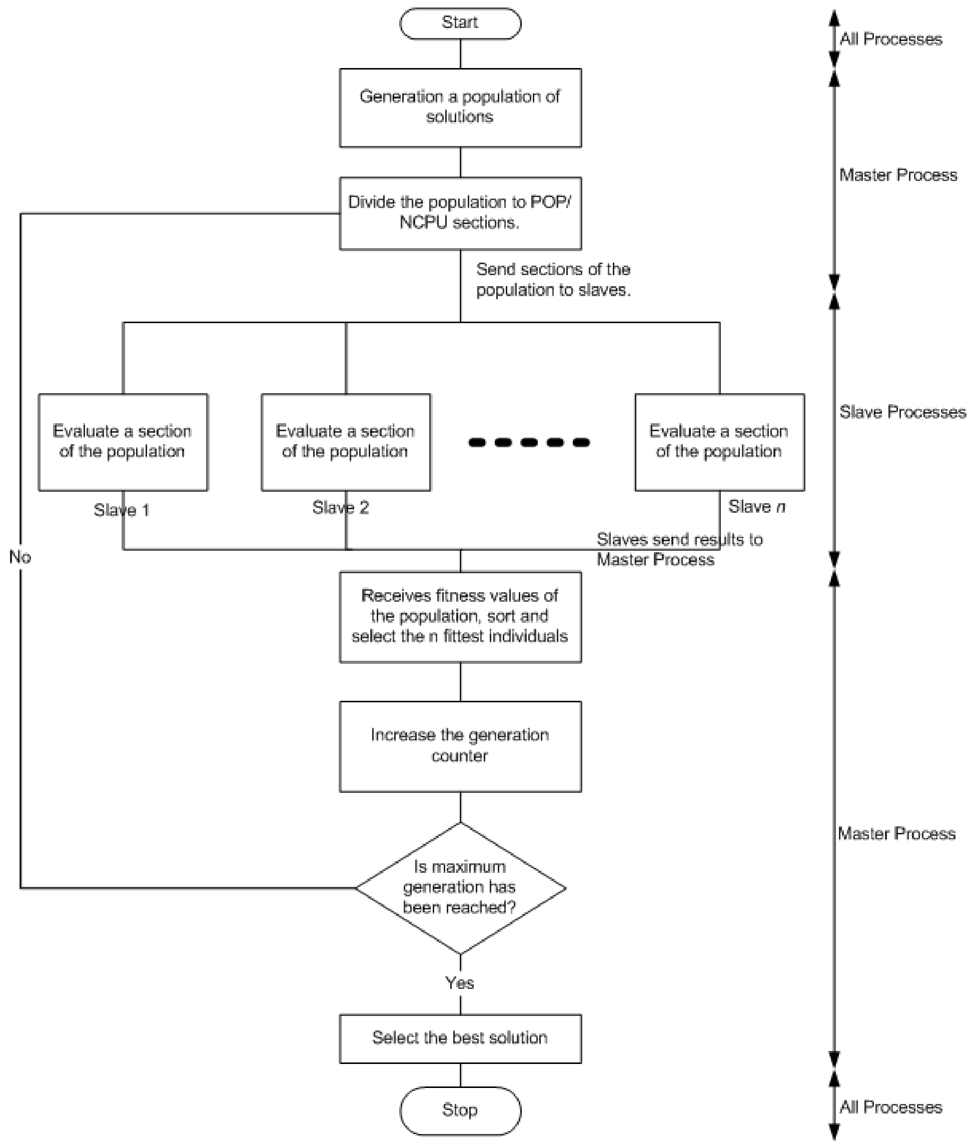

Start
Generation a population of solutions
Divide the population to POP/ NCPU sections.
Send sections of the population to slaves.
Evaluate a section of the population
Evaluate a section of the population
Evaluate a section of the population
Slave 1
Slave 2
Slave n
Slaves send results to Master Process
No
Receives fitness values of the population, sort and select the n fittest individuals
Increase the generation counter
Is maximum generation has been reached?
Yes
Select the best solution
Stop
All Processes
Master Process
Slave Processes
Master Process
All Processes

Figure 5.10: Flowchart of the Master-Slave implementation.

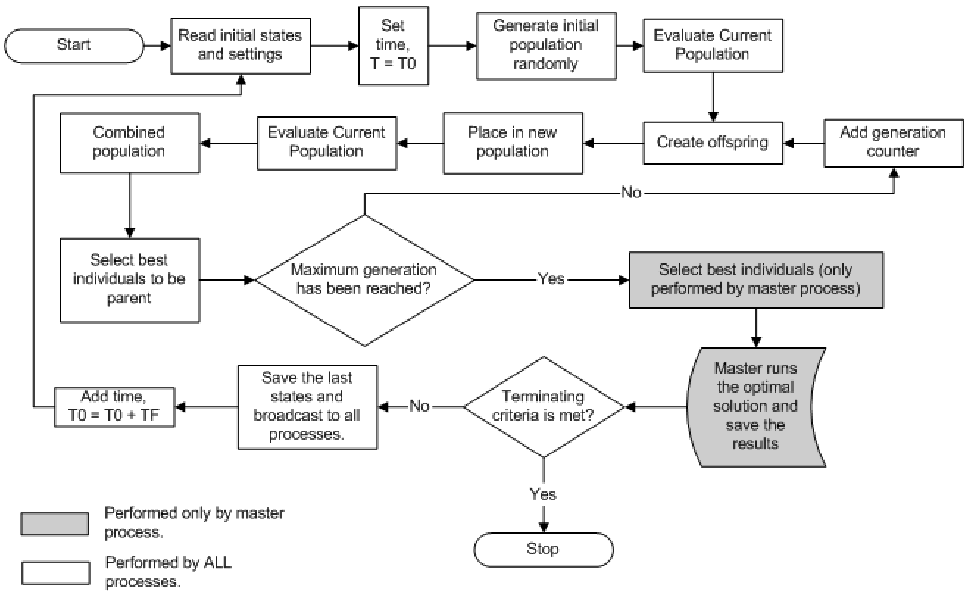

Start
Read initial states and settings
Set time, T = T0
Generate initial population randomly
Evaluate Current Population
Create offspring
Add generation counter
Place in new population
Evaluate Current Population
Combined population
Select best individuals to be parent
Maximum generation has been reached?
No
Yes
Select best individuals (only performed by master process)
Master runs the optimal solution and save the results
Terminating criteria is met?
No
Save the last states and broadcast to all processes.
Add time, T0 = T0 + TF
Yes
Stop
Performed only by master process.
Performed by ALL processes.

Figure 5.11: Flowchart of the Coarse-Grained implementation.

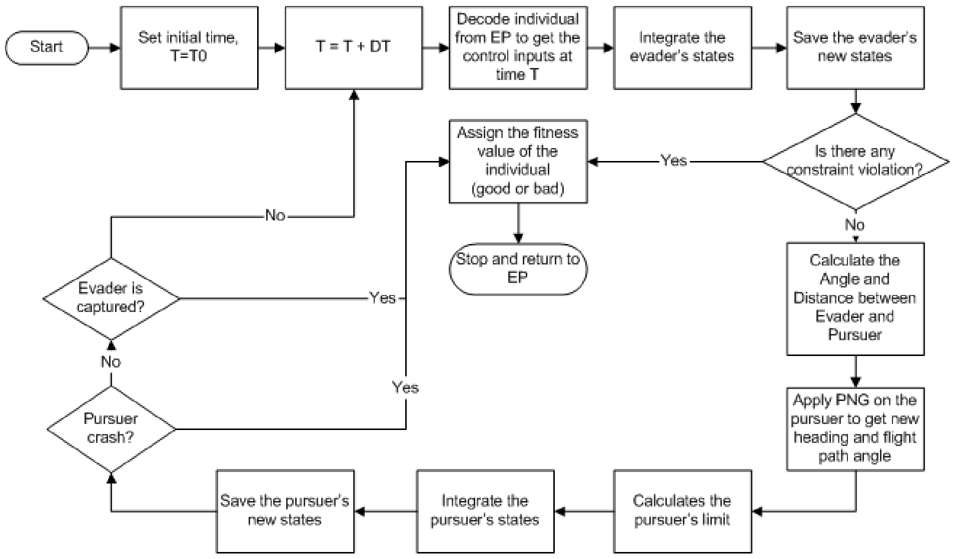

Start
Set initial time, T=T0
T = T + DT
Decode individual from EP to get the control inputs at time T
Integrate the evader's states
Save the evader's new states
Is there any constraint violation?
Yes
Assign the fitness value of the individual (good or bad)
Stop and return to EP
No
Calculate the Angle and Distance between Evader and Pursuer
Apply PNG on the pursuer to get new heading and flight path angle
Calculates the pursuer's limit
Integrate the pursuer's states
Save the pursuer's new states
Pursuer crash?
Yes
No
Evader is captured?
Yes
No

Figure 5.12: Flowchart of the Air Combat Simulation.

6 Numerical Results and Discussion

6.1 Introduction

This chapter discusses the performance of EP in solving pursuit-evasion problems. Several aspects of the performance of EP are investigated. These include the effect of EP parameters and the effect of the initial conditions on the quality of the solutions.

The search for a feasible control for 6DOF problems is done by using FORTRAN programming language. The type of FORTRAN used is Intel FORTRAN 90 on Intel-platform and IBM FORTRAN on PowerPC-platform. Since the outcome of the game is very sensitive to the initial conditions as well as the nature of the game, thus, the global optimal solution is not known *a priori*. The complexity of the problem is further increased when noise and errors are added.

The analysis begins by searching for a feasible control solution for several scenarios experienced by the evader. The Monte Carlo analysis is conducted to validate the robustness of the algorithm. The initial population of control inputs are randomized appropriately. For the same initial conditions, the robustness of the solution is studied using Monte Carlo analysis. The benefit of using EP is studied by comparing the results from a purely random search. Finally, the benefit of parallel implementation is studied.

6.2 Aircraft Data and Configuration

6.2.1 The Evader

The evader is modeled after the 6DOF model based on a generic fighter aircraft as given in Appendix A.1.

6.2.2 The Pursuer

The pursuer is only modelled using the 3DOF due to the complexity of the control system design if the 6DOF model is employed for the pursuer. Using the 3DOF model simplifies the control system of the pursuer. In the effort to make the dynamic of the pursuer as close as possible to the real scenario, the aerodynamic constraints are employed. The constraints are maximum turning rate, minimum turning radius, maximum angle of attack, fuel limitation and maximum thrust. The engine performance is discussed in Section 5.6.5.1.

6.3 The Hardware

Two computer clusters were used in the analysis. The clusters are part of several clusters operated by the Victorian Partnership of Advanced Computer (www.vpac.org). The first cluster, named BRECCA, consists of 97 nodes of dual Xeon 2.8 GHz CPU. The operating system is Fedora Core Linux. The communication between nodes is via Myrinet. Myrinet is a high performance network communication technology used widely in computing clusters. This cluster uses Intel Fortran Compiler with MPICH2. Further information on this cluster can be found on its website at **www.vpac.org**.

The second cluster, named EDDA, consists of 184 Power5 CPUs. There are 46 nodes of 4 CPUs each. 23 nodes have 16GB of RAM and another 23 nodes have 8GB of RAM. The operating system is SUSE Linux. This cluster uses IBM High Performance Fortran Compiler with MPICH2 library. Further information on this cluster can be found at the website at **www.vpac.org**.

6.4 Computing Time

The execution time taken by EP can be divided into several segments:

1. Pre-processing - to create the initial solutions and to read the initial parameters.

2. Communication among processes.

3. Evaluation of solutions.

4. Selection and recombination.

Item 1,2 and 4 are usually fast and consume minimal time and the longest execution time is taken by item 3. The evaluation time for 6DOF problem is relatively fast on EDDA with an average of 0.142s with the standard deviation of 0.03875 and sample size of 8000, as depicted in Figure 6.1.

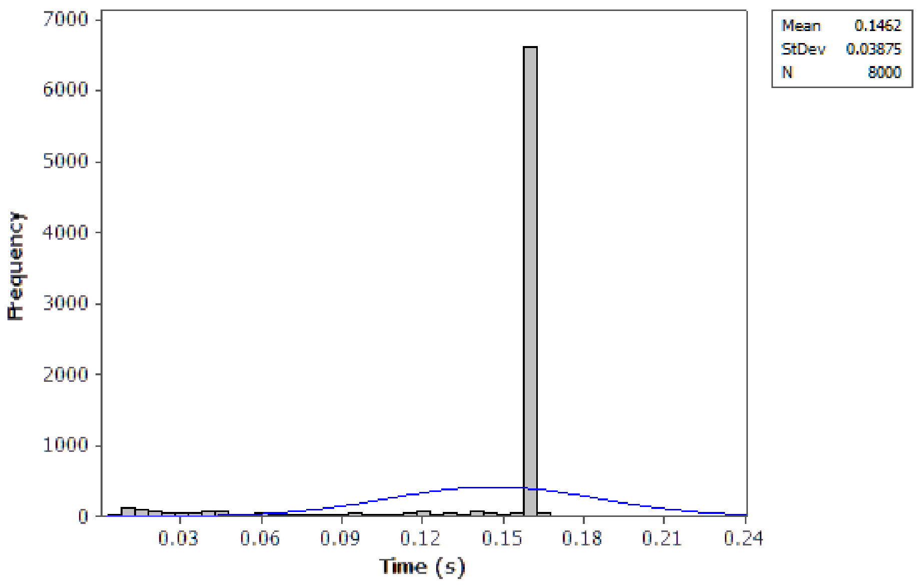


Figure 6.1: Evaluation times for a 6DOF simulation by each processor on EDDA.

6.5 Pursuit Evasion Games

The performance of the algorithm is tested by running a number of simulations. Several scenarios are considered. These scenarios are selected based on the established literature. In order to find a feasible solution for the evader during an air combat between two players, it is assumed that the players know the capability of its adversary. This includes the aerodynamic limitations, performance limitations and guidance system limitations, which are highly nonlinear and discrete. The search is performed by initially generating a population of candidate solutions. A solution is a series of instructions of the control variables (the aileron, the elevator, the rudder and the throttle) over a period of time. Each solution is evaluated in a virtual game. The outcome of the game is the fitness value of the solution. It takes about 0.17 seconds to evaluate one solution for a simulation of 100 s. Thus, thousands of possible solutions can be tested within a relatively short period of time.

Finding ways to exploit the weakness of the missile analytically is very difficult to do due to the reasons discussed in the previous section. Furthermore, the exploitation of the weakness is highly time-dependent and the opportunity to defeat the missile only exists for a very short period of time [76]. In order to grab this window of opportunity, it usually requires the evader to move optimally from the very beginning. However, such a requirement is very difficult to implement especially if one resorts to try and solve it in real time.

In this section, the pursuit-evasion encounter between two players is explored. One player is a missile which acts as a pursuer. The other player is a generic fighter aircraft and acts as the evader. The missile is launched from a distance from another aircraft. The evader detects the launch and tries to defeat the missile by following a series of manoeuvres.

For a particular type of missile with a given pursuit-evasion game, it is difficult to predict whether the pursuer can always capture the evader [98]. It is possible that when a game starts in a certain states space, the evader can avoid interception. This is known as the escape zone of a game. Analytically, there is a barrier that separates escape and no-escape zone as defined by Isaac [59]. Fortunately, most modern day missiles do not behave optimally because they are guided by an intelligence guidance system. The missile has physical limitations, and if known, one could exploit these weaknesses to the fullest extent.

Another aim of this research is to study the performance of the Evolutionary Programming algorithm to find feasible solutions for the evader. However, not all possibilities are explored in this thesis. Only some selected scenarios and conditions are being explored. As explained earlier in this chapter, the missile parameters are only estimation and may not represent any missile already in service. The difficulty of finding the information hampers the initiative to model the missile as accurately as possible. The aerodynamic coefficients, maximum load factors, the amount of fuel and the rocket motor thrust are only estimates. Efforts have been made to ensure the performance of the missile is comparable to the actual one.

Stevens [118] has prepared a working six degree of freedom jet fighter model written in FORTRAN. A modifications on the source code is required to suit it with the optimisation algorithm. The aerodynamics and performance limitations of the 6DOF model given by [118] are incorporated in the optimisation algorithm so that the dynamics of the aircraft will behave as close as the real one.

The performance is measured in terms of: the size of the population, the total number of populations and the number of processors used. Apart from that, the initial states are also play a very important aspect in determining the ability of the algorithm to find feasible solutions. Besides that, the evader also has a complete knowledge of the pursuer's capability.

The initial conditions are based from Breitner [68] and listed in Table 6.1 and 6.2. The simulations only display the encounter for 40 seconds. If no interception is made within the 40 seconds time-frame, the evader will be assumed to have survive the encounter.

Table 6.1: Initial Parameters for the pursuer and the evader.

Parameter	Pursuer	Evader
X,m	0.0	5000.0
Y,m	0.0	0.0
Z,m	10000.0	10000.0
Velocity,m/s	250.0	150.0
		Tail,Side,Head
Heading Angle,rad	0	0,1.57,3.141
Interception Radius, m	15	-
Max Load Factor	22	9

In a preliminary run, the algorithm was able to find feasible solutions. A game is considered good if the pursuer experiences two situations, i.e. it crashes to the ground or simply unable to intercept the evader. In most cases, the easiest way to

Table 6.2: General Parameters.

Parameter	Value
No. of Generations	50
Size of population	100
Migration Rate	10
No. of processors	4
Spawn Time	10 s
Duration	max 40 s

defeat the pursuer is to fly close to the ground and performing manoeuvres that will eventually drive the pursuer to the ground.

The study considers the three scenarios as depicted in Figure 6.2. The first scenario is when the missile is launched from behind the evader, i.e. the evader's heading angle (horizontal flight path angle), ψ , is 0°. The second scenario is when the missile is launched from the port side of the evader, i.e. the evader's heading angle, ψ , is 90° and the third scenario is when the missile is launched from the front, i.e. the evader's heading angle, ψ , is 180°. These scenarios are based from the work of Breitner [101]. The results of his study are shown in Figure 6.3. He considered 7 initial heading angles in his study. They are 0°, 30°, 60°, 90°, 120°, 150° and 177° (his algorithm may not work if the heading angle is set to 180°). Although, the flight paths were optimal, the missile was still able to intercept the evader.

In all scenarios, the evader has to find feasible solutions. The search for feasible solutions is performed in parallel using Linux cluster owned by VPAC (http://www.vpac.org) under the research grant EPAN.RM.153. The three scenarios are described in detail in Section 6.5.1 - 6.5.4

In each scenario, the optimisation simulation run is repeated 100 times. Each run is independent between each other. The results are calculated from the average of the independent runs. The results are the execution time, and the time to find the first feasible solution. Three independent runs from each scenario are arbitrarily selected and discussed in the following sections. These runs are numbered 1, 2 and 3.

Generally, any feasible solution can be used to evade interception but if the computing is fast enough, the best feasible solution is the one that has the highest fitness value. In an air combat, the survival of a pilot or an aircraft depends on how fast one could react optimally in a close air combat. If the computing time is considerably long, the first feasible solution has to be accepted.

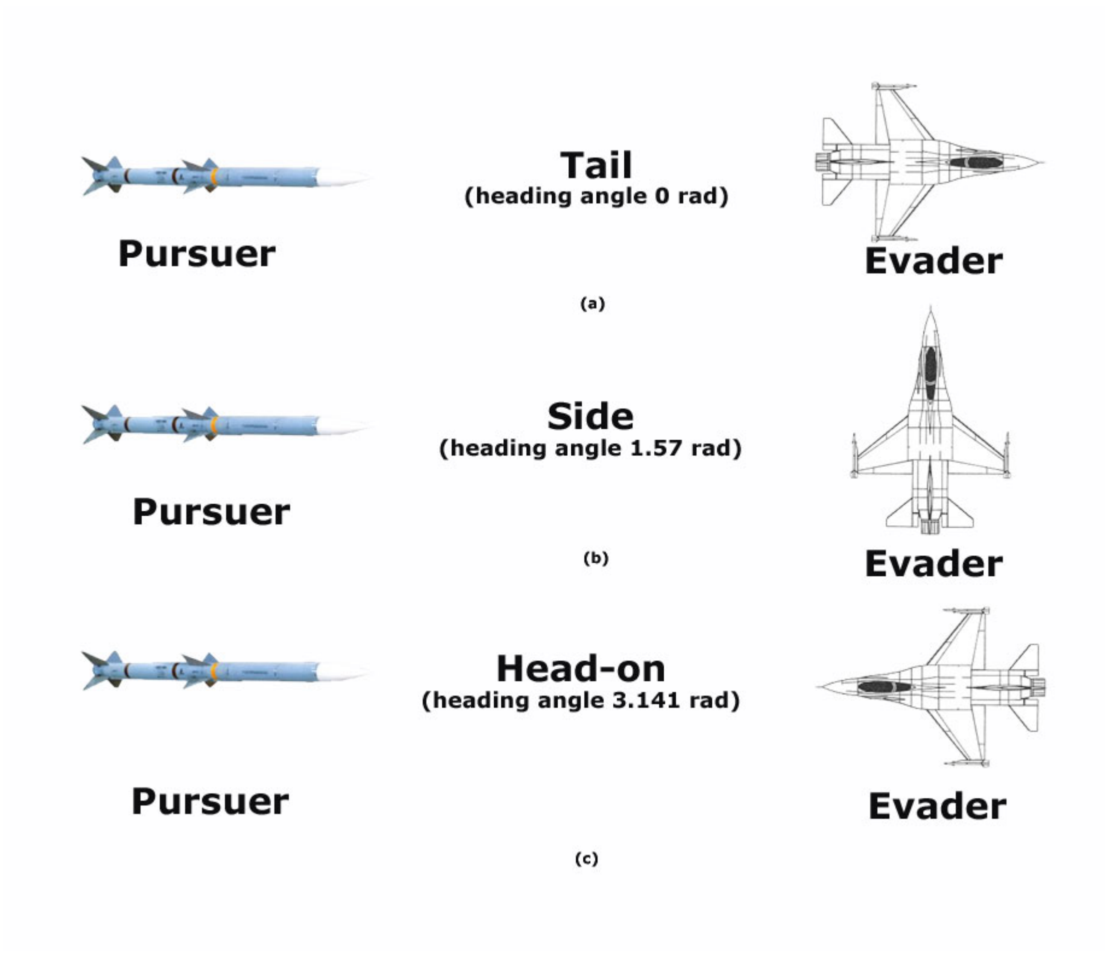


Figure 6.2: Three missile-aircraft encounter scenarios studied: rear attack, side attack and head-on attack.

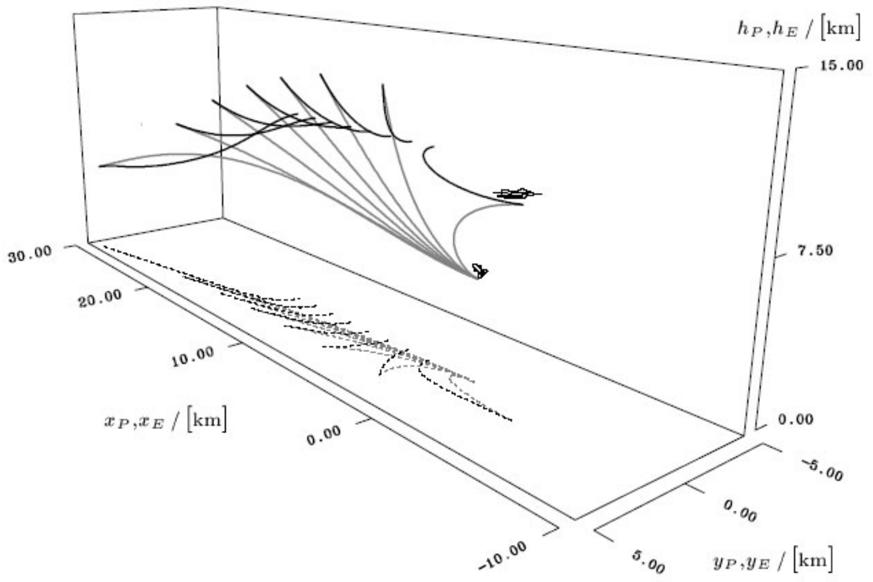


Figure 6.3: Optimal flight paths for 7 heading angles calculated by [68]. Flight paths of the pursuer are gray and the evader are black. Other initial conditions are similar.

6.5.1 Scenario 1: Head to Tail Encounter

This section discusses the result for the first scenario. In this scenario, the pursuer is located behind the evader. The evader has a good chance to escape interception. A total of 100 independent runs were conducted. The evader is a manned jet fighter aircraft. The maximum load factor is set to $9g$.

The fitness values, J is given in Figure 6.4. It can be seen that EP is able to find feasible solutions as earliest as in the fourth generation which was found by processor 1. Since each processor independently searches for the feasible solution, thus they find feasible solutions at different number of generations.

The average execution time for this scenario is 235 seconds (3 min 55 sec) with a standard deviation of 44 seconds. Although it can be considered fast when compared to other classical optimisation algorithms, it is not fast enough for real time application.

One interesting aspect of EP is that it searches for feasible solutions and it improves feasible solutions found until the termination criteria, i.e. the maximum number of generations has been reached. If the fitness value of the feasible solution is not significant, the first feasible solution found is adequate to be chosen as

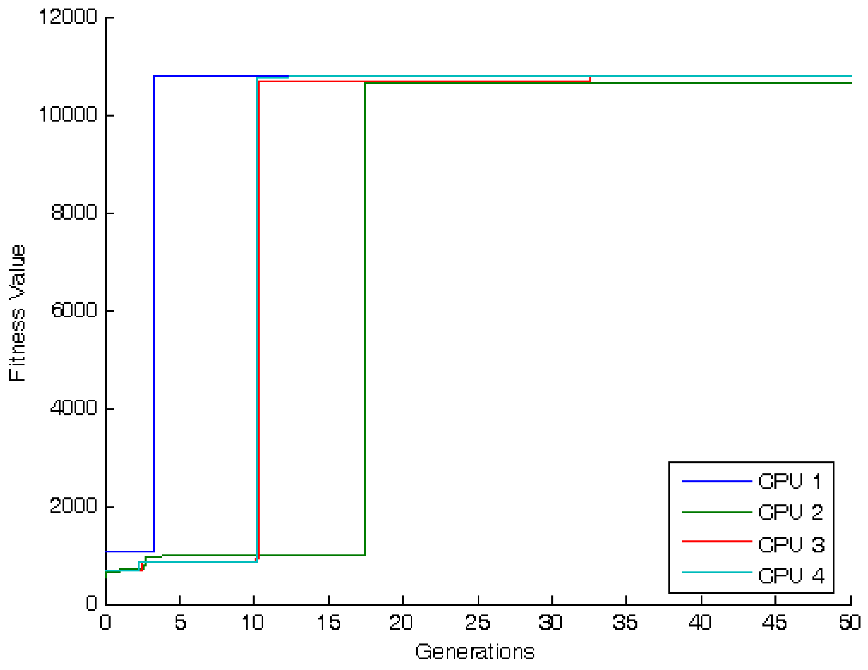


Figure 6.4: Fitness values for scenario 1.

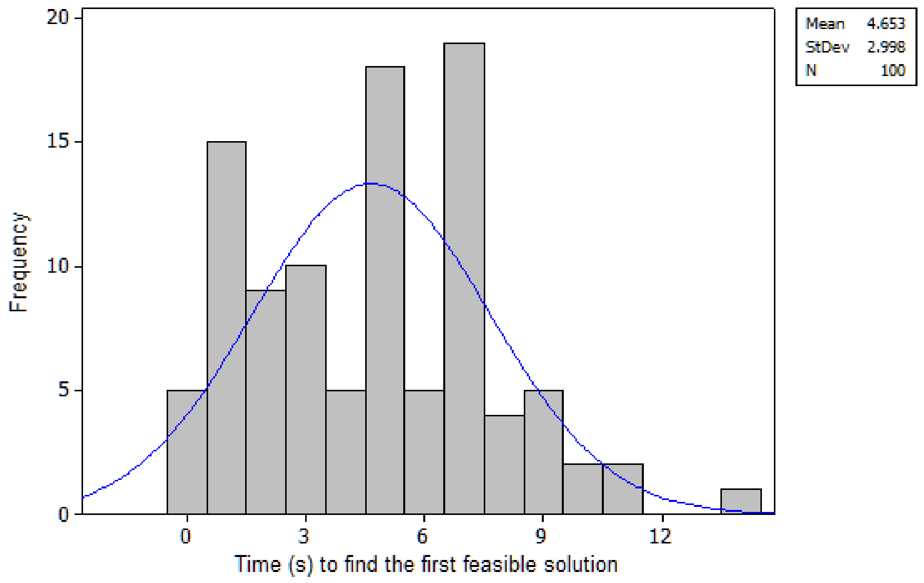


Figure 6.5: Computing time to find the first feasible solution for scenario 1.

the best as long as it allows the evader to avoid interception, albeit not with the highest fitness value. Figure 6.5 shows the histogram of the time taken to find the first feasible solution. On average, it takes 4.65 s to find the first feasible solution.

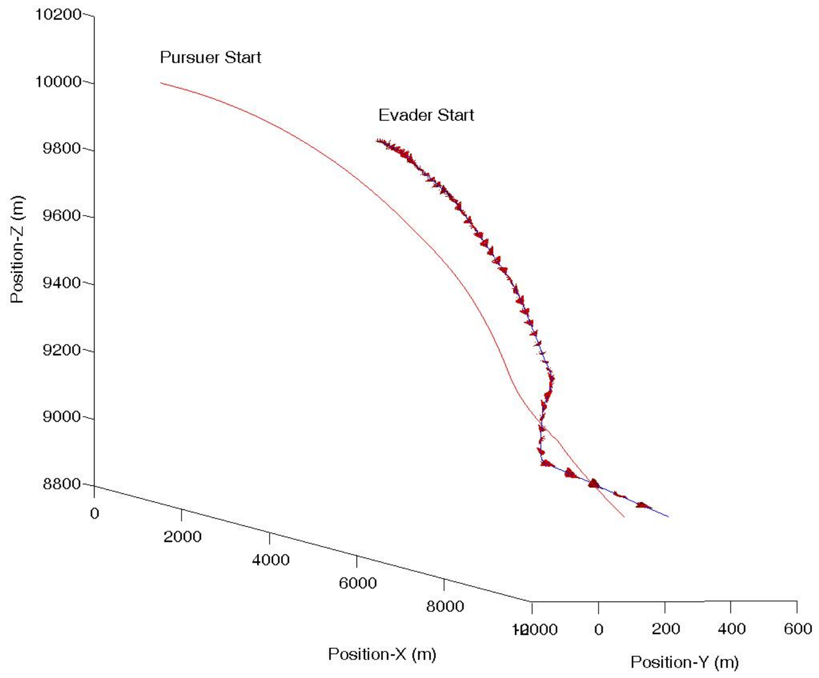


Figure 6.6: 3D view of the encounter in scenario 1 (run 1) .

Figure 6.6 shows the 3D view of the encounter of feasible solutions from three independent runs. It can be seen that all three display a similar pattern of manoeuvre. The manoeuvres are basically diving and performing rolling scissors until the pursuer misses the interception opportunity.

From the start of the encounter, the evader dives to a lower altitude and performs a weaving manoeuvre. The pursuer follow-suits using PN guidance system. When the separation distance is closer, i.e. at time 14 s, the evader performs a high-g manoeuvre up to 7-g to evade interception. The manoeuvre closely resembles a manoeuvre commonly known as a rolling scissor [103]. At that point, the pursuer's g-load is already at its maximum at 22-g. This timely manoeuvre performed by the evader saves it from interception. The pursuer then overshoots the evader.

The timing to execute a particular manoeuvre is important as illustrated in Figure 6.9. The reason is because the opportunity to act is only for a very short period of time and the manoeuvres are only "optimal" when performed during

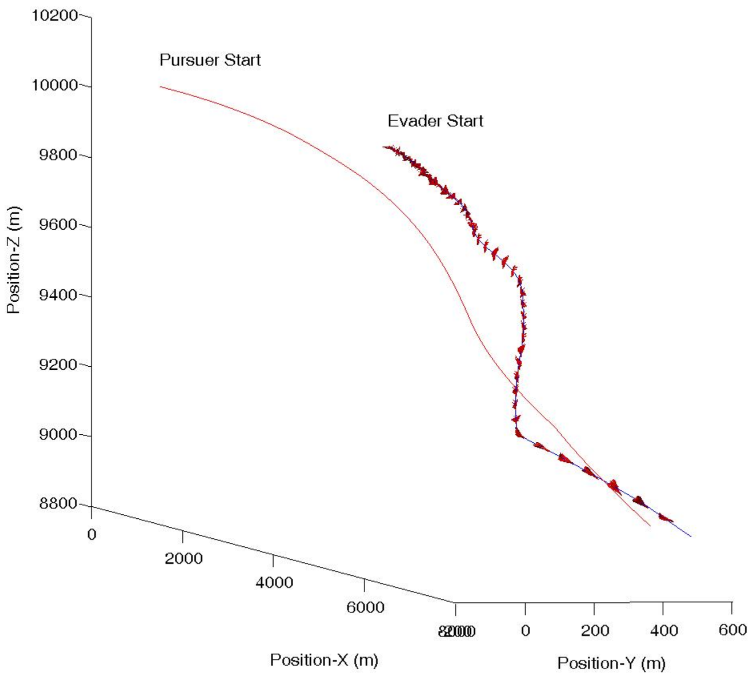


Figure 6.7: 3D view of the encounter in scenario 1 (run 2) .

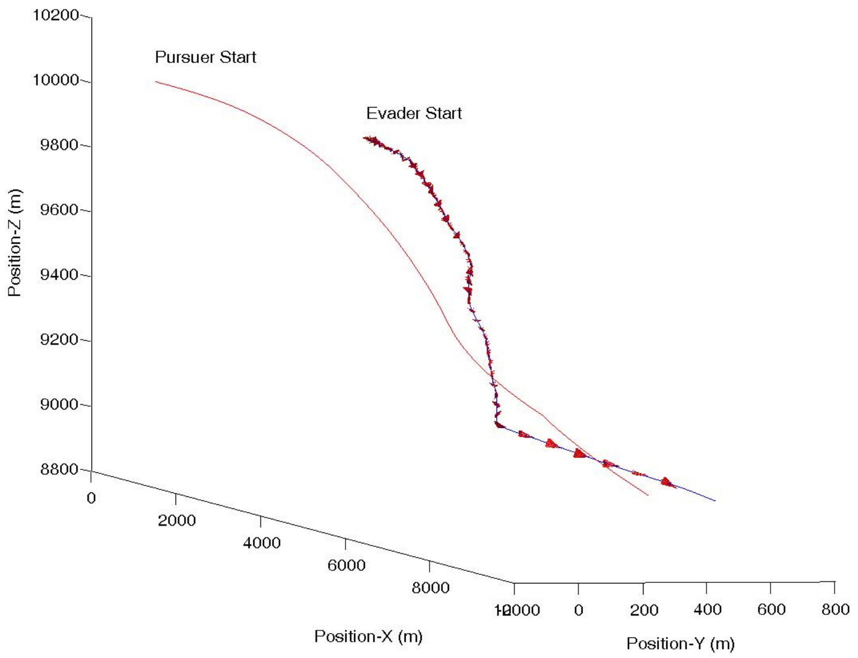


Figure 6.8: 3D view of the encounter in scenario 1 (run 3)(.

that period else it will not be beneficial to the evader.

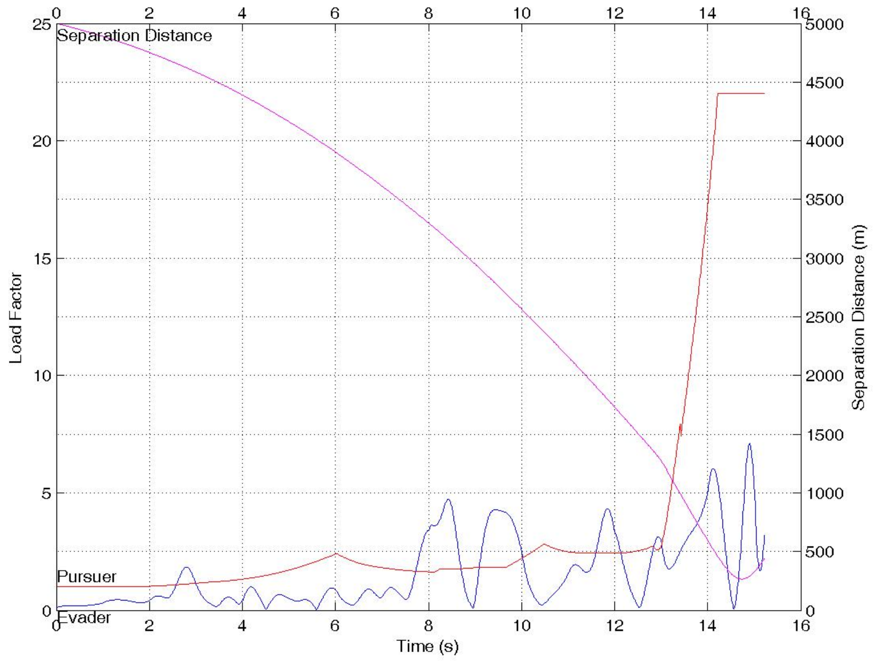


Figure 6.9: Load factor and separation distance of the players for scenario 1.

As seen in Figure 6.9, initially both players experience a small load factor. This is due to the larger separation distance between them. At time 8 s, the evader starts to perform a higher-g manoeuvre to position itself for the evasion. At the separation distance becomes smaller, at time 14 s, the evader performs a hard 6-g and 7-g manoeuvre. Such timely manoeuvres have enabled the evader to avoid interception. If the missile response lag is added, such last minute manoeuvres could result in an increase in miss distance. In this study, we assume the missile has no computing lag.

Figure 6.10 shows the encounter from XY-plane. Since the trajectory is predominantly on the z-axis, the distance covered in the x-axis is about 2800 m. From the start, the evader turns to the left and dive as shown in Figure 6.11. The evader's high g-manoeuvre towards the line of sight saturates the missile's lateral acceleration capability causing a miss [76].

The time-history plot of the velocity of the players is given in Figure 6.12. As expected, the higher thrust of the pursuer produces higher acceleration compared to the evader. But the evader achieves a better acceleration by diving which will then exploit the advantage of gravitational acceleration.

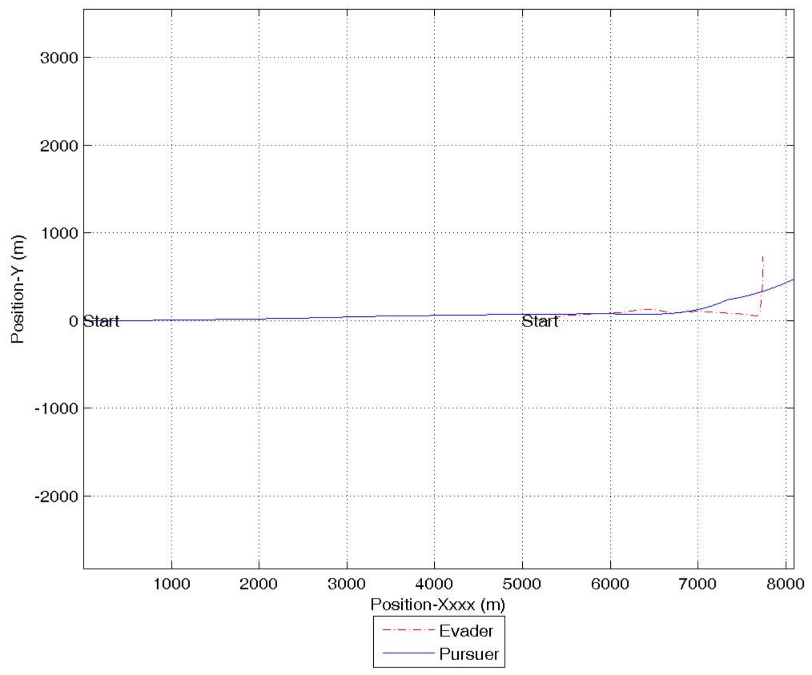


Figure 6.10: Feasible encounter in the XY-plane for scenario 1.

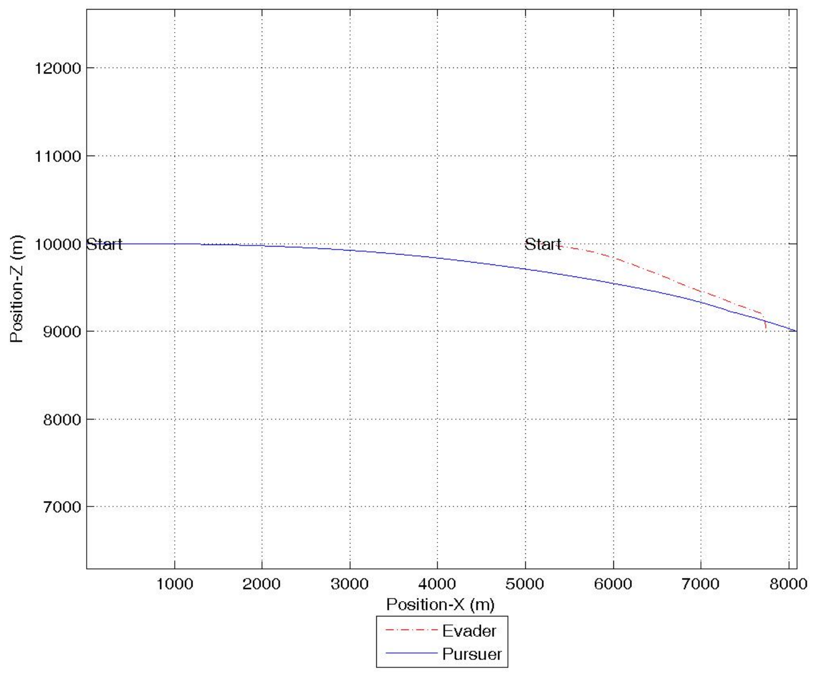


Figure 6.11: Feasible encounter in the XZ-plane for scenario 1.

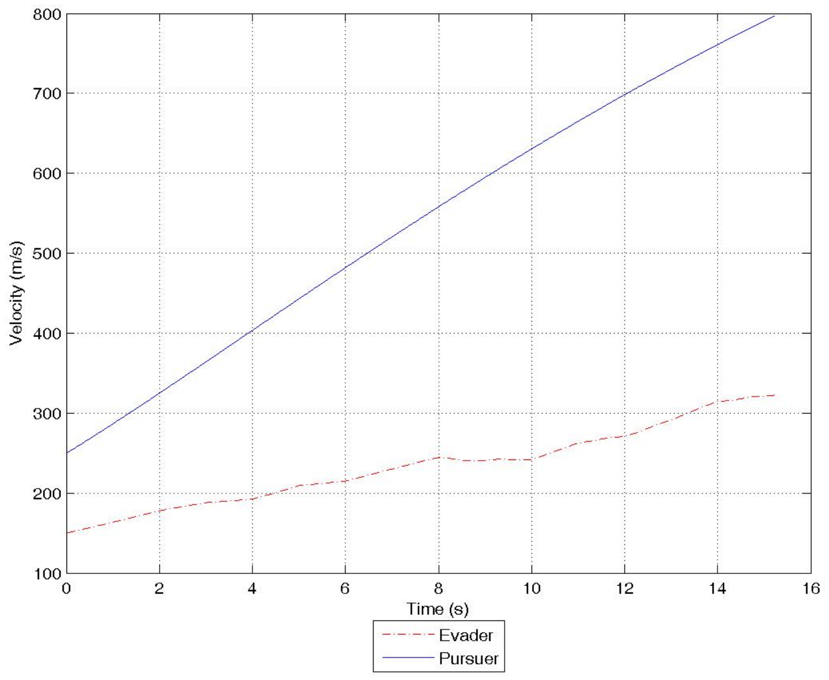


Figure 6.12: Velocity of both players for scenario 1.

The feasible control deflections are depicted in Figure 6.13 - 6.15. The aileron and rudder deflections time-histories show a high deflection angle at opposing directions in most of the time. This is required to perform rolling scissors effectively. However, the elevator deflections time-history is not as high in magnitude compared to the aileron. Only at the 12th second of the encounter, the elevator is applied to its maximum.

As expected the throttle setting (Figure 6.16) shows a bang-bang type of control between 100%, 50% and 0% of throttle settings. However, at the final part of the encounter, the evader seems to employ a maximum thrust for a successful evasion.

The throttle settings and the time-history of the velocity is shown in Figure 6.16. The throttle settings are set at the maximum level for most of the time, except at the 8th second. Prior to the game end, the evader's throttle is reduced to slow down the aircraft in its effort to improve the turning performance. The velocity profile follows the throttle setting closely. For example, at the 8th second, the velocity does not increase because the throttle is set to zero.

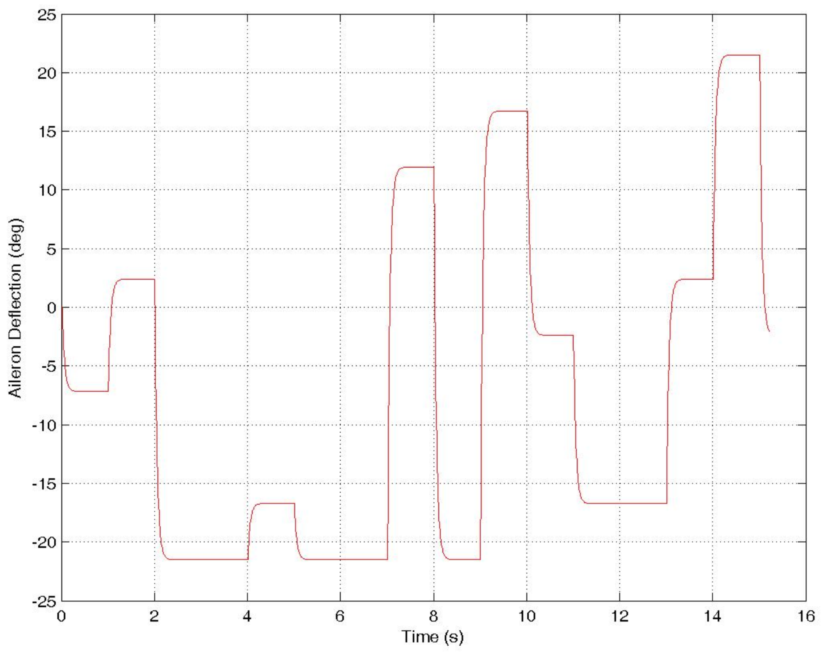


Figure 6.13: Aileron deflections of the evader for scenario 1.

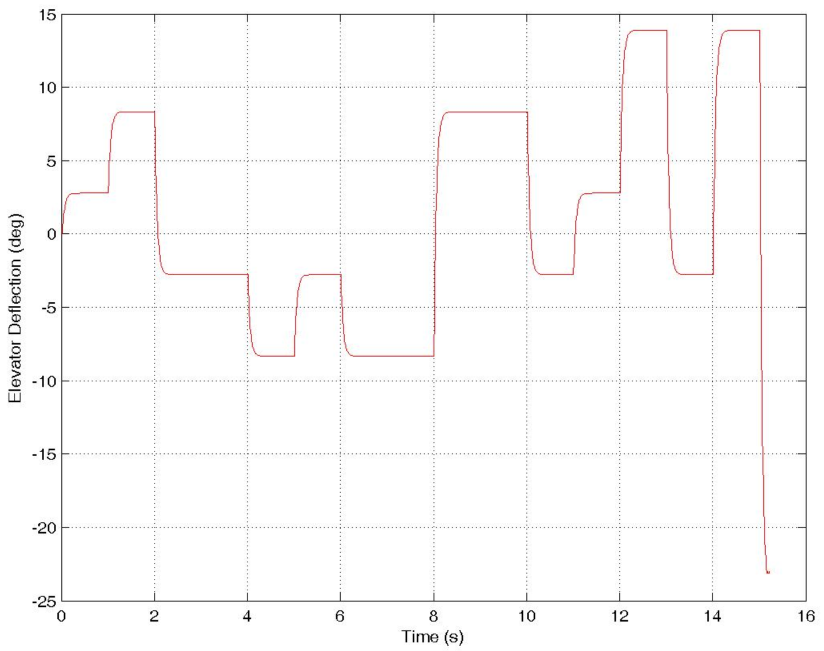


Figure 6.14: Elevator deflections of the evader for scenario 1.

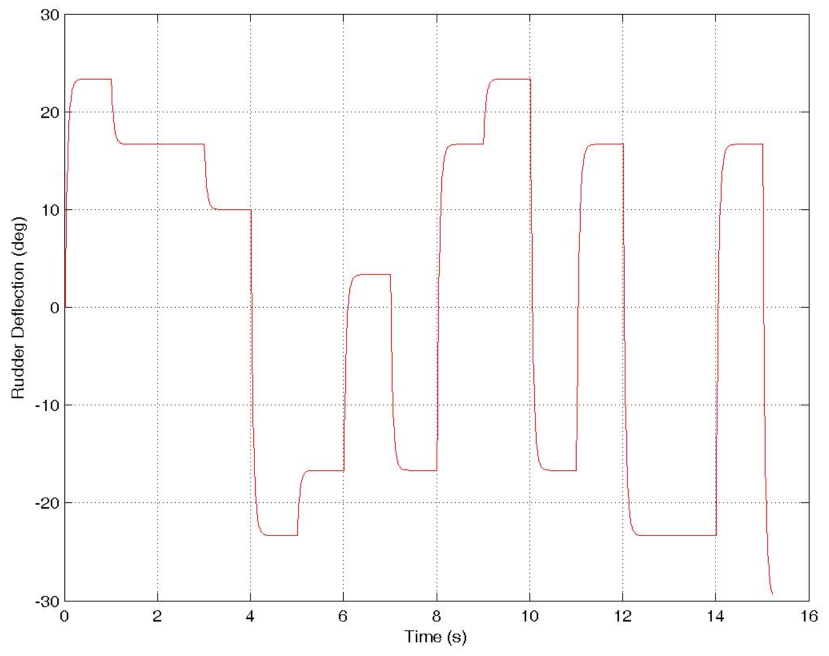


Figure 6.15: Rudder deflections of the evader for scenario 1.

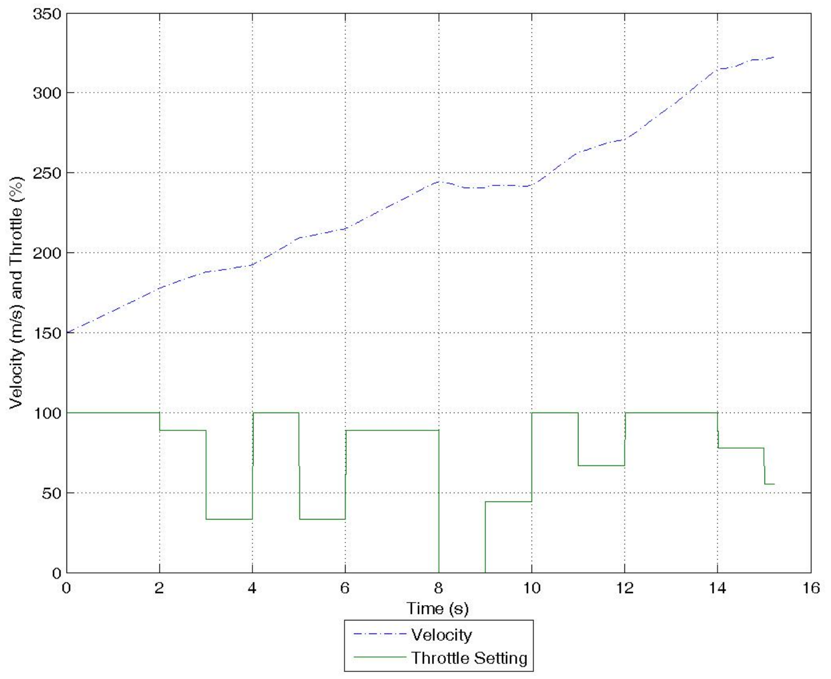


Figure 6.16: Throttle settings and velocity of the evader for scenario 1.

6.5.2 Scenario 2: Head to Side Encounter

In this scenario, the pursuer approaches the evader from the evader's port side. A total of 100 independent runs are conducted. On average, EP takes about 223 s (3 minutes and 44 s) and 9.85 seconds of standard deviation to complete a search. The evader is a manned jet fighter aircraft which means its maximum load factor is limited only to 9g.

The fitness values, *J* of Scenario 2 is given in Figure 6.17. It can be seen that EP is able to find feasible solutions as earliest as in the third generation, which was found by processor 4. Each processor independently searches for the feasible solution and feasible solutions at different number of generations.

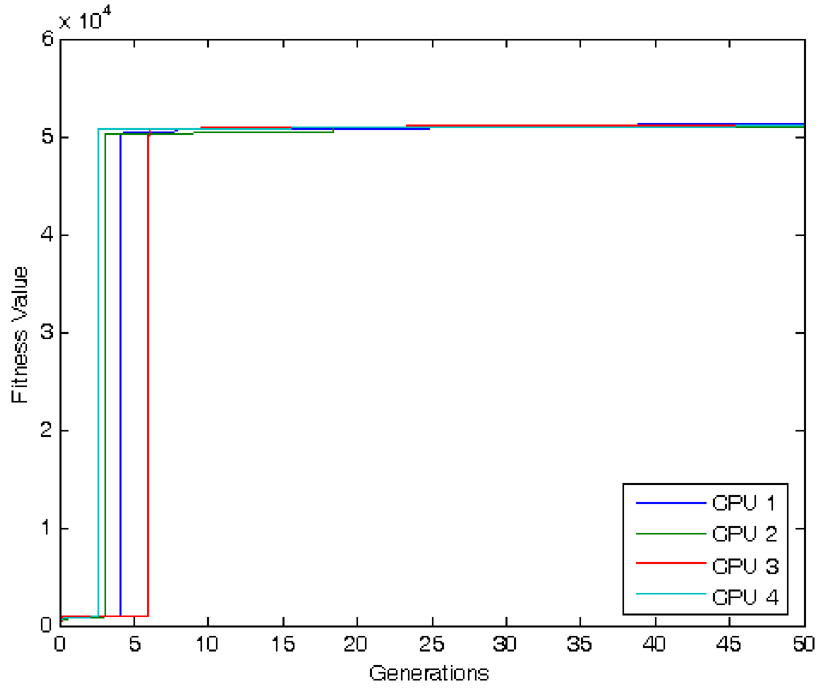


Figure 6.17: Fitness values for scenario 2.

However, EP finds the first feasible solution at a relatively short period of time, i.e. 4.7 seconds with 3.5 seconds standard deviation. In this type of problem where time is very important, the first feasible solution is would be adequate. The distribution of the time to find the first feasible solution is shown in Figure 6.18.

Out of 100 independent runs, three runs were selected and their 3D time-history trajectories are given in Figure 6.19 - Figure 6.21. Three trajectories from three independent runs are displayed. It can be seen that the feasible trajectories

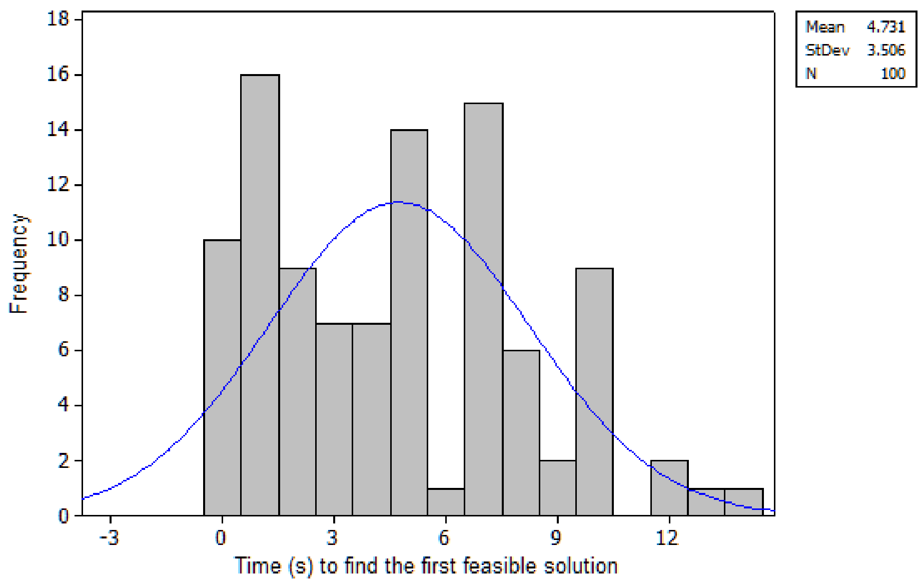


Figure 6.18: Computing time to find the first feasible solution.

are similar, i.e. flying away from the pursuer and performing rolling scissors to saturate the missile lateral acceleration that causes the miss.

The time-history of the load factors for both players and the separation distance is shown in Figure 6.22. The pursuer experiences an increasing load factor at the initial phase of the encounter due to a higher lateral acceleration to steer toward the evader. This is done with ease by the pursuer because at the 4th second, the load factor decreases which shows that the pursuer has positioned itself on a collision course with the evader. The chase continues until the 10th second. At that point in time, the pursuer is on the brink of interception, but the evader starts a high g manoeuvre which causes the load factor of the pursuer to increase and saturate its lateral acceleration. The manoeuvres performed by the evader causes the pursuer to overshoot and miss.

Figure 6.23 and Figure 6.24 show the trajectory of the feasible solution in XY- and XZ-plane, respectively. The feasible trajectory at the initial phase of the encounter is to increase the energy level, i.e. to fly in a straight line but descending. The evader chooses not to turn, because a high g turning will causes it to lose energy advantage and reduces the probability of successful avoidance. The high-g manoeuvres are reserved for the last phase of the encounter, i.e. before the predicted interception at the 11th second.

The players' velocity is shown in Figure 6.25. The pursuer's velocity increases

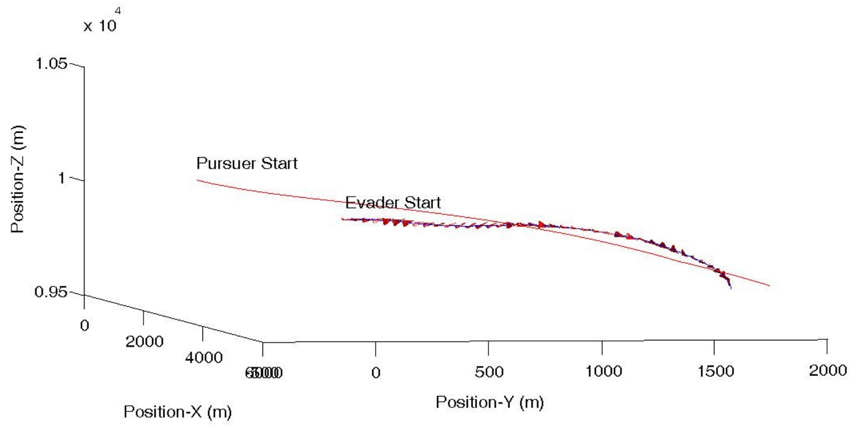


Figure 6.19: 3D view of the encounter in scenario 2 (run 1).

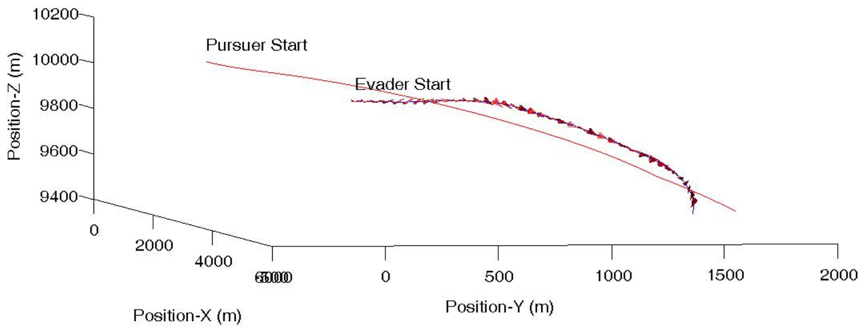


Figure 6.20: 3D view of the encounter in scenario 2 (run 2).

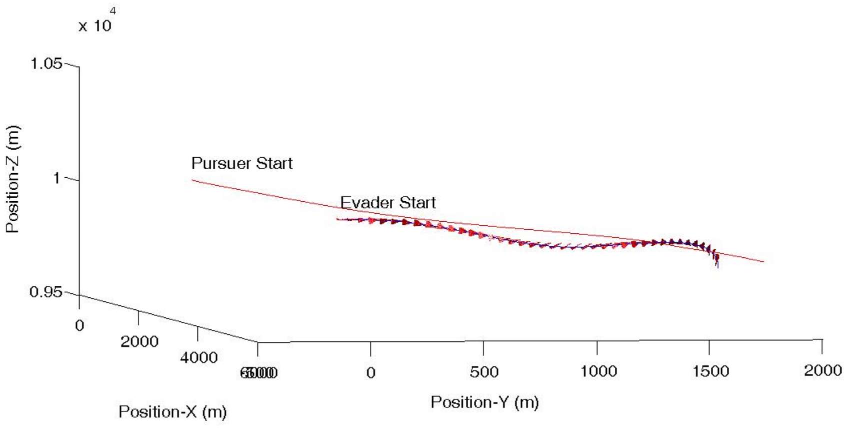


Figure 6.21: 3D view of the encounter in scenario 2 (run 3).

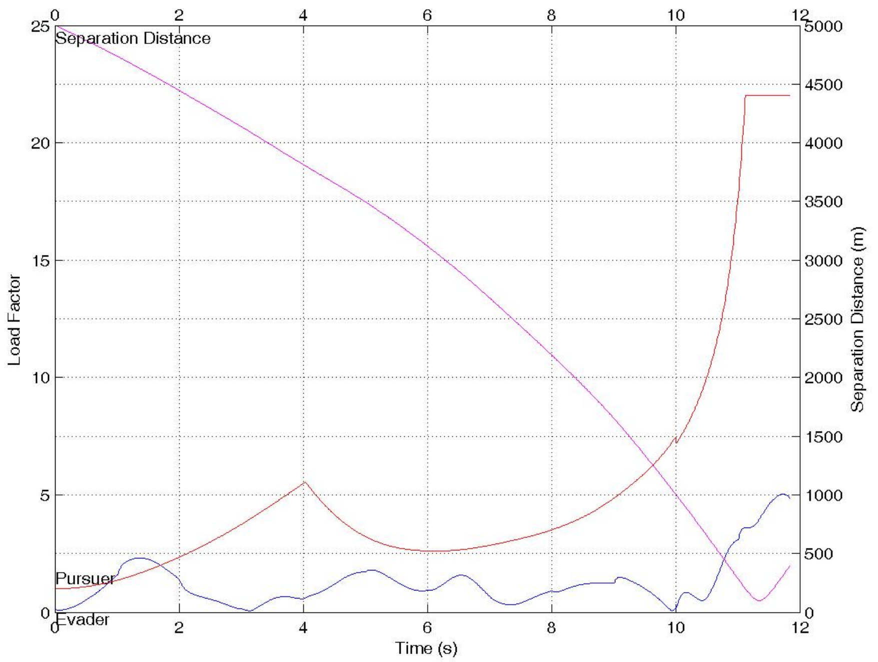


Figure 6.22: Load factors and the separation distance for scenario 2.

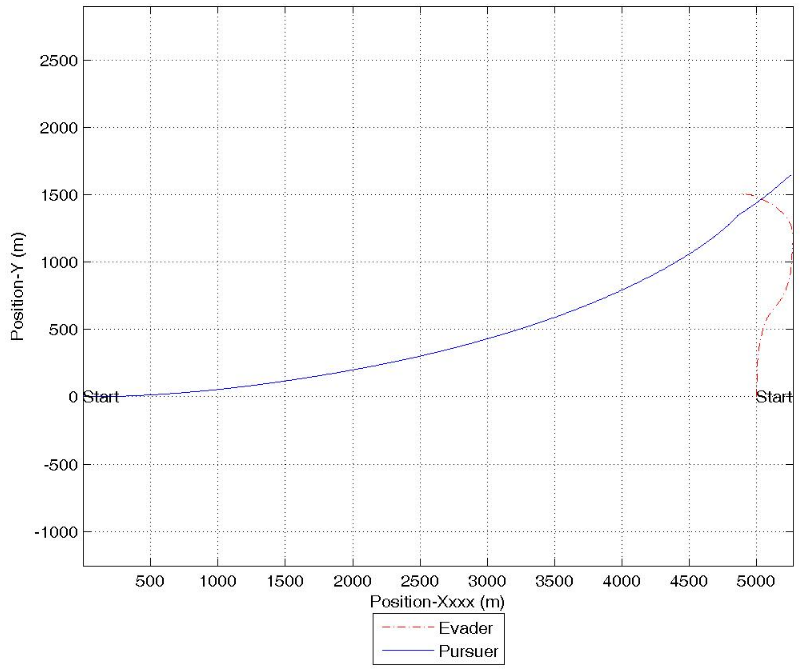


Figure 6.23: Encounter viewed in the XY-plane for scenario 2.

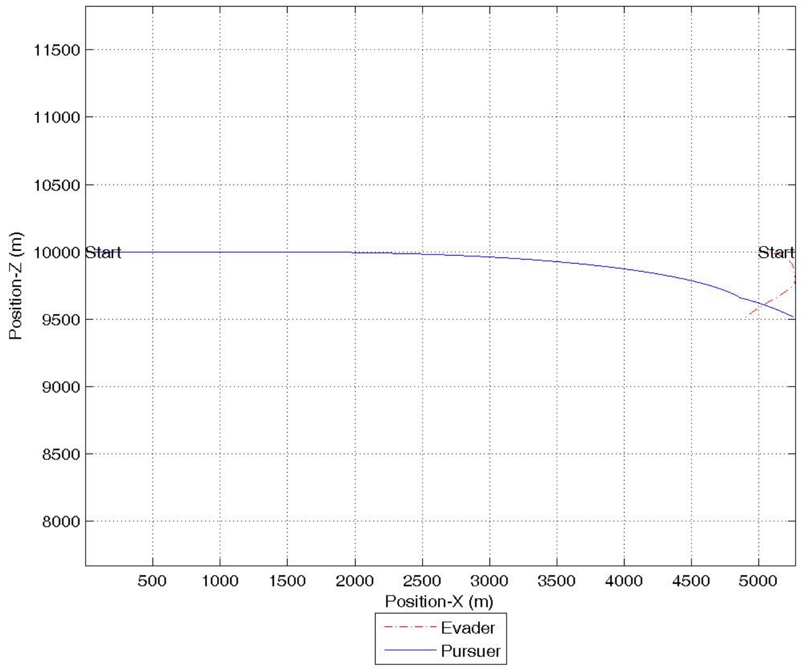


Figure 6.24: Encounter viewed in the XZ-plane for scenario 2.

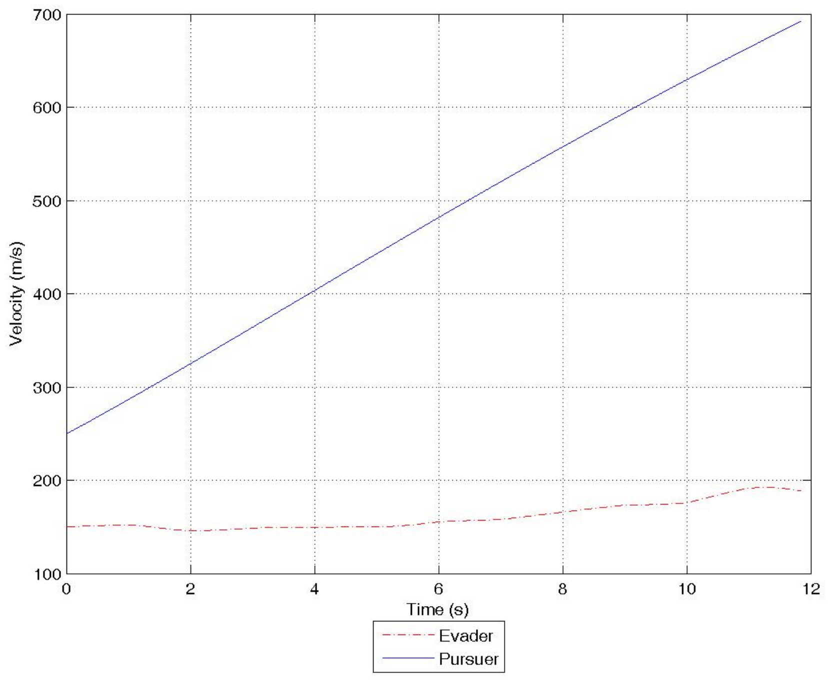


Figure 6.25: Velocity of both players for scenario 2.

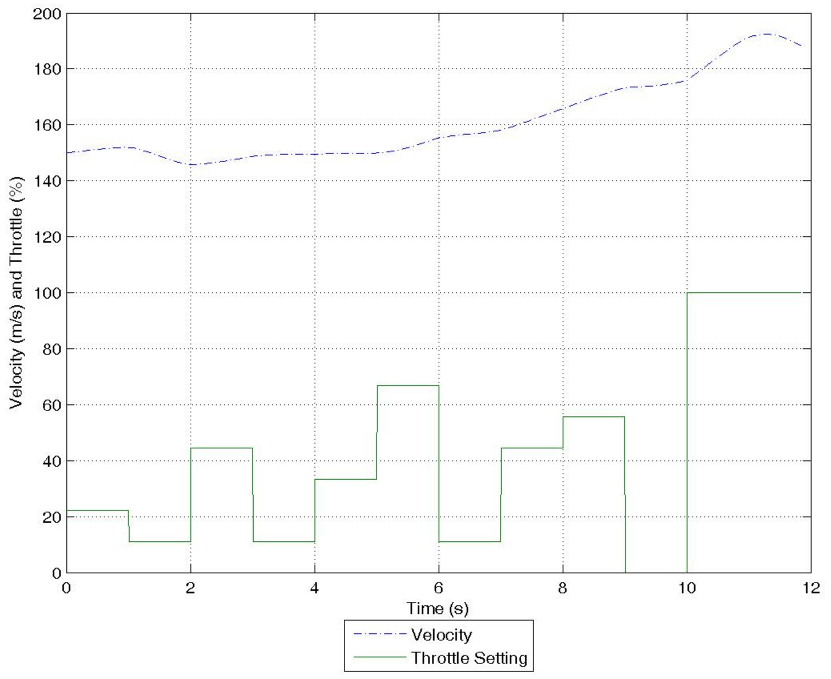


Figure 6.26: Evader’s velocity and throttle setting for scenario 2.

at a higher acceleration due to the high thrust its rocket motor produces. However, the evader's velocity does not increase much until the 8th second of the game. At the 9th second the evader set the throttle setting (Figure 6.26) to zero in order to maintain the velocity to achieve a higher turning performance. The aim is to steer the evader to a position that will saturate the lateral acceleration of the pursuer. At the 10th second, the evader engages maximum throttle to increase its velocity and further saturate the lateral acceleration of the pursuer. This causes the pursuer to miss the interception.

The behaviour of the control surfaces is shown in Figure 6.27, 6.29 and 6.30. The aileron deflection time-history shows bang-bang behaviour on the early part of the encounter. The evader deflects the aileron to its maximum positive deflection angle for one second, and then moves close to a neutral position. At the 4th second the evader applies a negative aileron deflection. However at the 9th second, the evader slowly decreases the aileron's deflection from the maximum positive until it reaches -6° at the 12th second. This action reverses the roll-rate from a negative roll-rate to a positive roll-rate as depicted in Figure 6.28. The manoeuvre is known as a split-S.

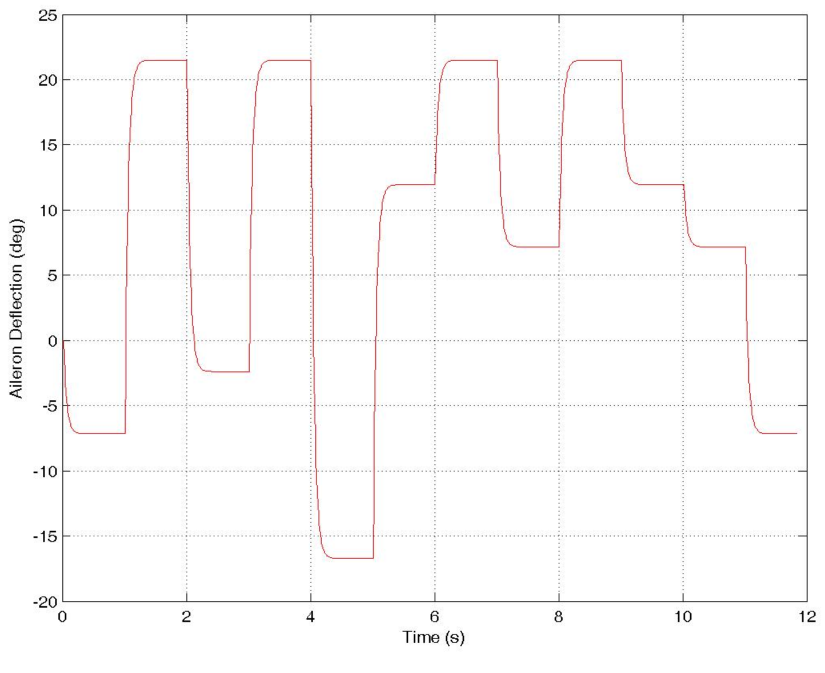


Figure 6.27: Aileron deflections of the evader for scenario 2.

The elevator deflection time-history is shown in Figure 6.29. In the figure, a

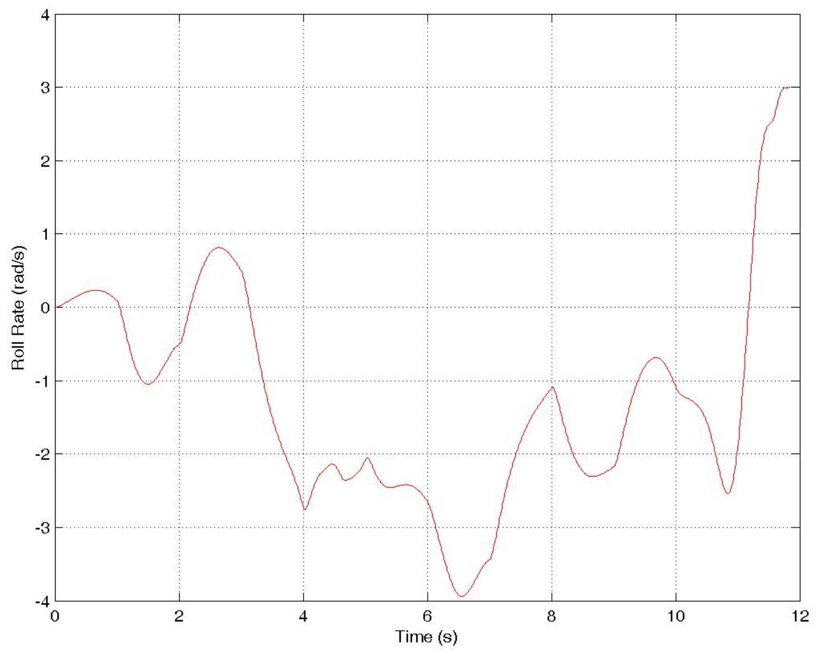


Figure 6.28: Roll-rate of the evader for scenario 2.

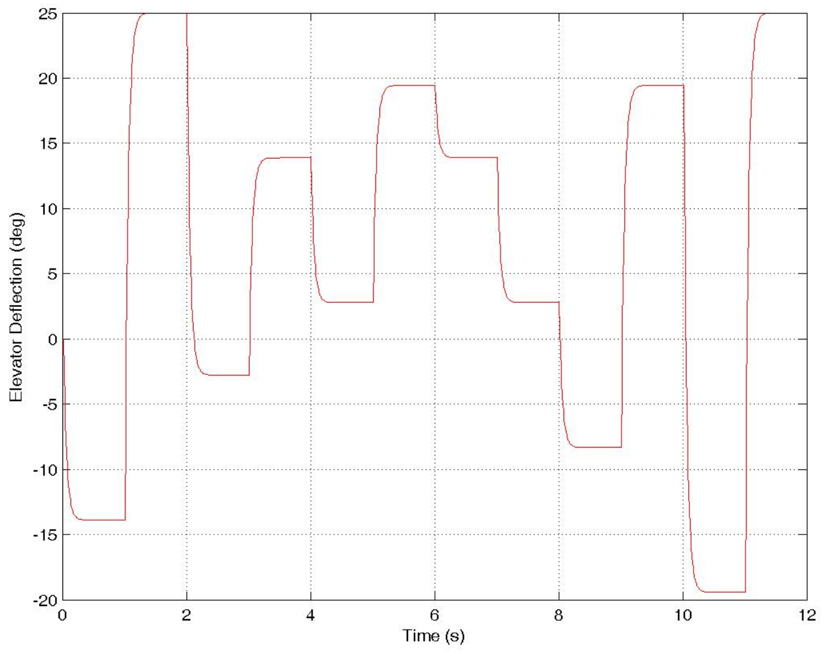


Figure 6.29: Elevator deflections of the evader for scenario 2.

large deflection angle of the elevator are observed in the initial and final phase of the encounter. Initially, the evader applies a large positive deflection angle of the elevator to point the nose of the evader down. The weaving manoeuvres are achieved by deflecting the rudder (Figure 6.30) and the aileron (Figure 6.27) in a correct and timely manner.

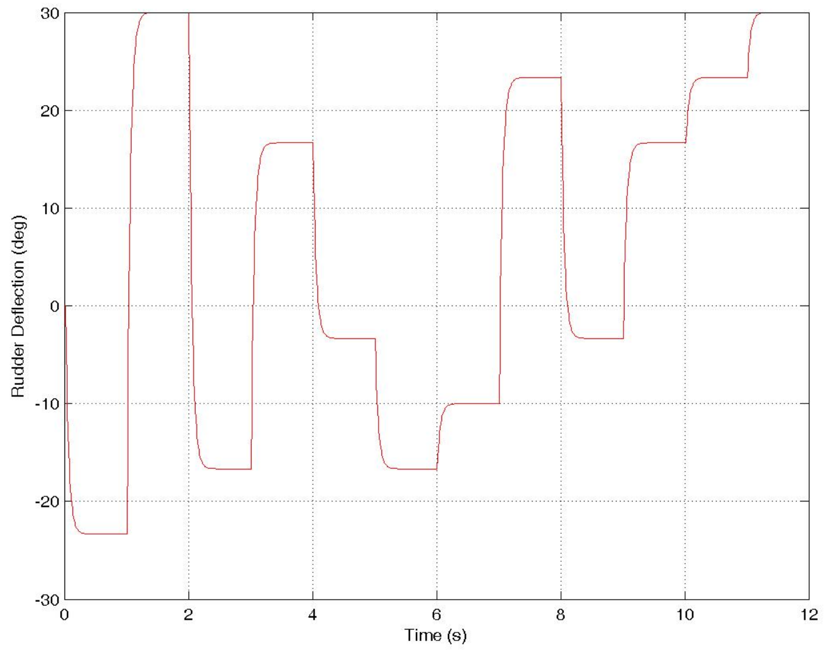


Figure 6.30: Rudder deflections of the evader for scenario 2.

6.5.3 Scenario 3: Head to Head Encounter

In this scenario, the pursuer and the evader are facing each other from a distance of 5000 m. The evader has to find feasible solutions in order to escape from interception. Similarly to the previous scenarios, 100 independent runs were conducted. The average execution time is 233 seconds with 30 seconds standard deviation. The evader is a manned jet fighter aircraft which means its maximum load factor is limited to 9g.

The fitness values, J is given in Figure 6.31. It can be seen that EP is able to find feasible solutions as earliest as in the third generation that is found by processor 1. Each processor independently searches for the feasible solutions and finds the first feasible solutions at different generations.

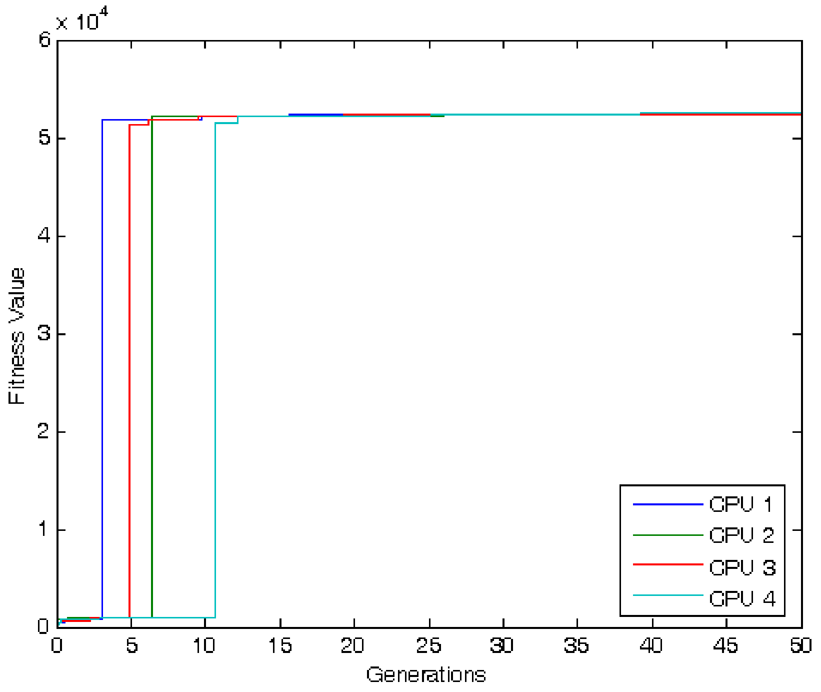


Figure 6.31: Fitness values for scenario 3.

The time to find the first feasible solution is shown as a histogram in Figure 6.32. EP finds the first feasible solution in a relatively short period of time, i.e. 4.4 seconds with standard deviation of 3.7 seconds.

The feasible trajectory for this scenario is shown in Figure 6.33 - Figure 6.35. These figures are plotted from three independent runs. Obviously, the only option to escape from interception is to dive and weave. On the other hand, such a turn and try to run away will only caused the evader to be captured.

As soon as the encounter begins, the evader immediately dives straight down. This increases the kinetic energy of the evader and hence the velocity. To increase the chances of a successful evasion, the evader also performs weaving manoeuvres.

Figure 6.36 shows the time-history of the load factor experienced by the players and the corresponding separation distance. At the initial phase of the encounter, the players perform a mild manoeuvre that involves a moderate load factor. As the separation distance becomes smaller, the evader starts to execute higher g manoeuvres. This saturates the pursuer's lateral acceleration. The timely manoevres performed by the evader causes the pursuer to overshoot the evader by 173 m.

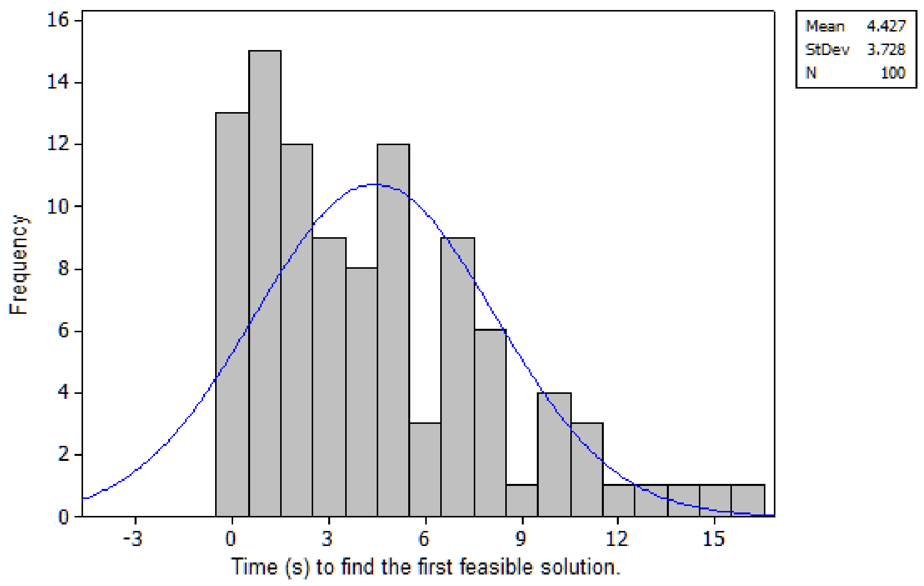


Figure 6.32: Computing time to find the first feasible solution for scenario 3.

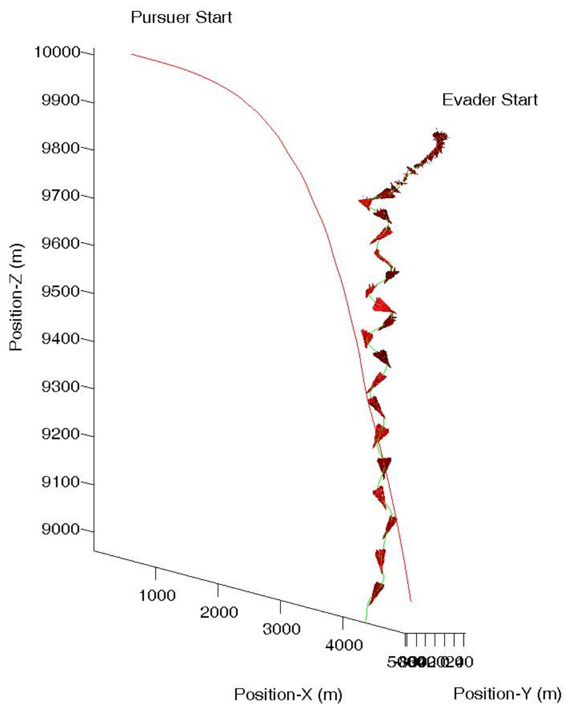


Figure 6.33: 3D view of the encounter in scenario 3 (run 1).

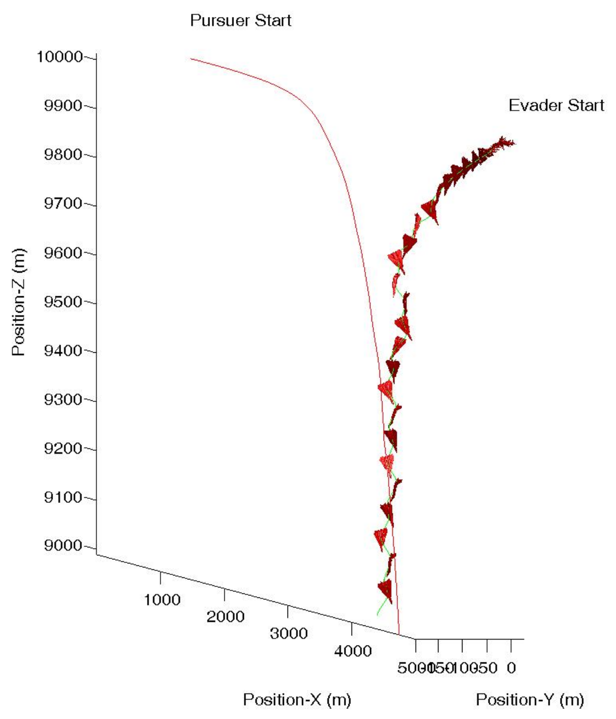


Figure 6.34: 3D view of the encounter in scenario 3 (run 2).

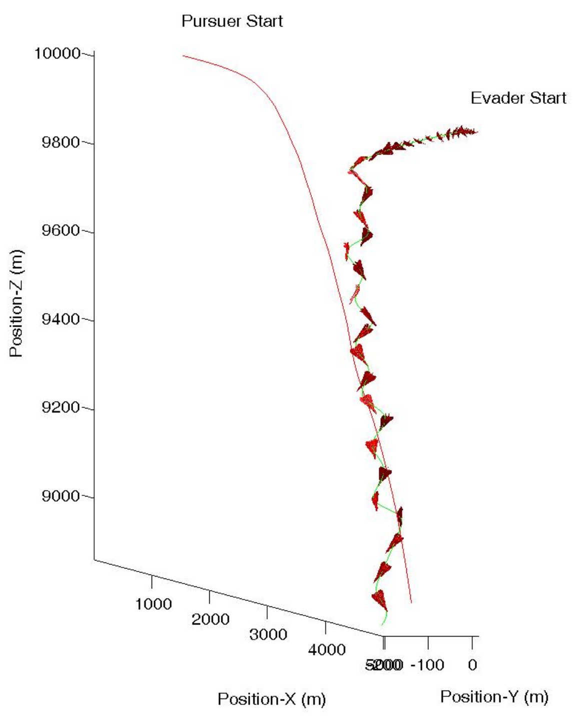


Figure 6.35: 3D view of the encounter in scenario 3 (run 3).

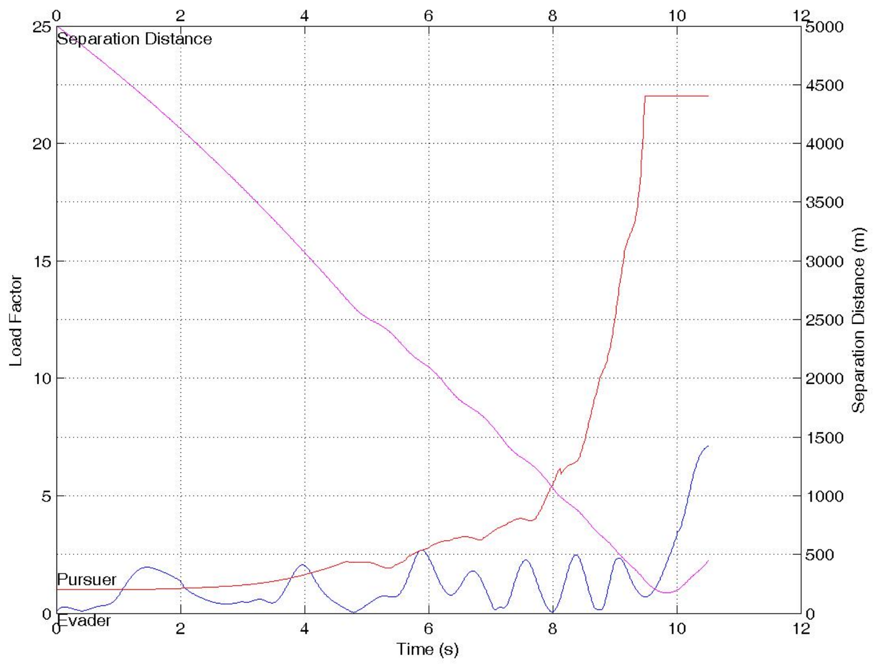


Figure 6.36: Load factors and the separation distance for scenario 3.

The feasible vertical dive performed by the evader is best visualised in the XY- and XZ-plane. The encounter in the XY-plane is shown in Figure 6.37. As seen in Figure 6.37, there is little movement in the X- and Y- axis. Most of the movements are predominantly in the Z-axis.

The trajectory is visible when viewed in the XZ-plane as shown in Figure 6.38. The evader's feasible solution is to dive orthogonally to the ground with a decrease in altitude by taking advantage of the positive gravitational acceleration. The evader maintains the 90^o dive until escapes from interception.

The velocity profile of the pursuer is similar to those in other scenarios. The powerful thrust produced by the rocket motor has provided a constant acceleration to the pursuer. Since the evader is diving straight to the ground, the evader's velocity is also increasing. At the end of the game, the pursuer's velocity is 654 m/s and the evader's velocity is 251 m/s.

The evader does not push the throttle to the maximum in this scenario. This is shown in Figure 6.40. In the early part of the encounter the evader maintains the throttle at 60%. Only at the 7th second, the evader pushes the throttle to 100% but that is only lasts for 1 second before the throttle is reduced to 55% and finally settled at 40%. The reason is not to increase the velocity immediately to

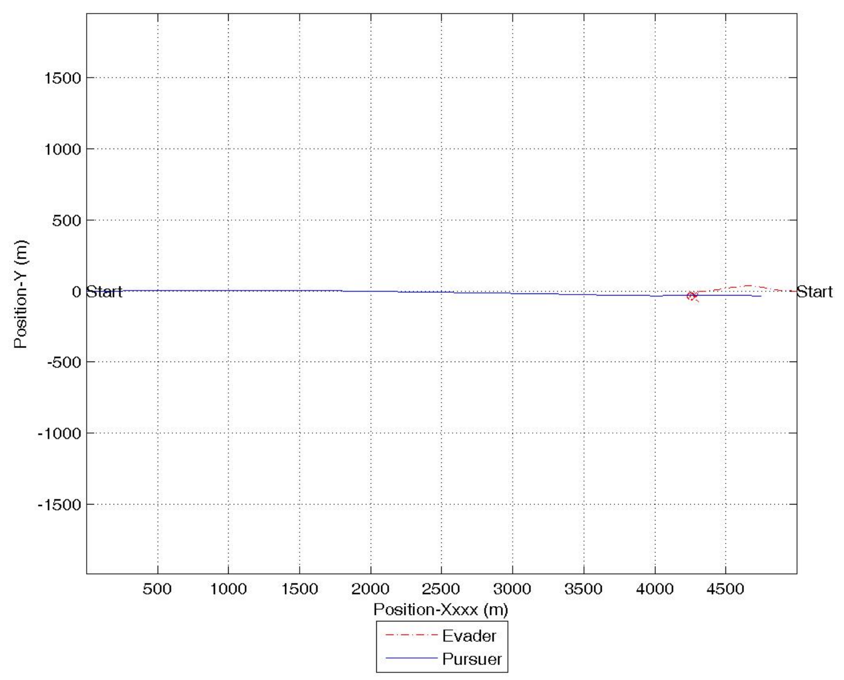


Figure 6.37: Encounter in the XY-plane for scenario 3.

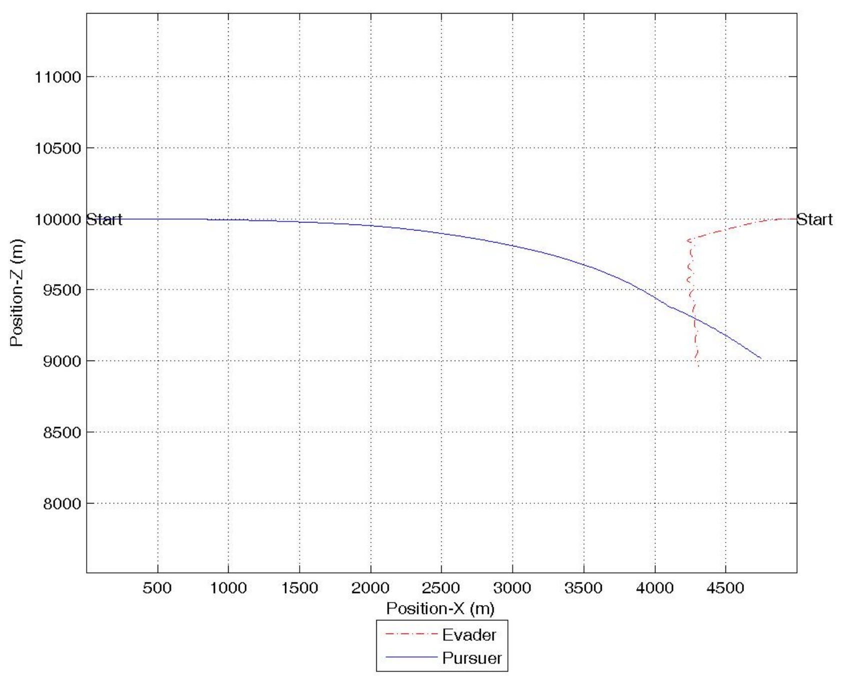


Figure 6.38: Encounter viewed in the XZ-plane for scenario 3.

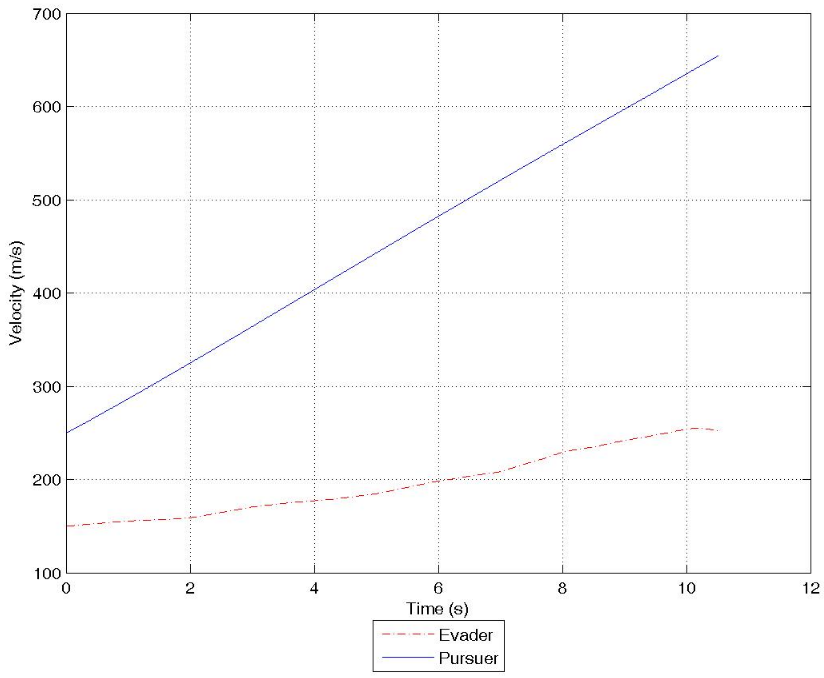


Figure 6.39: Velocity of both players for scenario 3.

the maximum as that will influence the turning performance of the evader, i.e. the minimum turning radius and the maximum turning rate at the end of the dive.

The control surface deflection time-histories of the evader are shown in Figure 6.41 - Figure 6.43. The weaving motion is accomplished by deflecting the control surfaces to a certain magnitude. Initially, in order to dive, the evader deflects the elevator to a maximum positive for 2 seconds. This causes the evader's nose to point downward and initiates diving. Then the evader's elevator is positioned to -30^o to reduce the pitch-rate to maintain the diving position. Over time, the control surfaces are deflected to a certain degree to produce a weaving motion. For example, the aileron is deflected to the maximum 21.5^o from time 0 to 1 second and the deflected to the -21.5^o for 1 second. The aileron is positioned in the negative deflection angle for 2 seconds before returning to almost neutral position at the 4th second. At the 5th second, the aileron is again deflected to -21.5^o and moved to a slightly lower angle until the 9th second of the encounter. It is important to note that these deflections are very well coupled so that the evader does not violate any of the constraints and in the same time provide a feasible trajectory for a successful evasion.

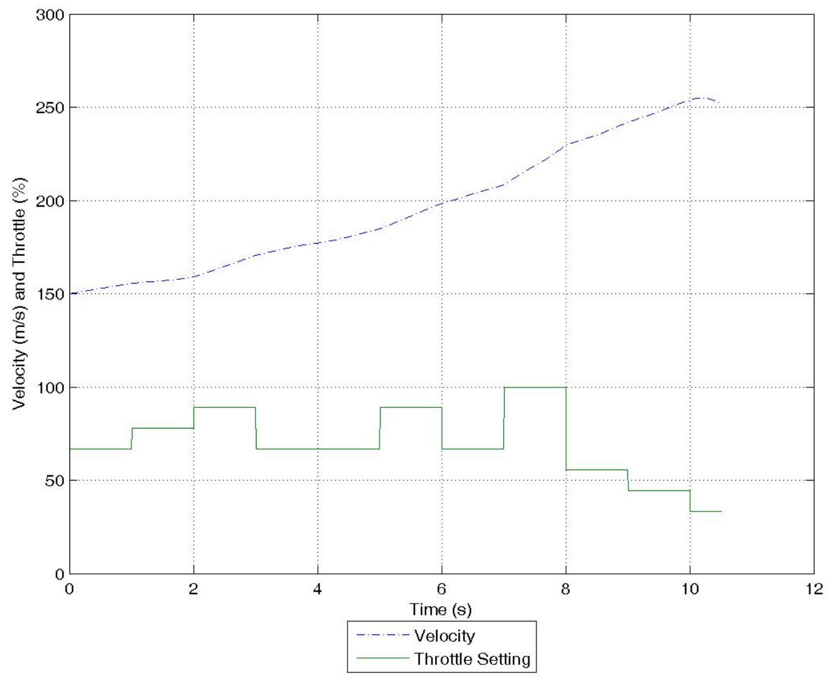


Figure 6.40: Throttle setting and velocity of the evader for scenario 3.

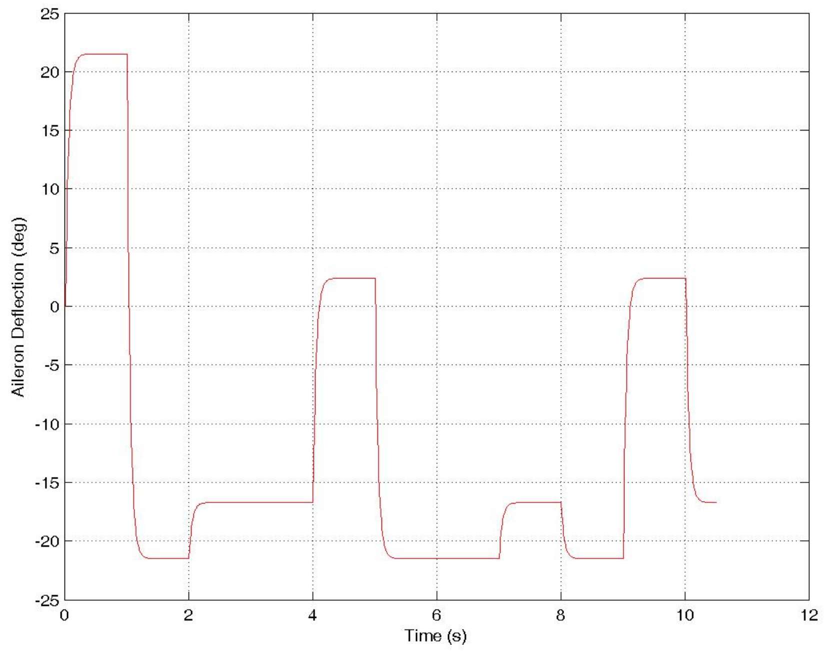


Figure 6.41: Aileron deflections of the evader for scenario 3.

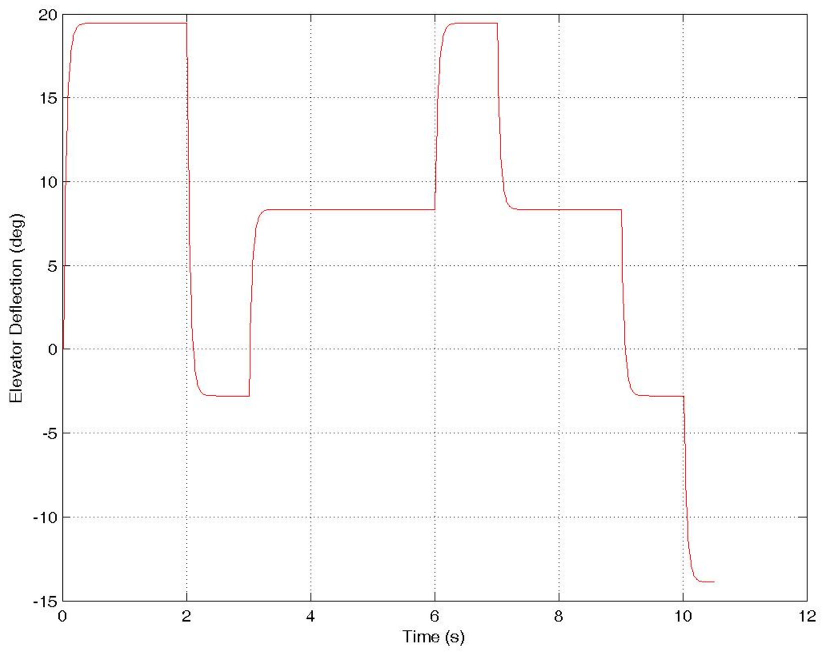


Figure 6.42: Elevator deflections of the evader for scenario 3.

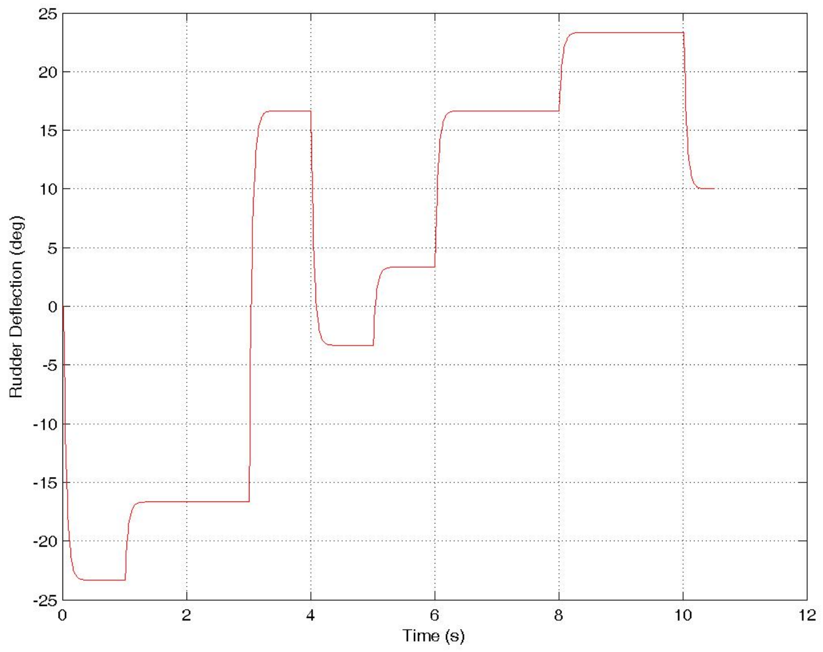


Figure 6.43: Rudder deflections of the evader for scenario 3.

6.5.4 Evader as Unmanned Air Combat Vehicle (UCAV)

In this scenario, the evader is assumed to be an Unmanned Air Combat Vehicle (UCAV) and the initial conditions are similar to scenario 1. The unmanned version allows the maximum load factor to be increased. At a higher *g* value, the evader has more agility because the constraint due to human limitations is alleviated. The evader is now free to perform higher than 9-g manoeuvres. Obviously, with this new advantage, it will be harder for the pursuer to intercept the evader.

A simulation was conducted to study the benefit of having a higher maximum load factor. The evader is the same as in the previous simulations except that the load factor limit is increased to 18. The time-history of the load factors of the players is displayed in Figure 6.44. Initially, the players' load factors are relatively low until the 15th second of the game. Initially, the pursuer misses the interception by a large margin before performing a turn for a second attempt.

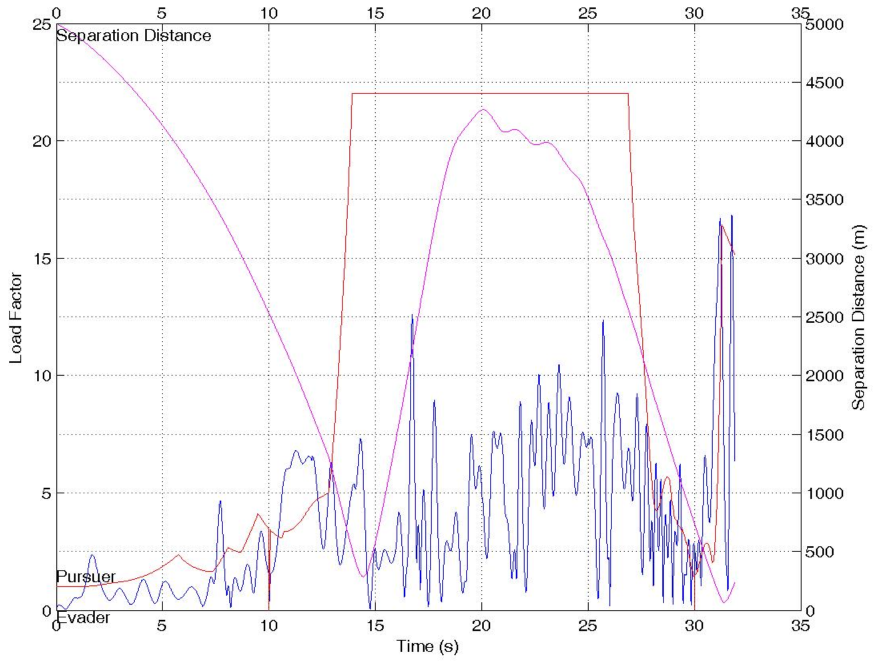


Figure 6.44: Load factors for both players and separation distance for UCAV (scenario 1).

Between 15th and 28th second of the encounter, the pursuer saturates its lateral acceleration when executing a high *g* turn to continue the intercept. The evader performs a weaving motion to increase its probability to avoid interception as show in Figure 6.45. The pursuer continues its attempt to intercept. Now, the pursuer has passed its rocket motor burnout stage. Its velocity is decreasing. This means the maximum turning rate becomes higher and the minimum turning radius becomes smaller. The evader has to manoeuvre at higher load factors to evade interception. This is shown in Figure 6.44. At the 30th second of the encounter, the separation distance is very small - i.e. the pursuer is close enough to intercept, the evader performs a 17-g manoeuvre to out manoeuvre the pursuer. Again, a feasible solution is a combination of weave and dive manoeuvre performed in a timely manner at high *g*.

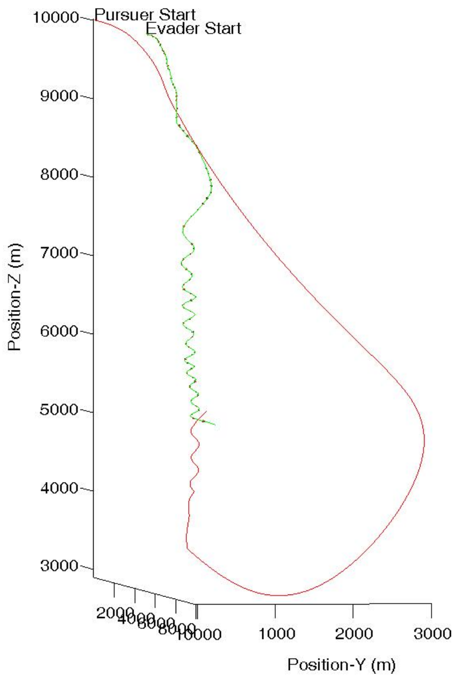


Figure 6.45: 3D view of the encounter with a UCAV for scenario 1.

As illustrated in Figure 6.44, at a higher maximum g-load, the evader could manoeuvre in such a way that increases the pursuer's miss distance. The evader is also has a better chance to avoid interception if the pursuer was able to turn around and continue the chase. This is because, after the pursuer has passed its burn-out stage, its velocity is decreasing due to the aerodynamic drag. The reduction of velocity increases the maximum turning rate and decreases the minimum turning radius of the pursuer. At one point, the pursuer's maximum turning rate matches with the evader's, even though the pursuer's velocity is still higher than the evader. At this moment, the pursuer can always keep up with the evader. The time to reach this situation can be delayed if the evader's maximum g-load is higher, as explained in Equation 5.26.

There only difference between manned and unmanned aircraft is the maximum load factor it could sustain while performing a manouvre. A higher maximum load factor means the aircraft could turn faster with a smaller turning radius. As a result, the aircraft has a higher chances to avoid interception.

6.5.5 Summary

To summarise the scenarios, it has been shown that for a different scenario, the feasible solution will also be different too. EP has shown that it can find a feasible solution for the given initial conditions. Besides that, it can also find the first feasible in a relatively short time. The feasible solutions found by EP are unique only to the given initial conditions and aircraft type. A different aircraft type and capability as well as a different missile type and capability will produce a different feasible solution. EP does the search without relying on past knowledge of feasible solutions.

6.6 Factors that Influence the Search

In this section, a simple design of experiment analysis was conducted to see the relationship between parameters. In parallel EP, it is acknowledged that the number of processors and the number of generations play an important role in locating feasible solutions.

As a rule of thumb, more processors, higher population size and larger number of generations will definitely improve the quality of the search space. But it comes with a price, i.e. a longer search time.

The experiment starts by considering two factors, i.e. the number of processors and the number of generations. The objective is to see the interaction between these factors. The initial parameters are given as in Table 6.3 and 6.4.

6.6.1 Interaction Effect Between Number of Processors, Number of Generation and Population Size.

The effect of interaction between these three factors, i.e. the number of processors, the number of generations and the size of the population is studied using a Design-of-Experiment (DOE) method. The statistical approach is employed due to the

Table 6.3: Initial Parameters.

Parameter	Pursuer	Evader
X,m	0.0	3000.0
Y,m	0.0	0.0
Z,m	3000.0	3000.0
Velocity,m/s	200.0	122.0
Heading Angle,rad	0	0
Interception Radius, m	15	-
Max Load Factor	20	10

Table 6.4: General Parameters.

Parameter	Value
Size of population	100
Migration Rate	10
No. of processors	4,8
Duration	100 seconds

Table 6.5: Experiment configurations and their results.

StdOrder	RunOrder	processors	No. of Gen.	Feasible solutions found
2	1	8	5	11
7	2	4	100	118
4	3	8	100	212
3	4	4	100	101
1	5	4	5	3
8	6	8	100	227
6	7	8	5	13
5	8	4	5	8

Table 6.6: Estimated Effects and Coefficients For Feasible Solutions.

Term	Effect	Coeff	SE Coef	T	P
Constant		86.23	2.913	29.74	0.000
CPU No.	58.25	29.12	2.193	10.00	0.001
No. of Gen	155.75	77.88	2.193	26.74	0.000
CPU*No.of Gen	51.75	25.87	2.193	8.88	0.001

Table 6.7: Analysis of Variance.

Source	Degrees	Sum	Mean	f_o	P-value
Source	of Freedom	of Square	Square	f_o	P-value
Main Effects	2	55302.2	27651.1	407.38	0.000
2-Way Interactions	1	5356.1	5356.1	78.91	0.001
Residual Error	4	271.5	67.		
Pure Error	4	271.5	67.9		
Total	7	60929.9			

stochastic nature of the optimisation algorithms. It involves the use of random numbers to generate initial populations to search for feasible solutions.

A DOE experiment is designed and conducted using the statistical package, MINITAB. The order and results of the experiment is given in Table 6.8 and 6.9.

Table 6.8: Experiment configurations of the DOE and their results.

No.	Standard Order	Run Order	CPU No.	Gen. No.	Population Size	No. of Feasible solution	Execution Time (s)
1	2	1	8	50	50	104	209.140
2	14	2	8	50	100	189	458.418
3	4	3	8	100	50	201	420.482
4	3	4	3	100	50	73	324.259
5	1	5	3	50	50	39	147.182
6	16	6	8	100	100	430	850.492
7	11	7	3	100	50	73	293.866
8	7	8	3	100	100	170	623.384
9	6	9	8	50	100	230	393.232
10	8	10	8	100	100	377	842.546
11	5	11	3	50	100	71	292.915
12	9	12	3	50	50	37	145.005
13	12	13	8	100	50	200	429.922
14	10	14	8	50	50	95	211.444
15	15	15	3	100	100	155	586.669
16	13	16	3	50	100	67	299.377

Table 6.9: Estimated Effects and Coefficients for the number of feasible solutions.

Term	Effect	Coefficient	SE Coef.	T	$P-$value
Constant		156.938	4.338	36.18	0.000
cpu	142.625	71.312	4.338	16.44	0.000
nogen	105.875	52.937	4.338	12.2	0.000
popsize	108.375	54.187	4.338	12.49	0.000
cpu*nogen	41.625	20.812	4.338	4.8	0.001
cpu*popsize	48.125	24.063	4.338	5.55	0.001
nogen*popsize	37.875	18.937	4.338	4.37	0.002
cpu*nogen*popsize	8.625	4.313	4.338	0.99	0.349

In Figure 6.46, using $\alpha = 0.05$, the effects of the number of processors, the number of generation , the population size and the interactions among themselves are found to be significant. These three parameters play an important part during the optimisation cycle. The search is improved if more processors, a higher

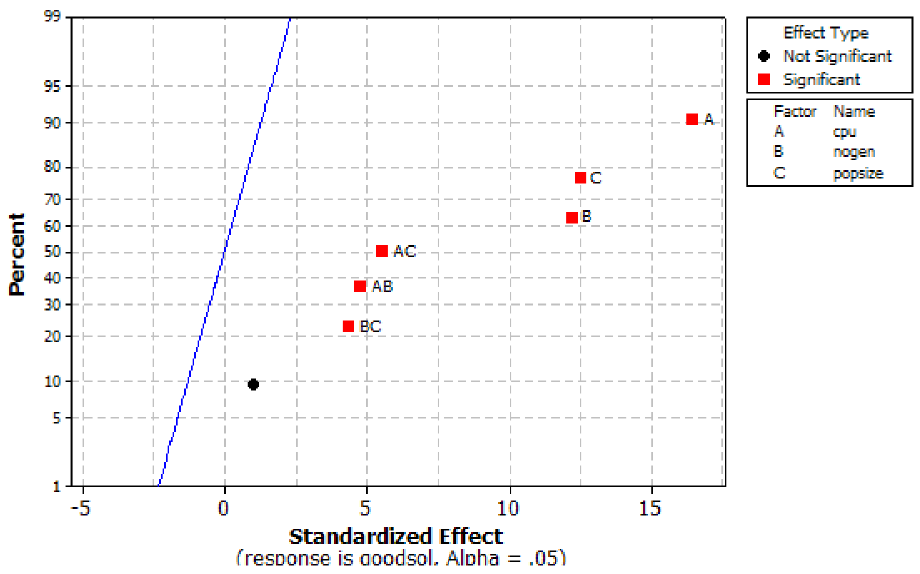


Figure 6.46: Normal plot of the standardized effects of the number of feasible solutions.

number of generations and a larger population size are used. The effect of these parameters on the number of feasible solutions and the execution time are studied.

The "main effects" plot for the number of feasible solutions found is given in Figure 6.47. As seen in the figure, all of the three factors give almost similar improvements to the number of feasible solutions. A slightly bigger effect is given by the number of processors (*cpuno*). It is represented by a steeper slope as compared to other effects.

The main effects plot for the execution time is given in Figure 6.48. It has been observed that there is a significant effect to the execution time for the number of generation (*nogen*) and population size (*popsize*). The number of processors does not as significant as the other two. The reason is because *nogen* and *popsize* contribute directly to the execution time, i.e. more solutions have to be evaluated. The effect of *cpuno* to the execution time is primarily due to the communication time and the serial part of the algorithm which is usually performed by the master processor.

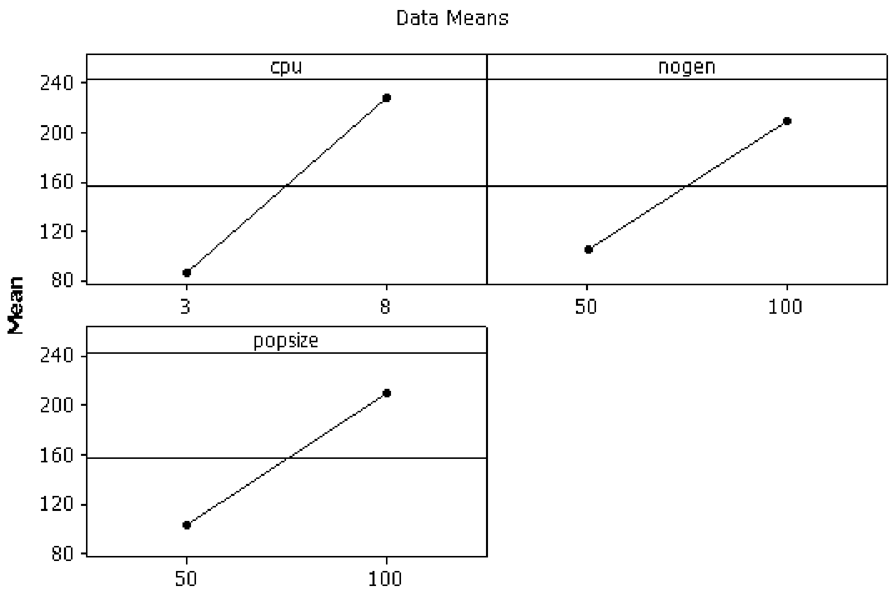


Figure 6.47: Main effect plot for feasible solutions.

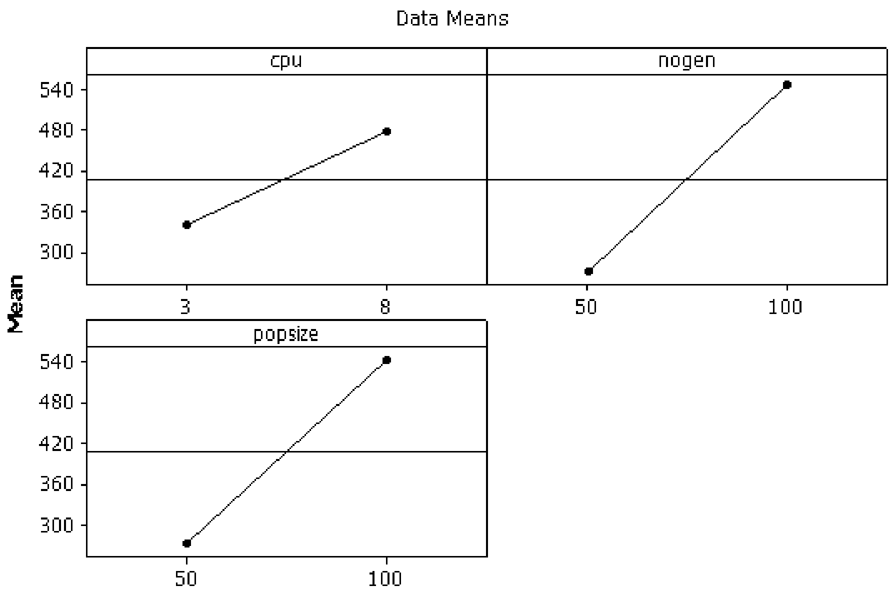


Figure 6.48: The main effect plot for execution time.

6.6.2 Effect of the Spawn Time

The effect of the spawn (simulation) time, as described in Figure 5.1, is studied. The optimisation performs better if the spawn time is short, e.g. 10 seconds. This opens the opportunity for the optimisation to fix itself from drifts due to the presence of errors and noise. The main contributors that cause drifts are measurement errors from the sensors. Table 6.12 lists the percentage of successful runs at different spawn times. The general and initial parameters of the simulation are given in Table 6.10 and 6.11.

Table 6.10: Initial Parameters.

Parameter	Pursuer	Evader
X,m	0.0	3000.0
Y,m	0.0	0.0
Z,m	3000.0	3000.0
Velocity,m/s	200.0	243.0
Heading Angle,rad	0	0
Interception Radius, m	15	-
Max Load Factor	40	10

Table 6.11: General Parameters.

Parameter	Value
No. of genertion	100
Size of population	100
Migration Rate	10
No. of processors	4

Table 6.12: Percentage of Finding of Feasible Solution after n number of runs.

Simulation Time (s)	Percentage of Successful Runs (%)	Total Number Runs
10	98	1009
20	90	1145
40	50.8	638
60	20.6	1503
80	6.4	1415
100	0	0

Table 6.12 gives the percentage of success runs for different simulation times. The simulation time, t_s is a predefined time in which the simulation will take place.

Evolutionary Programming will evaluate an individual for a given t_s. As predicted, when t_s is small, the chances of an individual to survive (i.e. becoming a feasible solution) in EP is higher. The best solution is then played in the simulation for t_s seconds. The optimisation algorithm will search for feasible solution for nt_s until no feasible solution is found.

The optimisation is very sensitive to constraint violations such that even if the violation happened at the last second of the simulation, the whole individual is considered infeasible, as explained in Figure 6.49.

Figure 6.49 shows the results of several spawn times. In each spawn time, 1000 independent optimisation runs were conducted. The result of each run is recorded. Some of the runs were able to play the game up to more 100 seconds because EP was able to locate feasible solutions within the time frame. But some of the runs were cut short. No feasible solutions were found even after 30 seconds. The results are plotted in a histogram. shows the results of several spawn times. In each spawn time, 1000 independent optimisation runs were conducted. The result of each run is taken. Some of the runs were able to play the game up to more 100 seconds because EP was able to locate feasible solutions until that time frame. But some of the runs were cut short. No feasible solutions were found even after 30 seconds. The results are plotted in a histogram in Figure 6.49.

For a 10 second simulation time (see Figure 6.49(a)), the feasible solution behaves similarly like a nonlinear Model Predictive Control [66], which requires a short spawn time usually in seconds. In Figure 6.49, the 10 second spawn time has enabled EP to search for feasible solutions for more than 180 seconds of encounter. This means feasible solutions are easier to locate at smaller spawn time.

In Figure 6.49(b), the 20 second spawn time shows that EP is still capable in locating feasible solutions and able to extend the duration of the encounter up to more than 200 seconds without interception.

However, a shorter simulation time could influence the overall outcome of the game. For example, for the 10 second simulation time, although a feasible solution is found, it may lead the evader to fly close to the ground and at the end leaves a very little space to manoeuvre.

If the simulation spawn time is increased to 20 seconds, a feasible solution could be found and at the same time provides more opportunities to other feasible trajectories in the next spawn time.

A longer spawn time has its own disadvantage. The chances of finding a feasible solution is lower due to higher probability for the evader to violate a constraint.

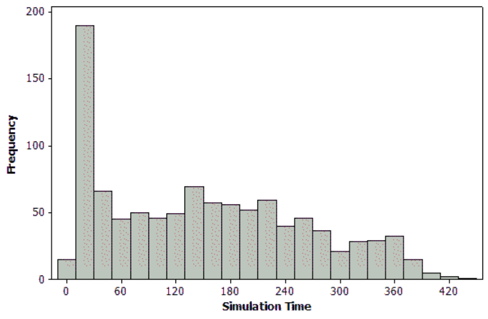
This requires a larger search space and subsequently, consuming more time and computing power. The effect is shown in Figure 6.49(c) - (f). For example, at 100 second spawn time, none of the runs conducted is able to extend the game to more than 100 seconds. This shows that the search for feasible solutions with longer spawn time is difficult because the probability of a solution with a long spawn time that violates the given constraints is higher. The selection in EP is set to be very conservative. It rejects a solution that even violates one constraint. As a summary, a moderate spawn time should be used to locate feasible trajectories. It must not be too short to avoid ”shortsightedness” in locating feasible solutions, but it should also not be too long to avoid longer computing time and failure to even find one feasible solution.

6.6.3 Effect of the Pursuer's Thrust

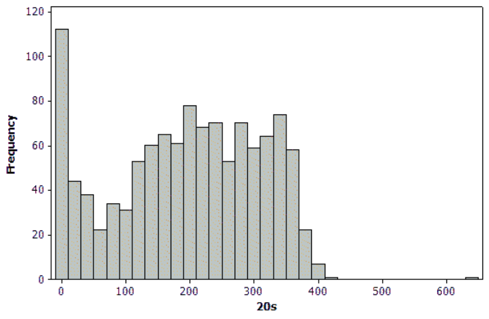
The effect of the thrust of the pursuer is studied. If the evader has more thrust to weight ratio than the pursuer, it could simply choose to fly away from the pursuer. On the other hand, if the pursuer's thrust to weight ratio is higher, the evader may choose the perform hard manoeuvres to escape interception.

In this experiment, the pursuer's thrust is arbitrarily set to be 1080 N, 2080 N, 4080 N and 5880 N. The results are plotted in Figure 6.50. Figure 6.50(a)-6.50(b) and shows the feasible trajectory taken by the evader is to fly away from the pursuer. This is true since the evader's thrust produces higher velocity and outruns the pursuer is the best option.

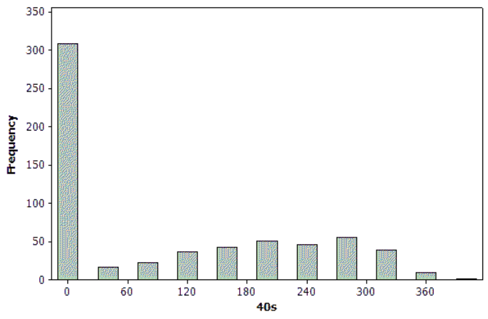
As the pursuer's thrust is higher, as shown in Figure 6.50(c)-6.50(d), the feasible path taken by the evader is to perform timely and optimal manoeuvres as described in Section 2.5. In the figures, the pursuer's velocity closely matches the evader's velocity at the final moment before the game ends, as shown by the staggered trajectory.



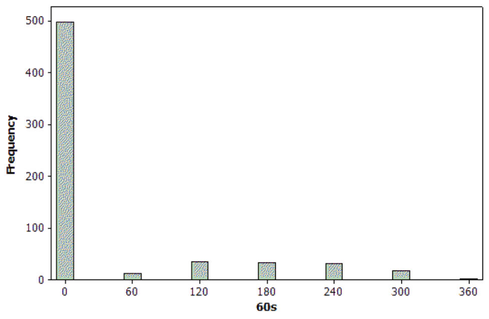
(a) Simulation of 10 seconds.



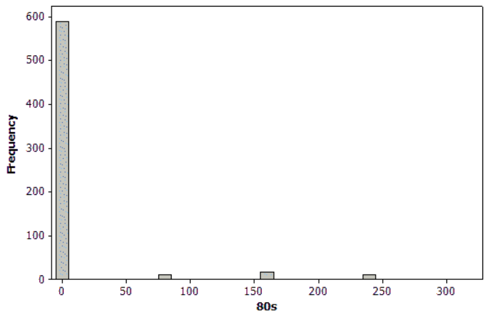
(b) Simulation of 20 seconds.



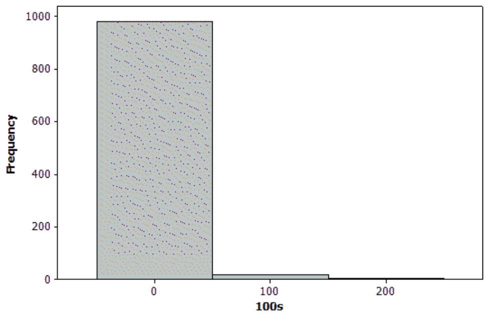
(c) Simulation of 40 seconds.



(d) Simulation of 60 seconds.

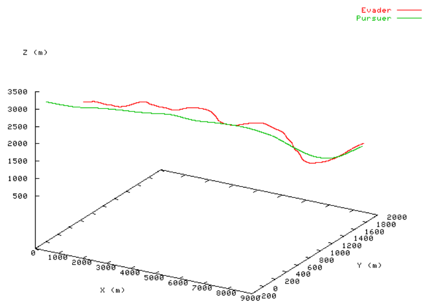


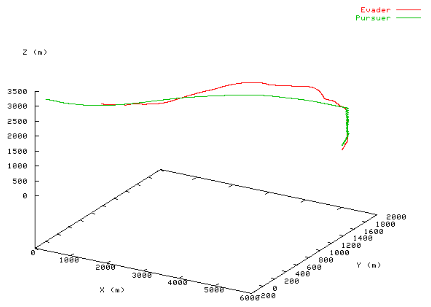
(e) Simulation of 80 seconds.

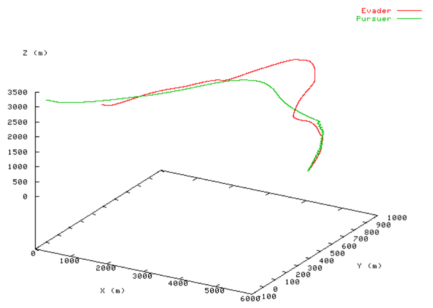


(f) Simulation of 100 seconds.

Figure 6.49: Histogram of runs for different simulation (spawn) time.


(a) Pursuer's Thrust 1080 N


(b) Pursuer's Thrust 2080 N


(c) Pursuer's Thrust 4080 N

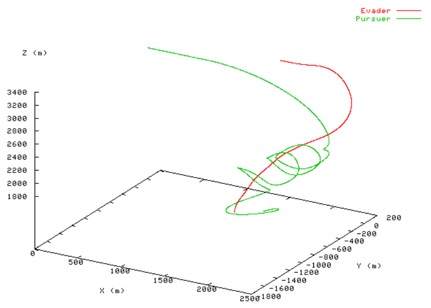

(d) Pursuer's Thrust 5880 N

Figure 6.50: Three-dimensional encounters for different thrust.

6.7 Robustness

The robustness of the the algorithm is studied by using Monte Carlo simulations. In this experiment, both players are assumed to know each other's states. No noise is incorporated in the simulation model. The only random parameter is the initial solution in EP. In each run, EP initialise the first population randomly. The outcome of each run is saved and statistically analysed.

The first attempt to find the consistency of the optimisation algorithm is to run the same configuration for 300 times. In this experiment the configuration of the players are as given in Table 6.13 and 6.14.

Table 6.13: Initial Parameters.

Parameter	Pursuer	Evader
X,m	0.0	3000.0
Y,m	0.0	0.0
Z,m	3000.0	3000.0
Velocity,m/s	200.0	243.0
Heading Angle,rad	0	0
Max Load Factor	40	10

Table 6.14: General Parameters.

Parameter	Value
No. of generation	100
Size of population	100
Migration Rate	10
No. of processors	4
Duration	100 seconds

Further insight into the results given Figure 6.51 and 6.52 in has revealed some statistical results as given in Table 6.15.

Table 6.15: Statistical Results.

Parameter	Value
Mean	205.606
Standard Deviation	14.565
95% CI	204.240,206.972
P	0.000 (¡0.005)

The experiment is repeated by increasing the pursuer's interception radius to 20 m. The result is given below in Figure 6.53 and 6.54. In this experiment, the

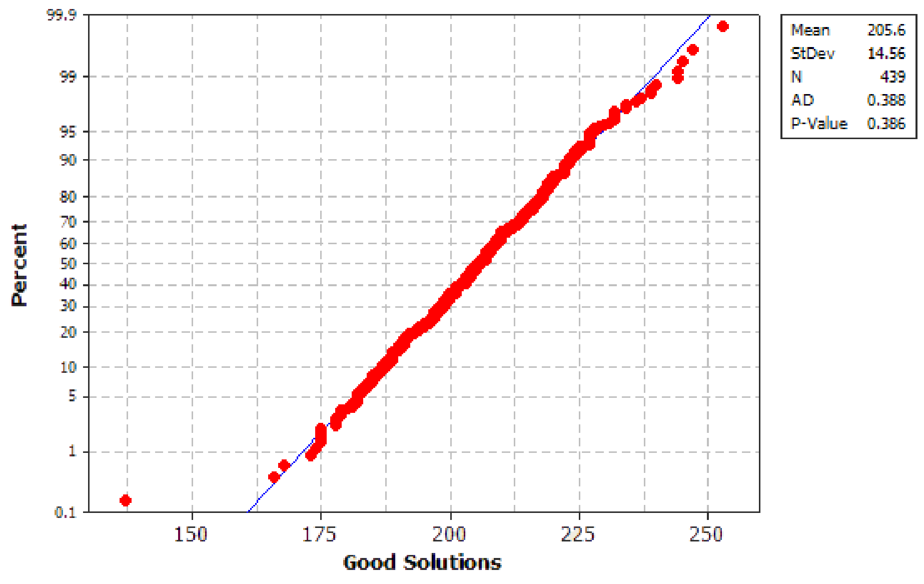


Figure 6.51: Normal plot of the number of feasible solutions using Anderson-Darling Method.

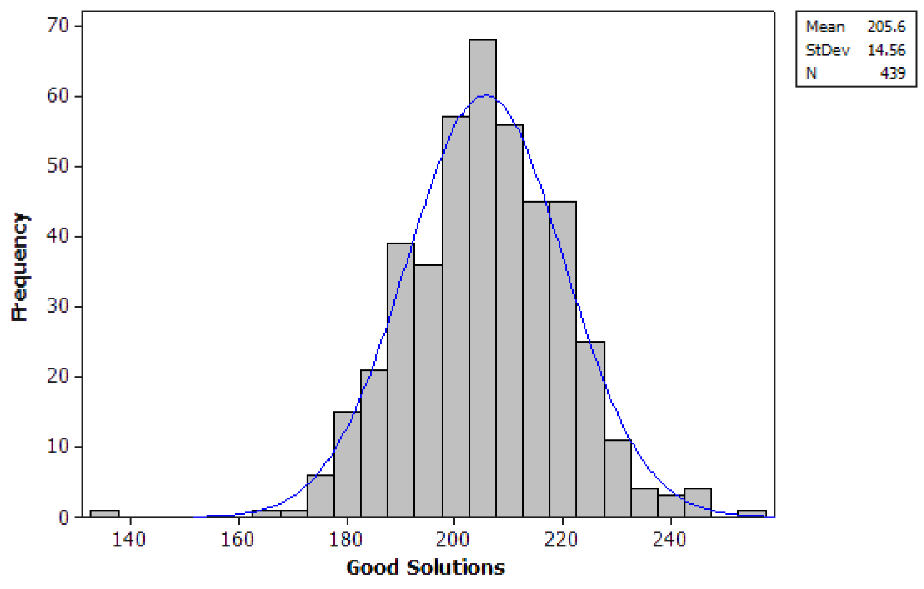


Figure 6.52: Histogram plot of the number of feasible solutions.

same configuration is ran 300 times. The assumption that the optimisation algorithm is able to find feasible solutions is proven by using the *t*-test. The normality test shows that the result is within normal distribution which shown in Figure 6.53. The histogram of the results is given in Figure 6.54. The Anderson-Darling test shows the result of 0.338 (Figure 6.53) is less than 0.752 at 5% level test which means the hypothesis of normality is accepted.

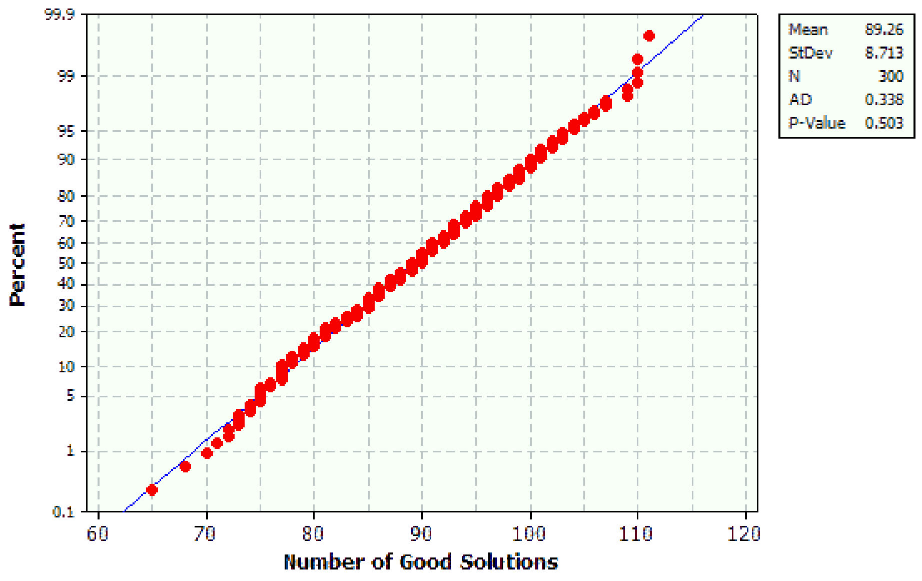


Figure 6.53: Normal plot of feasible solutions using Anderson-Darling Method for 20 m interception radius.

The next test is to verify the hypothesis that at a level of 95% confidence interval and using the test value of 90 for the number of feasible solutions, the results agree the null hypothesis is accepted. The statistical results are given in Table 6.16. It can be deduced that in 95% of the time, the number of feasible solutions found is between 88 and 90 which is more than enough for the evader to avoid interception.

Table 6.16: Statistical Results.

Parameter	Value
Mean	89.26
Standard Deviation	8.713
95% CI	88.27,90.25

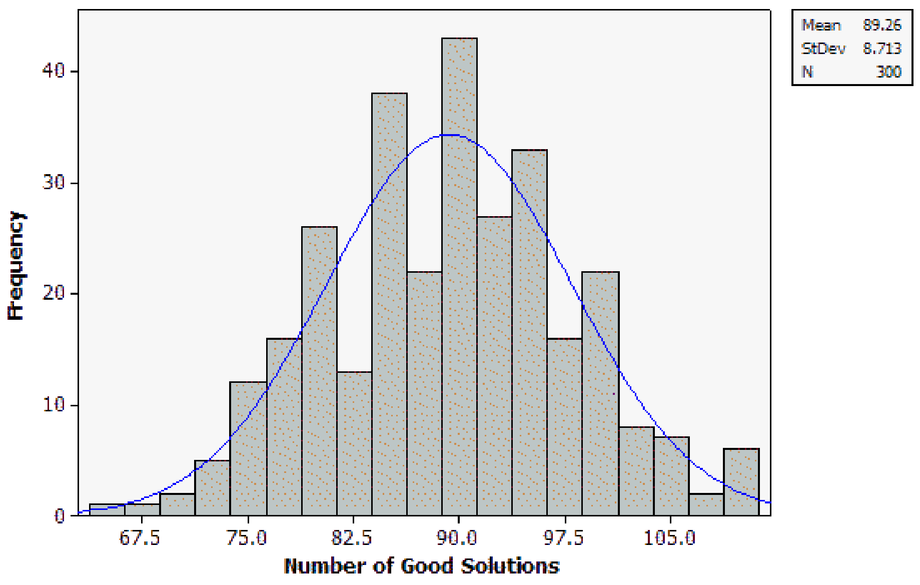


Figure 6.54: Number of feasible solutions for 20 m interception radius.

6.8 Presence of Uncertainties

Finding a feasible solution in the presence of errors and noise is a challenge. This is due to the imperfect sensors and other types of measurements needed to solve the problem. Also, the noise may also come from exogenous sources such as turbulence. A feasible solution may no longer be feasible if the presence of errors and noise will alter the states and eventually change the predicted outcome. It is imperative to ensure the algorithm is robust against factors such as noise.

There is a wide range of uncertainties involved, and only a handful is covered in this thesis. However, as a note, uncertainties involved in EP can be categorised into four classes [62], (i) the fitness function is not certain or noisy. In this particular problem, the problem is mainly from sensory noise and measurement errors, (ii) the design variables or environmental variables (which is beyond of our control) fluctuates over certain values, (iii), the fitness function is time-variant, and finally, (iv) the fitness function is just an approximation of the model to be studied which definitely has approximation errors in it.

In this experiment, the evader searches for feasible solutions by assuming there is no error and noise. The feasible solutions found is tested again for a noisy environment. The outcomes of the objective function are compared with the noise-free result.

Several degrees of noise and errors are tested in this experiment. Single noise experiments are also conducted and discussed further in the following section.

6.8.1 Effect on Errors in the Evader's Initial Speed

The effect of noise and errors to the feasible solutions are studied. In contrast to the perfect simulation, i.e. no noise and errors, the feasible solution may not perform as expected if it is run again in a noisy environment. The arguments are based on the factors that the accumulation of errors and noise will eventually modify the states of the evader at every time step. At the end of the simulation, the final position of the evader in the noisy environment differs greatly from the one used in a noise-free simulation. To demonstrate this, an experiment was conducted.

In this experiment, several optimisation simulations were conducted to find a group of feasible solutions. These feasible solutions are then run again but this time; the initial speed of the evader was varied slightly by using normal distribution of zero mean and variance of one as given in Eq. 6.1.

$$V^* = V_i + \alpha N(0, 1) \tag{6.1}$$

where V^* is the perturbed variable, V_i is the original variable, α is a scale factor and $N(0, 1)$ is the standard normal distribution with zero mean and variance of one.

This experiment follows the parameters as given in Table 6.17 and 6.18. A total of 25 feasible solutions were tested. Each solution is repeated 500 times. The results are given in Table 6.19.

Table 6.17: Initial Parameters.

Parameter	Pursuer	Evader
X,m	0.0	3048.0
Y,m	0.0	0.0
Z,m	3000.0	3000.0
Velocity,m/s	200.0	243.0
Interception Radius, m	15	-
Max Load Factor (g)	40	10

In Table 6.19, the effect of a small change in the initial state is significant. At each run, the same solution is simulated 500 times and out of that, a very low percentage of the outcomes is considered feasible. In all cases, the feasible solutions were unable to perform as intended if the initial speed is perturbed.

Table 6.18: General Parameters.

Parameter	Value
No. of Generations	100
Size of population	100
Migration Rate	10
No. of processors	4
Duration	100 seconds

Thus, we can say that the solution is sensitive to the evader's initial speed. The slightest change in the initial speed will change both acceleration, as well as the distance. Furthermore, the timing of the deflection of the control surfaces is important. If the timing is incorrect, due to the noise, the trajectory that was once feasible will no longer be feasible. The error is accumulated at every time step. It has been seen that a small perturbation to the evader's initial speed strongly influences the outcome of the game as seen in Table 6.19. Any small perturbation on the evader's initial speed will significantly affect the outcome of the simulation. It can be said that the simulation model is sensitive to the initial speed of the evader.

Even though the initial error is small due to the complexity of the mathematical models and the build up of errors over a period of time, the aircraft will still fly but at a different trajectory. This would be true even if the control deflection sequence is the same (since the sequences are decoded from the feasible solution found by EP). This will cause the final position of the evader to vary significantly. It can be seen that initially, there is no significant path change made by the evader. The evader's path starts to deviate after flying for about 1000 m.

A second experiment is conducted in a slightly different manner. For each candidate solution, EP will run the evaluation twice. The first evaluation is without noise and the second evaluation is with noise. The same initial parameter as in Table 6.17 and 6.18 is used in this experiment.

A total of 155 independent EP simulations were conducted. On average, only 50% of the simulations managed to locate feasible solutions. However, feasible solutions from 64% of the simulations have become non feasible solutions when noise was added. Only 36% of the feasible solutions remained as feasible solution after noise was added. This shows that EP is sensitive in the presence of noise in relation to the speed of the evader.

Table 6.19: The number of successful (not captured) simulations when a feasible solution is rerun 500 times with errors in the evader's initial speed.

No	Actual Number of Feasible Solutions	per 500	Percent (%)
1	44	0.088	8.8
2	65	0.13	13
3	31	0.062	6.2
4	23	0.046	4.6
5	42	0.084	8.4
6	1	0.002	0.2
7	1	0.002	0.2
8	29	0.058	5.8
9	57	0.114	11.4
10	21	0.042	4.2
11	42	0.084	8.4
12	19	0.038	3.8
13	3	0.006	0.6
14	22	0.044	4.4
15	6	0.012	1.2
16	39	0.078	7.8
17	6	0.012	1.2
18	19	0.038	3.8
19	138	0.276	27.6
20	13	0.026	2.6
21	25	0.05	5
22	28	0.056	5.6
23	34	0.068	6.8
24	48	0.096	9.6
25	104	0.208	20.8

6.8.2 Effects on Errors in the Pursuer's Initial Speed

Similar to section 6.8.1, white-gaussian, $N(0, 1)$, noise was added to the pursuer's initial speed. During the search for feasible solution, the pursuer's initial states are assumed to be correct. Once the best solution is found by the optimisation algorithm, the best solution is tested again in a simulated game where the pursuer's initial speed is randomly perturbed. A total of 500 independent runs were conducted.

The outcome is 74.1% (371) of the runs gave a positive result (i.e. no interception). As an example, Figure 6.55 show 10 different trajectories flown by the evader using the same feasible trajectory found by EP. But the pursuer's initial speed is slightly perturbed using the white Gaussian noise. As seen in Figure 6.55, a small perturbation to the pursuer's velocity does not affect the outcome of the algorithm. The selected feasible solution is able to steer the evader away from interception until the sixtieth seconds.

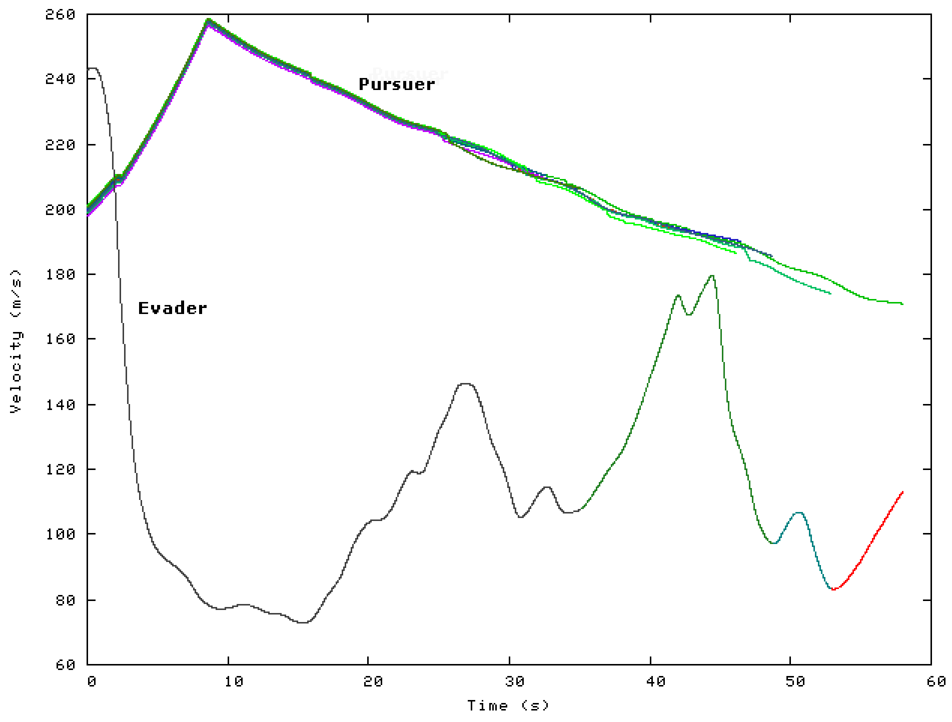


Figure 6.55: The deviation of the pursuer's trajectory when the initial speed is slightly perturbed. In this figure 10 runs are displayed.

6.9 Random Versus EP

Several simulations were conducted to study the difference between random search, standard-EP and meta-EP are employed. The simplest search is the random search. It simply generates new solutions randomly at every generation cycle and retains the best solution. Standard-EP does not use any self-adaptation mechanism to evolve. However, meta-EP uses self-adaptation mechanism to evolve candidate solutions to search for a better one.

The general parameters used in the experiment are given in Table 6.20. The aircraft initial states are given in Table 6.21. 200 independent runs were conducted for each case and the results are given in Figure 6.56.

Table 6.20: General Parameters.

Parameter	Value
No of generations	100
Population Size	100
Integration Time Step	0.01 second
Duration	100 seconds
Migration Rate	0
PNG Ratio	1

Table 6.21: Aircraft Parameters.

Parameter	Evader	Pursuer
X (m)	3000	0
Y (m)	0	0
Z (m)	3000	3000
Heading (rad)	0	0
Initial Velocity (m/s)	250	200
Interception Radius (m)	n/a	15

In Figure 6.56, random search is ranked third in its ability to locate feasible solutions. Over 200 runs, the random search can only find an average of 2.7 feasible solutions with a standard deviation of 2.1. Interestingly, on average, standard-EP finds more feasible solutions, i.e. 525 feasible solutions with a standard deviation of 674 than meta-EP, i.e. 217 feasible solutions with a standard deviation of 158. The performance of the random search is abysmal compared to standard-EP and meta-EP. This shows that both EPs are able to guide the search in the right direction to find feasible solutions.

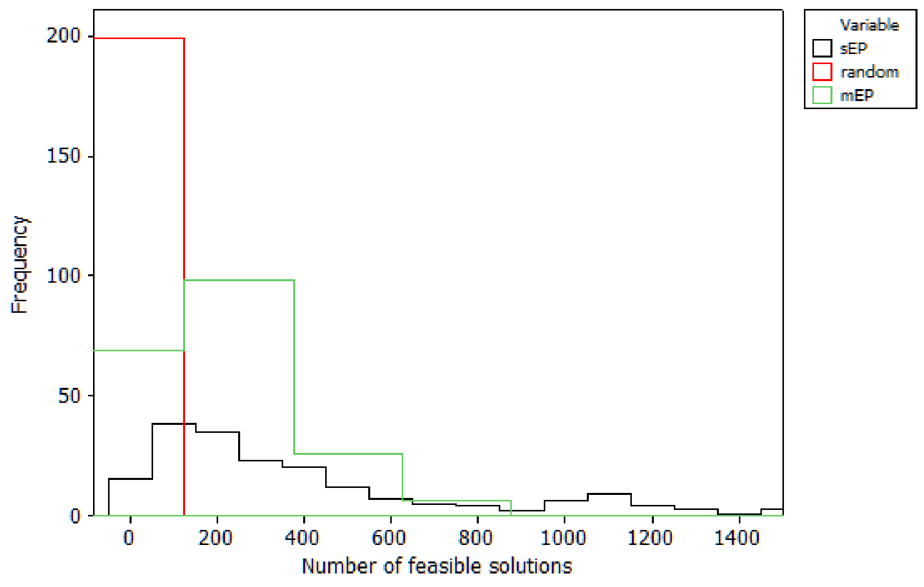


Figure 6.56: Number of feasible solutions found for standard-EP, meta-EP and Random.

In terms of reliability, meta-EP is found to be more consistent in locating feasible solutions than a random search and standard-EP. For the random search, the minimum number of feasible solutions is 0 and the maximum is 10. This is very low compared to meta-EP and standard-EP. On the other hand, the minimum number of feasible solutions found by meta-EP is 10. But there are runs where standard-EP cannot find any feasible solution. This means that although standard-EP is able to find more feasible solutions at certain times, the reliability is still an issue. Instead, meta-EP seems to be the most reliable in locating feasible solutions.

Overall, the use of Evolutionary Programming (EP) algorithms is able to improve the search space. This is done by comparing the number of feasible solutions found with random search and EP variants.

6.10 Parallel Implementation

The effects of migration rate and the number of processors on the performance of the EP algorithm are studied in this section. The quality of parallel EP depends on the choices of topology, migration rates and the size of demes [20]. It is made mode complicated because of the nonlinear relationship between each parameter.

6.10.1 Effect of Migration Rate

An experiment is carried out to find the contribution made by the migration rate to the quality of the solution. Figure 6.57 and Figure 6.58 show the effect of migration on the number of feasible solutions and the execution time, respectively.

The effect of migration rate is found to be insignificant at lower population sizes (30-100) in locating the feasible solutions. As the size of the population increases over 100, higher migration rates will be able to locate more feasible solutions. Zero migration has the lower success rate to find feasible solution as seen in Figure 6.57.

The higher number of feasible solutions found is due to more feasible solutions being migrated to adjacent demes and many good offsprings are produced from the parent which are already having a higher fitness value. The diversity of the search space is maintained because the number of feasible solutions is usually small and other sub-optimal solutions are retained and mutated to produce offsprings for the next generation.

The execution time is found to be unpredictable as depicted in Figure 6.58. Although the execution time increases as the population time increases, its effect to the migration rate is not apparent. The same observation is also true in Figure 6.59. In the figure, due to the highly competitive selection in EP, the fitness value increases as the population size increases. But, the migration rate does not seem to influence much to the fitness value. However, in this case, the migration rates does not seem to have much influence on the fitness value.

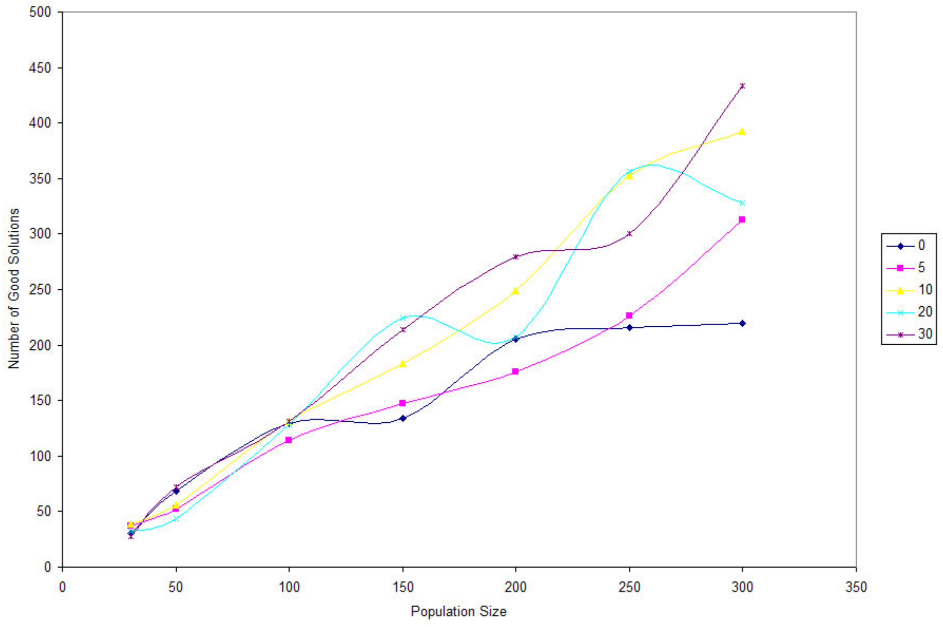


Figure 6.57: Effect of migration rate on the number of feasible solutions.

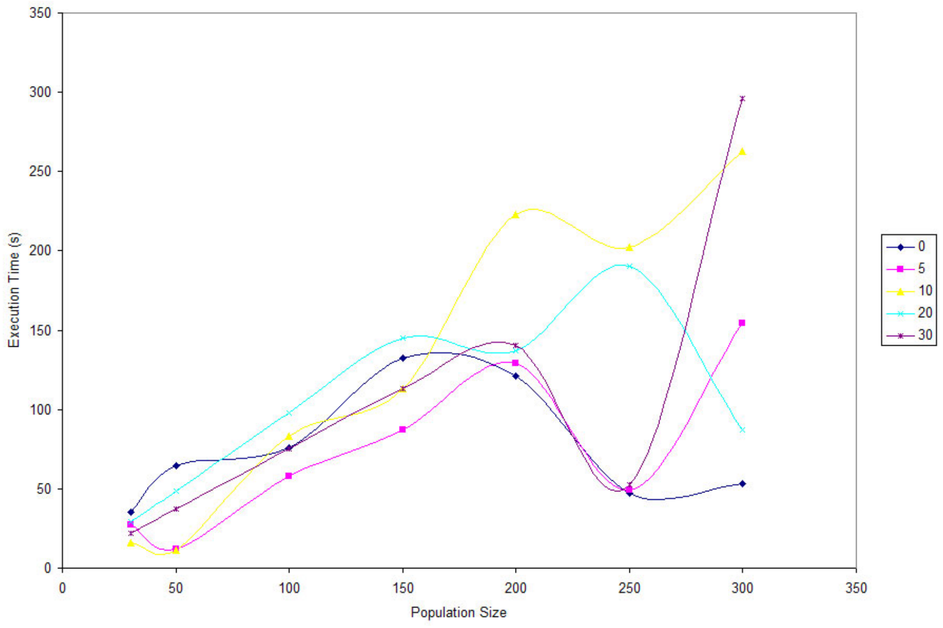


Figure 6.58: Effect of migration rate on the execution time.

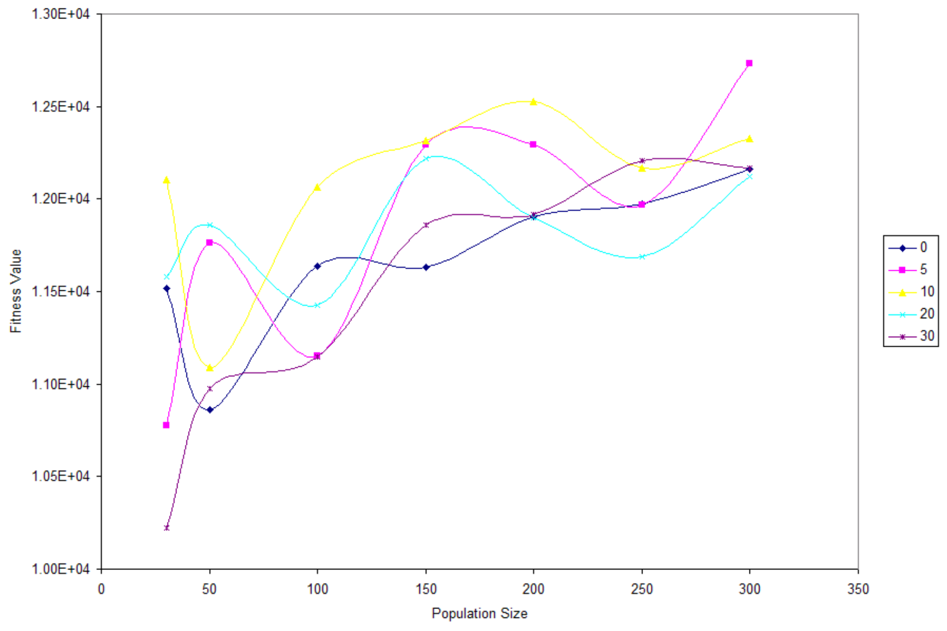


Figure 6.59: Effect of migration rate on the fitness values.

6.10.2 Speedup

A speedup is an important performance indicator for a master-slave approach. It shows the efficiency of the parallel implementation. The analysis of speedup for the master-slave approach is conducted in this section.

The general parameter and the aircraft parameter of the experiment is given in Table 6.22 and 6.23. Three population sizes, i.e. 100, 200 and 500 are chosen to measure the speedup. The results are given in Figure 6.60.

The maximum speedup of 4 is achieved for population sizes of 100. It reaches the maximum speedup if 8 processors are used. Beyond 8 processors, additional number of processors do not improve the speedup. For a population size of 200, the maximum speedup of 6 is achieved using 12 processors. For a population size of 500, the maximum speedup achieved is 9 for 14 processors. A steady increase of the speedup is observed when the population size increases as shown in Figure 6.60. An almost linear increment of the speedup versus the population size can be seen in Figure 6.61.

This shows that for master-slave parallel applications, there is a limit on the number of processors that gives the highest speedup. At this level, the processor utilisation is higher than the cost of communication. This will result in having a better speedup. Beyond that point, if more processors are used the speedup will

Table 6.22: General Parameter.

Parameter	Value
No of generations	100
Population Size	100,200,500
Integration Time Step	0.01 second
Duration	100 second
Migration Rate	0
PNG Ratio	1

Table 6.23: Aircraft Parameter.

Parameter	Evader	Pursuer
X (m)	1524	0
Y (m)	0	0
Z (m)	304.	1000.0
Heading (rad)	0	0
Initial Velocity (m/s)	243	200
Interception Radius (m)	n/a	15

decrease. The frequent communications offset the reduction in the time required to evaluate the solutions. The evaluation time is dominated by the communication time between processors. One way to improve the speedup is to increase the communication speed.

The speedup is observed to increase linearly in accordance with the size of the population. For smaller population sizes, a moderate size of around 8 processors is sufficient. For higher population sizes, more processors can be utilised to achieve a better speedup. However, the determination of the optimal number of processors can only be determined through trial and error.

The results in Figure 6.62 use a coarse-grained approach. It uses a fixed total population size. A total population size is a multiplication of the number of generations, the population size of each generation and the number of processors used. 9 sets of number of processors are used in this study. The number of generations, population size and the number of processors are given in Table 6.24.

A linear increment is observed for a coarse-grained configuration when a predetermined total population size is used. Since the total population size is kept constant, the increase in the number of processors means each processor will then evaluate fewer individuals. Since there is minimal communication among processors for coarse-grained approach, the addition of more processors will certainly improve the computing time.

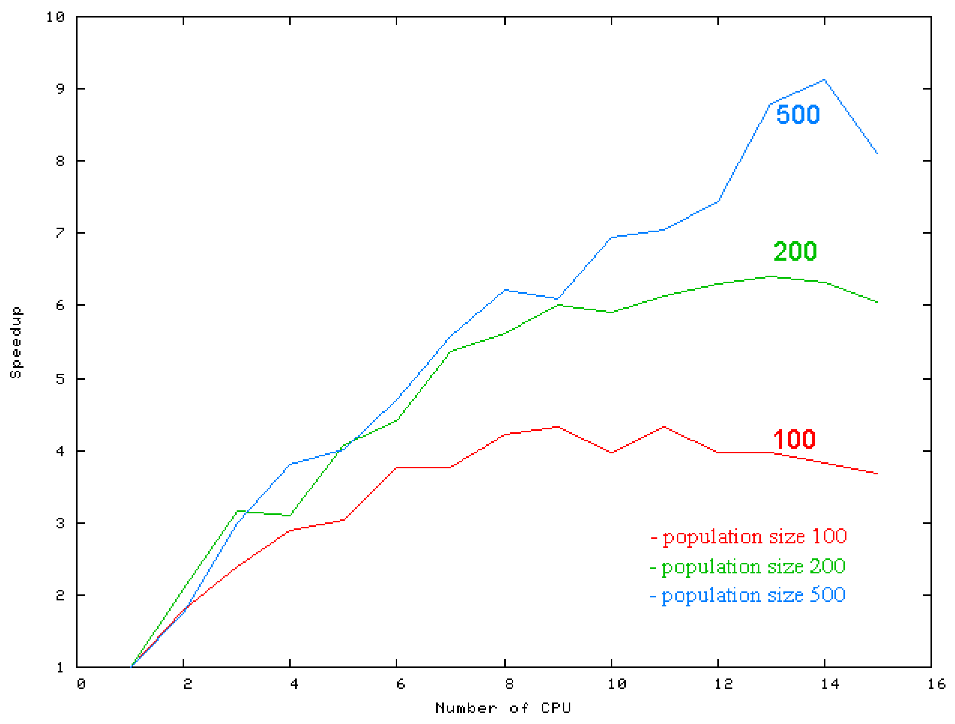


Figure 6.60: Speedup of different population size (100, 200 and 500). (Number of Generation = 100).

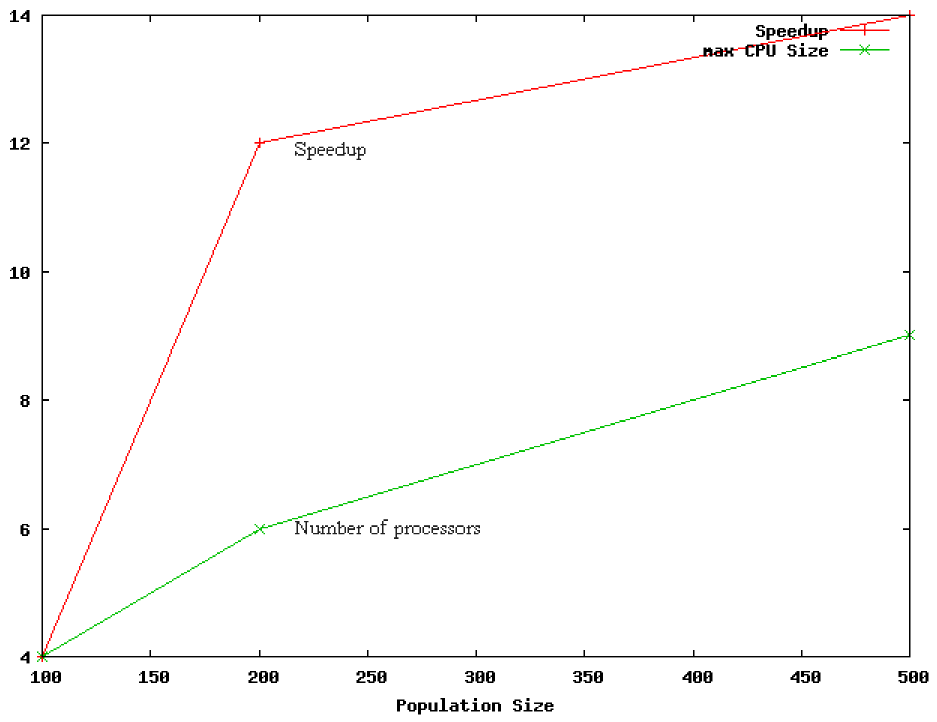


Figure 6.61: Maximum speedup and processor number against the population size for master-slave configuration.

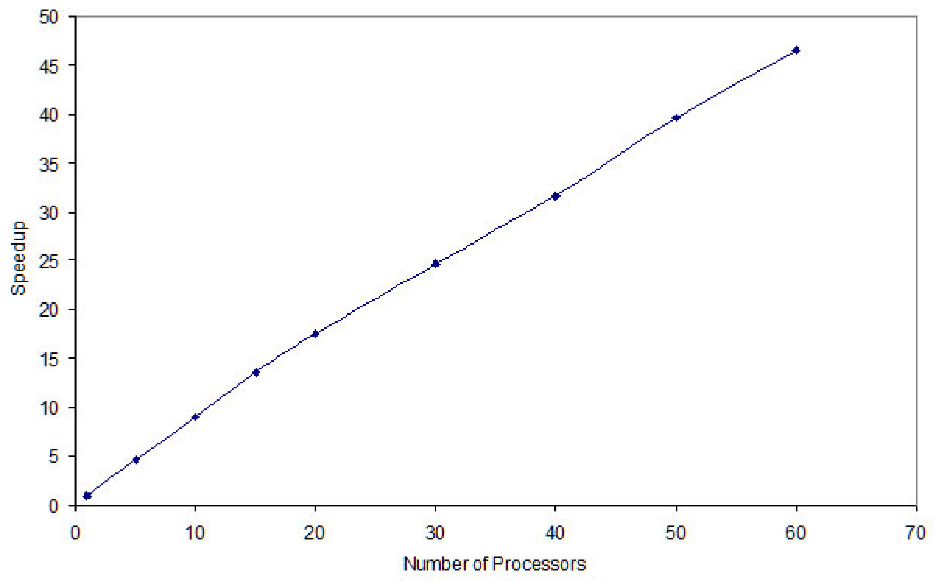


Figure 6.62: Speedup for different number of processors in coarse-grained configuration with total population size 50,000.

Table 6.24: Parameters and results for coarse-grained configuration.

No.	No. of Generations	Population Size	No. of Processors	Computing Time (s)	Speedup
1.	500	100	1	792	1
2.	100	100	5	169	5
3.	100	50	10	87	9
4.	101	33	15	58	14
5.	100	20	20	46	18
6.	104	30	30	32	25
7.	125	40	40	25	32
8.	100	50	50	20	40
9.	83	60	60	17	47

6.11 Summary

The search for feasible solutions in an air combat proves to be challenging. Namely, the fitness function itself is time-variant. A small change in both environmental variables and design variables would significantly alter the outcome of EP. Each variable plays an important contribution towards locating the feasible location in the near infinite search space. Thus each run is unique by itself and the results can only be interpreted through statistical analysis.

This chapter discusses the results of the simulations of selected initial states and conditions. It is beyond the scope of the thesis to study the whole spectrum of the problem. The first section explains the type of aircraft used by the players. Three scenarios have been explored. The first scenario is tail encounter. In this scenario, the pursuer is located behind the evader. The second scenario is when the pursuer is initially located at the side of the evader and the last scenario is when the pursuer and the evader are facing each other.

In most cases, the feasible solution aims to saturate the turning rate of the pursuer. This causes the pursuer to miss the evader in a small margin. Sometimes, it misses the evader by only a few meters beyond its predefined lethal radius. This is an achievement because the search of the feasible solution takes place without having any prior knowledge of the type of feasible manoeuvres.

The next is the discussion on the type of hardware used to run the simulations. The computing time is discussed in relation to the time needed to compute a solution. The next stage is to run the optimisation algorithm for the three cases described in 6.5.

Some of the encounters are quite extreme in which only suitable for unmanned aircraft. For example, the aircraft might be performing a continuous loop at a very high angular rate. Other example is turning at a load factor over than 9. Using the same cases, other factors that influenced the quality of the solutions are also discussed in this chapter. One example is the robustness that is calculated by running the same problem multiple times using a statistical study.

The effect of errors is also studied. The feasible solutions are tested by perturbing the initial positions of the pursuer. For small perturbations, their effect on the feasible solution is small and insignificant. But if the perturbation is large, the outcome can be totally different, i.e. the solution is no longer feasible. The spawn time does play an important role in locating the feasible solution.

For a short spawn time, it is likely that the optimisation algorithm has a high chance of locating feasible solutions. The reason is that the probability of violating the constraints is less for a short spawn time simulation. If the spawn time is longer, there is a strong possibility that the aircraft will violate the constraints. This will make the optimisation algorithm more susceptible to violate the constraints which will result in finding less feasible solutions.

Since the algorithm is written in FORTRAN and parallel-ready, the evaluation of the solutions are performed on multiple processors. As discussed in section 6.10.2, the speedup can reach up to 9 for a population size of 500 (see Figure 6.60).

Overall, the effective manoeuvre to evade interception is to turn hard and dive [103]. However, locating feasible trajectories are highly dependent on many factors and very hard to predict. Quoting from [103], “Missile defense often requires instant analysis and rapid reactions. The tactics to be employed in any conceivable situation must be predetermined and practiced so that they become automatic. Once the missile is launched, it is too late for leisurely development of a response.”

Depending on the initial conditions and configuration, EP is found to be able to locate feasible solution in a relatively short period of time. With the advent of computing power, this statement is not far from reality.

7 Conclusion

7.1 Summary

Efforts to find an efficient and fast algorithm to solve pursuit-evasion problem is undoubtedly, a continuous process. With the advent of computing power, highly capable aircrafts, highly lethal weapons, the workload of a pilot keeps increasing. The problem has been described by many using various techniques, such as differential game theory, expert systems, heuristic value driven systems, discrete dynamic games, and influence diagram game. The objective an air combat has always been to find the best solution for the targeted player to avoid interception i.e. the evader.

In this thesis, the use of Evolutionary Programming (EP) is applied to solve an air combat problem. It has been described in the previous chapters that EP is able to find the optimal solution for the evader against a highly agile pursuer within a reasonable computing time. The computing time is further improved by running a parallel evaluation using several concurrent computers

It is found that the computing time is dependent on the EP parameters, the initial states and the capabilities of the players. The test cases follow those used by [68], see Section 6.5. Feasible solutions have been found, but to locate feasible solutions is still a challenge. However, this mission is not impossible. The ability to search for feasible solutions is dependent on various factors:

1. Initial states - These will determine the ability for EP to search for feasible solutions. If the initial separation distance is too close, it may be difficult to locate feasible solutions. The initial separation distance must be far enough to allow the evader to perform the required optimal manoeuvre.

2. Vehicle performance - Performances such as the maximum load factor, the thrust, the interception radius, and the aerodynamic characteristics are important. The players have to be properly modeled as close as possible to the real vehicle.

3. Search Parameters - The parameters used in EP, such as the size of the population, the number of generations, the migration rate and the number of processors influence the ability of EP to locate feasible solutions.

4. External factors. These are external disturbances, instrument noise and other types of noise that would affect the quality of feasible solutions.

The innovative approach in this study is to move away from typical approaches, i.e. by representing the vehicle in its natural differential equation forms without any major modification made to the mathematical equations. The fidelity of the mathematical model can be increased by adding more state variables into the equations.

This approach is suitable for scenarios where "*a conflict is never the same twice*" [37]. This is achieved by running a new optimisation run every time the situation changes. This is possible because EP could search the solution space in a relatively short period of time.

Two consecutive developments have been conducted. Initially, the first development used three degree of freedom models for both players. Then, the second development used a six degree of freedom model to model the evader and a three degree of freedom model to model the pursuer.

The 3DOF model was initially used to model the evader and the pursuer. The pursuer was more agile than the evader by having a much more powerful thrust and a higher load factor. The pursuer targeted the evader by using a well known proportional navigation guidance system.

The solution space of the 3DOF problem is a vector that represents the change of heading angle and of flight path angle for the evader over a three dimensional space. The solution is presented in the temporal dimensions since the changes are performed at every second. Thus, a minor perturbation could largely alter the original trajectory.

The results indicate that finding optimal solutions depends on the players' initial states and their configurations. If the interception radius is small, the chances the evader could find optimal solutions will be higher than if the interception radius is larger.

A 6DOF model was used to represent the evader, a high performance fighter aircraft. The use of 6DOF model for the evader has added a higher level of complexity to the problem. The integrating time step had to be reduced in order to get a stable result, i.e. 0.01 second. This contributes to the increasing the computing time.

The limitation of the 3DOF model is that the attitude motions are not calculated. Since the 3DOF model uses a highly simplified aircraft data; in order to increase the realism, the analysis proceeds to develop the 6DOF model of the evader. Due to the complexity in designing the control system of the pursuer in 6DOF, the pursuer model remains in 3DOF. The use of a 6DOF model for the evader has added a higher level of complexity to the problem. The integrating time step has to be reduced in order to get a stable result, i.e. 0.01 second. This contributes to the increasing length of the computation time.

The benefit of using a 6DOF model is shown in Section 5.7. The input variables are the controls used by the pilot, i.e. the control surfaces and the throttle setting. The solution of the problem is represented in a temporal dimension rather than in a spatial dimension. This is required since the problem is highly dynamic in which both players move in the 3D search space.

Several typical evasion manoeuvres were observed from the feasible solutions. They are dependent to the situations such as, if the pursuer is close, the evader would choose not to run away, but instead try to sneak into the pursuer's turning radius. Since the pursuer's velocity is very high at the initial time, such a manoeuvre is logical. Another way is to dive close to the ground and climb up. This manoeuvre could cause the pursuer to hit the ground whilst trying to turn and climb.

If the pursuer is located farther away, the best option will be to run away by diving to a lower altitude and manoeuvring in an optimal way to out-manoeuvre the pursuer. This will definitely increase the evader's kinetic energy as well as making it easier for the pursuer to bleed off its energy. During diving, whilst the pursuer is pursuing by using a PNG system, the evader performs weaving. The motivation is to stretch the lateral acceleration of the pursuer to the maximum and lose its energy faster. At the end of the encounter, the velocity of the pursuer is lower than the evader and interception is no longer possible.

The time required to search for optimal solutions can be further reduced by using parallel processing. The advantage of using parallel processing in locating optimal solution has also been studied. The overall computation time is reduced if multiple processors are used to evaluate the solutions. However, the speedup does not show a linear relationship. This is due to the serial components present in the process. Speedup has been found to have a correlation with the size of the population being studied. The impact of communication is minimised by exchanging data in a larger block. Thus, fewer communications are required and this translates into faster computation time.

Two parallel approaches were studied, i.e. master-slave and coarse-grained. The paradigm of coding in parallel is difficult and it is further complicated by having to code the algorithm separately to suit master-slave and coarse-grained approaches. This is because since they require different programming techniques. To select the suitable MPI functions for sending and receiving variables is difficult and the chances to get logical errors are high. Careful coding and a lot of hours of debugging are essential to make the algorithm to work as intended.

The maximum number of processors in the master-slave configuration is governed by the speedup as shown in Figure 6.60. In this study, the selection of the optimal number of processors is done through trial and error. The speedup is influenced by the speed of the processor as well as the speed of the communication. Different computer architectures may give a different speedup.

The coarse-grained approach has also been studied. The number of demes does influence the quality of the solutions. Namely, it influences the diversity of the search space which, in turn, influences the quality of the solutions. In this approach, each processor will create its own deme and runs a complete EP cycle. At the end of a cycle, a number of selected solutions will migrate to adjacent deme. Finally, all demes will send their best solution to the master process for the final selection.

The effect of noise has also been studied. It is found that a smaller time horizon could reduce the noise effect. This is due to the fact that the optimisation algorithm always recalculates the optimal solutions at the new time horizon using the previously known noise states.

Further analysis of the algorithm shows that many variables are important for EP to perform as expected. The variables are:

1. The size of the population.
2. The number of generations.
3. The number of processors used.
4. The initial states of both players.
5. The simulation (spawn) time.

6. The noise and errors involved.

7. The players parameters.

The size of the population, the number of generations and the number of processors are found to influence the search space. As a rule of thumb, the chances to find feasible solutions is higher if the population size, the number of generations and the number of processors are increased. But these increments sacrifice the computing time because more time is needed to evaluate each solution. Thus, a trade-off is needed in order to have an adequate number of solutions without sacrificing the ability to explore the search space and the computing time.

The initial states of the players are important because if the initial separation distance is too close, for example 1 km, the evader has very little space to manoeuvre. In the study, an initial range of 5-20 km is sufficient to find feasible trajectories. But the altitude also plays an important factor. If the altitude is low, the feasible trajectory is to fly close to the ground. The motive is to cause the pursuer to over steer and crash to the ground. It is quite difficult to evade at a higher altitude because there is ample space for the pursuer to manoeuvre.

7.2 Conclusion

This thesis explores the possibility of using Evolutionary Programming as an intelligent agent in an air combat problem. The programme acts as a surrogate pilot and performs optimal behaviour through introspection via simulation.

An actual aircraft platform is selected to be the evader in the effort to increase the realism of the analysis. The analysis is a nonlinear six degree of freedom pursuit-evasion problem. The problem is made even more complicated by employing parallel processing. A novel way of representing the control variables has been presented. The control variables are presented in the form that EP could manipulate effectively in order to find an optimal control solution. In contrast to other optimisation algorithms such as artificial intelligence, multi-stage influence diagrams and differential games, EP searches for the window of opportunity, even if lasts only for a few seconds. The opportunity is represented by a temporal window where if a manoeuvre is performed at that point in time, there will definitely be no interception. If the same manoeuvre is performed too soon, the evader will be intercepted. Similarly, if the manoeuvre is performed too late, interception will be inevitable. Thus, the timing to perform a manoeuvre is important to avoid interception. Without having to explicitly list optimal manoeuvres such as explained by [103], the optimal solutions found by EP will perform such manoeuvres.

Most of the contributions of this research were made possible by focusing on realism and simplicity. Realism is accomplished by employing an aircraft using six degree of freedom equations of motion with actual aerodynamics and 'as it is' stability data. Thus, the original structure of the mathematical model is maintained. This configuration provides room for improvement. One could add more functions, noise and weaponry to the existing code in the effort to increase its realism without having to worry about singularity, discontinuity and other mathematical obstacles.

Simplicity is achieved by performing the search for feasible solution in EP by directly interact with the aircraft control inputs, performing a simulation of pursuit-evasion game and getting the outcome. The control resembles the act of a pilot controlling the aircraft using the control stick. The outcome of the simulation is the fitness value used by EP.

In particular, this thesis made the following contribution to our understanding of pursuit-evasion problem by illustrating that Evolutionary Programming algorithms can find solutions to the highly non-linear, discontinuous, poorly behaved space of simulated air combat evasion. It has been demonstrated that the air combat missile evasion problem can be modelled as a complex search problem across a simulation of the engagement. This is possible by providing an adequately realistic simulation that involves models that are traditionally intractable thus creating very difficult search environment for many algorithms. However, Evolutionary Programming approaches have been used to formulate and solve this complex search problem and find feasible solutions quickly.

At the same time, parallel processing can reduce the search time for a feasible solution that is practical (near real-time). For the problem under consideration, parallel processing can reduce the time to find a feasible solution with *X* processors by a few seconds with a *Y* percent improvement in search quality. *X* being the number of processors that has been shown experimentally to be the degree of parallelisation typically best suited for the formulation of this problem. The issue of practicality of the suggested hardware configuration within future combat aircraft mission systems is beyond the scope of this thesis although the proliferation of low end multi-cored processors and a high degree of parallelisation utilized for supercomputing computations do give some cause for optimism.

There are many possible applications for this algorithm. If the aircraft and missile models are performing well and accurate enough, one possibility is to use it as an advisory or an automatic evasion system to help pilots during missiles threat. Similarly, it can be used on board a UCAV as an automatic evasion system to avoid interception from missiles. Another possible application is for pilot training purposes. It can be used to optimise the evader and the pilot who acts as the pursuer will try to intercept it. The performance of a new aircraft can also be tested using this algorithm. One possible application is to evaluate the performance of the aircraft against various types of missiles. This in turn will produce a comparative study on the performance and agility of the aircraft against known missiles at various conditions.

7.3 Future Work

There are many improvements that need to be investigated further in the pursuit evasion problem using Evolutionary Programming. Mainly, the improvement covers issues related to the robustness, accuracy, efficiency, and time-efficiency. The following improvements are suggested:

Improvement on the mathematical model One of the difficulty in this research is to find suitable aircraft data for 6DOF simulation. These equations form the basis of the objective function. Finding a good and a comprehensive aircraft data is important to ensure the validity of the feasible solutions. Apart from that, the use of the improved mathematical model of the pursuer (i.e. missile) will definitely increase the realism of the simulation.

Deriving expertise from playing against unknown opponent This is achieved by upgrading the algorithm to bootstrap expertise from past encounter against an unknown opponent. The Evolutionary Algorithm will learn from mistakes in the past and store the successful manoeuvres in the memory for later use. A proper information storage representation has to be devised and tested. For example, a newly developed missile can be tested against the Evolutionary Algorithm.

Validation of the mathematical model The mathematical model developed can be validated using established models developed independently by others, such the air force. The model can be integrated with existing simulation facilities to assist pilot to evade incoming missiles.

Improving the execution time The execution time can be improved further by optimising the source code. This is done by using a faster processor and a faster communication device. Optimising the execution time of the serial part of the algorithm is expected to improve the speedup. The use of different parallel topologies could also influence the performance of EP, such as by using hierarchical parallel approach suggested by Erick [35].

Hybridisation To accommodate the dynamic and complex nature of real world problems, we can integrate various evolutionary methods into a more powerful hybridised EAs to solve the problems.

Variable population size and mutation rate The effect of varying the population size and the mutation rate on the performance of EP can be studied, especially in noisy environment as suggested by [2].

Besides that improving the mathematical model on the existing aircraft, the optimisation algorithm can also be used in preliminary aircraft design. The limits and performance of the conceptual design can be achieved by adjusting the aircraft parameters and evaluated by EP. This could save time and cost in the conceptual design phase.

The methodology could be expanded to move beyond that of a pursuer-evader problem by including the possibility of role switching. In reality, in an air combat scenarios between two aircraft the evader may try to manoeuvre into a position where by it can achieve a tactical advantage and thus the roles are switched.

However, this study has yet to address the problem of several air vehicles engaged in air combat. The methodology however can be expanded to handle these types of scenarios.

References

[1] F.P. Adler. Missile guidance by three-dimensional proportional navigation. *Journal of Applied Physics*, 27(5):500–507, 1956. 23

[2] A. N. Aizawa and B. W. Wah. Dynamic control of genetic algorithms in a noisy environment. In *Conf. Genetic Algorithm*, pages 48–55, 1993. 152

[3] S.M. Amin, E.Y. Rodin, M.K. Meusey, T.W. Cusick, and A. Garcia-Ortiz. Evasive adaptive navigation and control against multiple pursuers. In *American Control Conference, 1997. Proceedings of the 199*, volume 3, pages 1453 – 1457, 6 1997. 18

[4] Peter J. Angeline. Tracking extrema in dynamic environments. In *Proceedings of the 6th International Conference on Evolutionary Programming*, 1997. 28

[5] T. Back, D.B. Fogel, and T. Michalewics. *Evolutionary Computation 1 : Basic Algorithms and Operators*. Institute of Physics Publishing, 2000. 5, 12, 34, 36

[6] Thomas Back. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996. 12, 32, 34

[7] J. Baltes and Y. Park. Comparison of several machine learning techniques in pursuit-evasion games, 2001. 21, 29

[8] Brian Barrett, Jeffry M. Squyres, and Andrew Lumsdaine. Integration of the LAM/MPI environment and the pbs scheduling system. In D. Senechal, editor, *Proceedings of the 17th International Symposium on High Performance Computing Systems and Applications and OSCAR Symposium*, pages 277–283, 2003. 78

[9] Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. SIAM, 1999. 2, 3

[10] Thomas Bäck. *Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms*. New York : Oxford University Press, 1996. 37, 38, 39

[11] A. D. Bethke. Comparison of genetic algorithms and gradient-based optimizerson parallel processors: efficiency of use of processing capacity. Technical report, Tech. Rep. 197, University of Michigan, Logic of Computers Group, Ann Arbor, MI, 1976. 44

[12] A.I. Blagodatskikh. Weak evasion of a group of coordinated evaders. *Journal of Applied Mathematics and Mechanics*, 69(6):891–899, 2005. 11

[13] J.H. Blakelock. *Automatic Control of Aircraft and Missiles*. John Wiley & Sons, New York, 1991. 13, 72

[14] H. C. Braun. On solving travelling salesman problems by genetic algorithms. *Parallel Problem Solving from Nature*, pages 129–133, 1990. Berlin, Springer-Verlag. 49

[15] M. Breitner, H. Pesch, and W. Grimm. Complex differential games of pursuit-evasion type with state constraints. 10

[16] M. H. Breitner, H. J. Pesch, and W. Grimm. Complex differential games of pursuit-evasion type with state constraint, part 1 necessary conditions for optimal open-loop strategies. *J. Optim. Theory Appl.*, 78(3):419–441, 1993. 11

[17] B.W. Brown and J. Lovato. Randlib90 fortran 95 routines for random number generation. http://hpux.cs.utah.edu/hppd/hpux/Maths/Misc/randlib-1.3/, 2000. 77

[18] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, UK, 1999. 52

[19] Mark E. Campbell, Jin woo Lee, and Eelco Scholte. Simulation and flight test of autonomous aircraft estimation, planning, and control algorithms. *Journal of Guidance, Control and Dynamics*, 30(6):1597–1609, Nov-Dec 2007. 23

[20] Erick Cantu-Paz. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD thesis, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportaion Building, 104 S. Matthew Ave, Urbana, IL 61801, July 1999. 44, 46, 47, 48, 49, 136

[21] Erick Cantu-Paz and David E. Goldberg. Efficient parallel genetic algorithms: Theory and practice. *Genetic Evolutionary Computation*, 186, 2000. 30, 41

[22] Y. Uny Cao, Alex S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 1:0, 0 1997. 29

[23] Brian J. Capozzi. *Evolution-Based Path Planning and Management for Autonomous Vehicle*. PhD thesis, University of Washinton, US, 2001. 6

[24] S.M. Cheang. *Genetic Parallel Programming*. PhD thesis, The Chinese University of Hong Kong, 2005. 11, 26, 30, 32, 40

[25] Han-Lim Choi, Hyo-Choong Bang, and Min-Jea Tahk. Co-evolutionary optimization of three-dimensional target evasive maneuver against a proportionally guided missile. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 1406–1413, 2001. 12, 18

[26] J. Cohoon, S. Hegde, W. Martin, and D. Richards. Punctuated equilibria: A parallel genetic algorithm. In J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. Lawrence Erlbaum Associates, 1987. 48

[27] DARPA. Unmanned combat air vehicle advanced technology demonstration, phase 1. MDA972-98-R-003,DARPA/TTO, 3701 N. Fairfax Drive, March 9 1998. 4

[28] Charles Darwin. *The Origin of Species*. Gramercy, May 1995. 28

[29] K. De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988. 37

[30] Dany Dionne. *Integrated Detection, Estimation, and Guidance in Pursuit of a Maneuvering Target*. PhD thesis, Department of Eletrical and Computer Science, McGill University, Montreal, 2005. 20

[31] US DoD. U.s. standard atmosphere. Prepared under the sponsorship of the National Aeronautics and Space Administration, the United States Air Force and the United States Weather Bureau, Washington, D.C., December 1962. 61

[32] M. Eaton, M. McMillan, and M. Tuohy. Pursuit-evasion using evolutionary algorithms in an immersive three-dimensional environment. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 2, pages 348–353, 10 2002. 29

[33] Harri Ehtamo and Tuomas Raivio. Applying nonlinear programming to pursuit-evasion games. *Systems Analysis Laboratory Helsinki University of Technology*, 0:0, 0 0. 24, 26, 31, 53

[34] T. M. English. Some information theoretic results on evolutionary optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.*, 1999. 77

[35] C. Erick. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1999. 47, 152

[36] Hamidreza Eskandari. *Multiobjective Simulation Optimization Using Enhanced Evolutionary Algorithm Approaches*. PhD thesis, Department of Industrial Engineering and Management Systems, College of Engineering and Computer Science, University of Central Florida, Orlando, Florida, 2006. 27

[37] Patrick Beautement et. al. Autonomous agents and multi-agent systems (aamas) for the military - issues and challenge. In S. G. Thompson and R. Ghanea-Hercock, editors, *DAMAS 2005, LNAI 3890*, Berlin, 2006. Springer-Verlag. 146

[38] Imado F. Some aspects of a realistic three-dimensional pursuit-evasion game. *Journal of Guidance, Control, and Dynamics*, pages 289–293, 1993. 21, 37, 180

[39] G. S. Fishman. *Concepts and Methods in Discrete Event Digital Simulation*. John Wiley & Sons, New York, 1973. 53

[40] R. Fletcher. *Practical methods of optimization*. A Wiley Interscience Publication, Chichester, N.Y.: Wiley, 1987, 2nd ed., 1987. 25

[41] D. B. Fogel. An evolutionary approach to representation design. In D. B. Fogel and W. Atmar, editors, *Proceedings of the First Annual Conference on Evolutionary Programming*, pages 163–168, La Jolla, California, 1992. Evolutionary Programming Society. 27

[42] David B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press, Piscataway, NJ, USA, 1995. 28

[43] D.B. Fogel. *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*. Ginn Press, Needham Heights, 1991. 33

[44] D.B. Fogel and W. Atmar. Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, 63:111–114, 1990. 38

[45] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Real time motion planning for agile autonomous vehicles. *J. Guidance, Control and Dyncs.*, 25(1), 1 2002. 23

[46] A.S. Fukunaga and A.B. Kahng. Improving the performance of evolutionary optimization by dynamically scaling the evaluation function. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, page 182, 12 1995. 29, 30

[47] W. P. Gilbert, L. T. Nguyen, and R. W. Van Gunst. Simulator study of the effectiveness of an automatic control system designed to improve the high angle-of-attack characteristics of a fighter plane. Technical Report NASA TN D-8176, Langley Research Center, 1976. 56, 66

[48] J. J. Grefenstette. Parallel adaptive algorithms for function optimization. Technical report, Computer Science Dept., Vanderbilt University, Nashville, TN., 1981. 46

[49] P. B. Grosso. *Computer Simulations of genetic adaptation: Parallel subcomponent interaction in a multilocus model*. PhD thesis, The University of Michign, 1985. 49

[50] M. Guelman, J. Shinar, and A. Green. Qualitative study of planar pursuit-evasion game in atmosphere. *Journal of Guidance, Control and Dynamics*, pages 1136–1142, 1990. 21, 24

[51] J.M. Hebert. *Air Vehicle Path Planning*. PhD thesis, University of, 2001. 19

[52] Emilio J. Contreras Hernndez. *Applications Of The Evolutionary Programming Optimization Technique In Power Systems Planning And Operation*. PhD thesis, University of Puerto Rico, 2005. 40

[53] Y. C. Ho. Differential games, dynamic optimization, and generalized control theory. *Journal Optimization Theory and Applications*, 6(3):179–209, 1970. 1

[54] K. H. Hoffman and Arnd Meyer. *Parallel Algorithms and Cluster Computing : Implementations, Algorithms and Applications*. Springer, Berlin, 2006. 50

[55] K. Horie and B.A. Conway. Optimal fighter pursuit-evasion maneuvers found via two-sided optimization. *Journal of Guidance, Control, and Dynamics*, 29(1):0731–5090, 2006. 21, 22, 24, 29

[56] Jianjun Hu. *Sustainable evolutionary algorithms and scalable evolutionary synthesis of dynamic systems*. PhD thesis, Department of Computer Science and Engineering, Michigan State University, US, 2004. 39

[57] Imado, Fumiaki, Miwa, and Susumu. Missile guidance algorithm against high-g barrel roll maneuvers. *Journal of Guidance, Control, and Dynamics*, 17(1):123–128, 1994. 11, 37

[58] F. Imado. Some practical approaches to pursuit evasion dynamic games. *Cybernetics and Systems Analysis*, 38(2), 3 2002. 26, 37

[59] Rufus Isaac. *Differential Games*. R.E. Krieger Pub. Co., 1975. 2, 10, 11, 22, 85

[60] B. Jarmark and C. Hillberg. Pursuit-evasion between two realistic aircraft. *Journal of Guidance, Control, and Dynamics*, 7(6):690–694, 1984. 21

[61] B. Jarmark and C. Hillberg. Pursuit-evasion between two realistic aircraft. *Journal of Guidance, Control, and Dynamics*, pages 690–694, 1984. 21, 24, 37

[62] Youchu Jin and Jrgen Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions On Evolutionary Computation*, 9(3):303–317, June 2005. 42, 130

[63] Kenneth A. De Jong. *Evolutionary Computation : A Unified Approach*. MIT Press, Cambridge, MA 02142, 2006. 27, 35, 39, 44, 48, 49

[64] H. Julie and J. B. Pollack. Parallel genetic programming and fine-grained simd architecture. In Siegel E. V. *et al*, editor, *Working Notes the AAAI Symposium Genetic Programming*, pages 31–37, Cambridge, MA, 1995. MIT Press. 30

[65] P.T. Kabamba, S.M. Meerkov, and F.H. Zeitz. Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking. *Journal of Guidance, Control, and Dynamics*, 29(2), Mar-Apr 2006. 19

[66] Janne Karelahti, Kai Virtanen, and Tuomas Raivio. Near-optimal missile avoidance trajectories via receding horizon control. *Journal of Guidance, Control and Dynamics*, 30(5):1287–1298, Sep-Oct 2007. 123

[67] D. E. Kirk. *Optimal Control Theory, An Introduction*. Prentice Hall, 1970. 2

[68] R. Lachner, M. Breitner, and H. Pesch. Three-dimensional air combat: Numerical solution of complex differential games. *Annals of the International Society of Dynamic Games, uni-bayreuth.de*, 1996. xi, 86, 89, 145

[69] Steven Landsburg. Nash equilibria in quantum games. RCER Working Papers 524, University of Rochester - Center for Economic Research (RCER), February 2006. 10

[70] E. Lazarus. The application of value-driven decision-making in air combat simulation. In *Proceeding of IEEE International Conference on Systems, Man and Cybernatics*, pages 2302–2307, Orlando, FL, 1997. 21

[71] Dongxu Li. *Multi-Player Pursuit-Evasion Differential Games*. PhD thesis, The Ohio State University, 2006. 29

[72] Jing Li, Jinhua Wu, and Shugui Kang. Learning evasive maneuvers using genetic-annealing algorithms. In *in Proc. The Sixth World Congress on Intelligent Control and Automation*, pages 6432–6435, 2006. 28

[73] Yong Liang. *Accelerated Strategies of Evolutionary Algorithms for Optimization Problem and Their Application*. PhD thesis, The Chinese University of Hong Kong, 2003. 29

[74] Shen Hin Lim, T. Furukawa, G. Dissanayake, and H.F. Durrant-Whyte. A time-optimal control strategy for pursuit-evasion games problems. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Vol.4, Iss., April 26-May 1, 2004*, pages 3962–3967, 2004. 11

[75] W. Austyn Mair and David L. Birdsall. *Aircraft Performance*. Cambridge Aerospace Space Series 5. Cambridge University Press, 1992. 64

[76] G. A. Mandt and T. L. Neighbor. Air-to-air missile avoidance. In *Guidance and Control Conference*, pages 124 – 131, 1982. 16, 17, 18, 85, 93

[77] E. Mayr. *Toward a New Philosophy of Biology: Observations of an Evolutionist*. Belknap Press, Cambridge, MA, 1988. 27

[78] J. W. McManus and K. H. Goodrich. Application of artificial intelligence (ai) programming techniques to tactical guidance for fighter aircraft. *AIAA Paper*, pages 89–3525, 1989. 21

[79] L. Meirovitch and I. Tuzcu. Control of flexible aircraft executing time-dependent maneuvers. *Journal of Guidance, Control, and Dynamics*, page 28, Nov-Dec 2005. 4

[80] P. K. A. Menon. Near-optimal midcourse guidance for air-to-air missiles. *Journal of Guidance, Control, and Dynamics*, 13(4):596–602, 1990. 11

[81] A. Miele. *Theory of Flight Paths*. Addison-Wesley Publishing Co., 1962. xi, 59, 60, 61

[82] F. W. Moore and O. N. Garcia. A new methodology for optimizing evasive maneuvers under uncertainty in the extended two-dimensional pursuer/evader problem. In E. Santos Jr., editor, *Proceedings of the Ninth IEEE*

International Conference on Tools with Artificial Intelligence (ICTAI-97), 1997. 11, 18, 25, 26, 30

[83] Sandeep Mulgund, Karen Harper, and Greg Zacharias. Large-scale air combat tactics optimization using genetic algorithms. *Journal of Guidance, Control, and Dynamics*, pages 140–142, 2001. 27

[84] John Nash. Equilibrium points in n-person games. In *Proceedings of the Nasiotnal Academy of Sciences of the United States of America*, volume 36, pages 48–49, 1950. 2

[85] T. J. Naylor, J. L. Balintfy, D. S. Burdick, and K. Chu. *Computer Simulation Techniques*. Wiley, New York, 1966. 53

[86] Robert C. Nelson. *Flight stability and automatic control*. McGraw Hill, Boston, Mass., 2 edition, 1998. 57

[87] James O. Nichols. Analysis and compilation of missile aerodynamic data. Technical report, Auburn University, Auburn Ala. 36830 prepared for NASA, 1977. NASA. 65

[88] G. Nitschke. Co-evolution of cooperation in a pursuit evasion game. In *Intelligent Robots and Systems, 2003. IEEE/RSJ International Conference on*, volume 2, pages 2037–2042, 2003. 29, 30

[89] Geoff Nitschke. Emergence of specialized behavior in a pursuit-evasion game. Lecture Notes in Computer Science, Volume 2691, Page 324, 2003. 21

[90] S. Y. Ong and B.L. Pierson. Optimal planar evasive aircraft maneuvers against proportional navigation missiles. *Journal of Guidance, Control, and Dynamics*, 19(6):1210–1215, 1996. 18

[91] Shaw Yang Ong. *Sequential Quadratic Programming Solutions to Related Aircraft Trajectory Optimization Problems*. PhD thesis, Iowa State University, 1992. 23

[92] Yaakov Oshman and David Arad. Enhanced air-to-air missile tracking using target orientation observations. *J. Guidance, Control and Dyncs.*, 27(4):0, 7 2004. 11

[93] H. Van Dyke Parunak, Ted Belding, Sven Brueckner, and John Sauter. *Emergent Behaviour in Planning as in Advanced Campaign Planning by Alexander Kott and Steven Morse*. to be published, 2007. 8, 9

[94] Michal Pechoucek, Simon G. Thompson, and Holger Voos. *AAMAS Technology for Military and Security Applications*. Springer, New York, 2007. xi, 8, 9, 27

[95] Anawat Pongpunwattana and Rolf Rysdyk. Evolution-based dynamic path planning for autonomous vehicle. *Studies in Computational Intelligence*, pages 113–145, 2007. 19

[96] Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000. 31

[97] D. B. Price, A. J. Calise, and D. D. Moerder. Piloted simulation of an onboard trajectory optimization algorithm. *Journal of Guidance, Control, and Dynamics*, 1984. 19

[98] Tuomas Raivio. Computational methods for dynamic optimization and pursuit-evasoin games. Technical report, Systems Analysis Laboratory Research Report, Helsinki University of Technology, Finland, 2000. 37, 85

[99] Tuomas Raivio. Capture set computation of an optimally guided missile. *J. Guidance, Control and Dyncs.*, 24-J(6-Ja):0, 11 2001. 31

[100] S. Rao Sathyanarayan, H.K. Birru, and K. Chellapilla. Evolving nonlinear time-series models using evolutionary programming. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, 1999. 34, 35

[101] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1981. 53

[102] A. Segal and T. Miloh. Novel three-dimensional differential game and capture criteria for a bank-to-turn missile. *Journal of Guidance, Control and Dynamics*, 17(5):1068, 9 1994. 11

[103] L. R. Shaw. *Fighter Combat: Tactics and Maneuvering*. Naval Institute Press, 1985. xi, 13, 15, 16, 18, 40, 91, 144, 150

[104] I. Shevchenko. Minimizing the distance to one evader while chasing another. *Computers & Mathematics with Applications*, 47(12):1827–1855, 2004. 11

[105] T. Shima. Capture conditions in a pursuit-evasion game between players with biproper dynamics1. *Journal of Optimization Theory and Applications*, 126(3):503–528, Sep 2005. 21

[106] J. Shinar. On the feasibility of —hit-to-kill— in the interception of maneuvering targets. In *American Control Conference, 2001. Proceedings of the 2001*, volume 5, pages 3358–3363, 6 2001. 21, 22, 23

[107] J Shinar, M. Guelman, and A. Green. Game optimal guidance law synthesis for short range missiles. *Journal of Guidance, Control, and Dynamics*, 15(1):191–197, 1992. 11, 22

[108] J. Shinar and S. Gutman. Three-dimensional optimal pursuit and evasion with bounded controls. *Automatic Control, IEEE Transactions on*, 25(3):492–496, 6 1980. 26

[109] J. Shinar, G. Silberman, and A. Green. On optimal missile avoidance - a comparision between optimal control and differential game solutions. In *ICCON'89 Control and Application*, 1989. 11

[110] J. Shinar and V. Turetsky. Applications of pursuit-evasion games theory for missile guidance, 2005. 11, 21

[111] U.S. Shukla and P.R. Mahapatra. The proportional navigation dilemma-pure or true? *Aerospace and Electronic Systems, IEEE Transactions on*, 26(2):382–392, March 1990. 73

[112] G.M. Siouris. *Missile Guidance and Control System*. Springer:New York, 2003. xi, 14, 72

[113] R. E. Smith, B. A. Dike, B. Ravichandran, A. El-Fallah, and R. K. Mehra. *Creative Evolutionary Systems*, chapter 19, pages 467–485. The Morgan Kaufmann Series in Artificial Intelligence. The Morgan Kaufmann Publisher, 2002. 28

[114] Robert E. Smith, B. A. Dike, R. K. Mehra, B. Ravichandran, and A. El-Fallah. Classifier Systems in Combat: Two-sided Learning of Maneuvers for Advanced Fighter Aircraft. In *Computer Methods in Applied Mechanics and Engineering*. Elsevier, 1999. 28

[115] Robert E. Smith, B. A. Dike, B. Ravichandran, A. El-Fallah, and R. K. Mehra. The fighter aircraft LCS: A case of different LCS goals and techniques. *Lecture Notes in Computer Science*, 1813:283–290, 2000. 28

[116] Marc Snir, Steve Otto, Steve Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, London, England, 1996. 78

[117] A. Speranzon and K.H. Johansson. Distributed pursuitevasion game: Evaluation of some communications scheme. In *IEEE Conference on Decision and Contro*, 2003. 21

[118] Brian L. Stevens and Frank L. Lewis. Aircraft control and simulation. *John Wiley and Sons, Inc*, 2nd:New Jersey, 0 2003. xi, xiv, 56, 57, 58, 65, 66, 67, 68, 86, 166, 169

[119] Tsung-Ying Sun, Shang-Jeng Tsai, Yan-Nian Lee, Shan-Ming Yang, and Shih-Hsiang Ting. The study on intelligent advanced fighter air combat decision support system. In *IEEE International Conference on Information Reuse and Integration*, pages 39–44, Sept. 2006. 20

[120] George Paul Sutton. *Rocket propulsion elements : an introduction to the engineering of rockets*. John Wiley & Sons, New York, 7 edition, 2000. 63

[121] R. Tuomas and E. Harri. Visual aircraft identification as a pursuitevasion game. *Journal of Guidance, Control, and Dynamics*, 23(4), 2000. 21

[122] Vladimir Turetsky and Josef Shinar. Missile guidance laws based on pursuitevasion game formulations. *Automatica*, 39(4):607, 4 2003. 21, 22, 23

[123] R. Vaidyanathan, C. Hocaoglu, T.S. Prince, and R.D. Quinn. Evolutionary path planning for autonomous air vehicles using multi-resolution path representation. In *Intelligent Robots and Systems, Proceedings IEEE/RSJ International Conference on*, volume 1, pages 69 – 76, 29Oct. 19

[124] S. Vathsal and M.N. Rao. Analysis of generalized guidance laws for homing missiles. *Aerospace and Electronic Systems, IEEE Transactions on*, 31(2):514–521, April 1995. 11

[125] X. N. Vinh. *Flight mechanics of high-performance aircraft*. Cambridge ; New York : Cambridge University Press, 1993., 1993. xi, 62, 63, 64

[126] Kai Virtanen. *Optimal Pilot Decisions And Flight Trajectories In Air Combat*. PhD thesis, Systems Analysis Lab, Dept. of Eng. Physics and Maths, Helsinki Univ. Tech., 2005. 53

[127] Kai Virtanen, Tuomas Raivio, and Raimo P. Hamalainen. Decision theoretical approach to pilot simulation. *Journal of Aircraft*, 36(4):632–641, 1999. 21

[128] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton: Princeton University Press, 1944. 1, 9

[129] Paul Walker. A chronology of game theory. http://www.econ.canterbury.ac.nz/personal_pages/paul_walker/gt/hist.htm, October 2005. 1

[130] D. Whitely, S. Rana, and R. Hechendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47, 1999. 48

[131] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, pages 67–82, April 1997. 28

[132] Paul Zarchan. Representation of realistic evasive maneuvers by the use of shaping filters. *Journal of Guidance, Control and Dynamics*, 2(4):290–295, 1979. 18

[133] Peter H. Zipfel. *Modeling and simulation of aerospace vehicle dynamics*. AIAA education series, 2000. 57, 65, 67

[134] D. Zu, J.D. Han, and M. Campbell. Artificial potential guided evolutionary path plan for multi-vehicle multi-target pursuit. In *Robotics and Biomimetics, ROBIO IEEE International Conference on*, pages 855 – 861, 22-26. 19

A Appendices

A.1 Jet Fighter Aircraft Configuration

A.1.1 Mass and Dimensional Characteristics Used in Simulation

Table A.1:

Weight, N (lb)	73 480 (16 519)
Moments of inertia, kg-m^2 (slug-ft^2)	
I_X	12 662 (9 339)
I_Y	53 147 (39 199)
I_Z	63 035 (46 492)
I_{XZ}	179 (132)
Wing dimensions:	
Span, m (ft)	8.84 (29.0)
Area, m^2 (ft^2)	26.0 (280.0)
Mean Aerodynamic Chort, m (ft)	3.335 (10.94)
Surface deflection limits:	
Horizontal tail-	
Symmetric (δ_h), deg	± 25.0
Differential (δ_D), deg	± 5.0 per surface
Aileron (flaperons), deg	± 21.5
Rudder, deg	± 30.0

A.1.2 6DOF Model

The following is the subroutine that calculates the state equations [118]:

```
subroutine f(time,x,xd,ail,rdr,elv,thtl,ireturn)
real x(13),xd(13),d(9),mass
common /param/xcg
```

```
    common /output/an,alat,ax,qbar,amach,q,alpha,theta,vt
    parameter (axx=9496.0,ayy=55814.0, azz=63100.0, axz=982.0)
    parameter (axzs=axz**2, xpq=axz*(axx-ayy+azz),gam=axx*azz-axz**2)
    parameter (xqr=azz*(azz-ayy)+axzs,zpq=(axx-ayy)*axx+axzs)
    parameter (ypr=azz-axx)
    parameter (weight=25000.0,gd=32.17, mass = weight/gd)
    data s,b,cbar,xcgr,hx/300.0,30.0,11.32,0.35,160.0/
    rtod=180.0/asin(1.0)/2.0

! ################################
! Assign state & control variables !
! ################################

    vt=x(1)
    alpha=x(2)*rtod
    beta=x(3)*rtod
    phi=x(4)
    theta=x(5)
    psi=x(6)
    p=x(7)
    q=x(8)
    r=x(9)
    alt=x(12)
    pow=x(13)

! ! Air data computer and engine model !

    call adc(vt,alt,amach,qbar)
    cpow=tgear(thtl)
    xd(13)=pdot(pow,cpow)
    t=engine_thrust(pow,alt,amach)

! ! Lookup tables and component buildup ! !

cxt = cx(alpha,elv)
cyt = cy(beta,ail,rdr)
czt = cz(alpha,beta,elv)
dail = ail/20.0
drdr = rdr/30.0
clt = cl(alpha,beta) + dlda(alpha,beta)*dail +dldr(alpha,beta)*drdr
cmt = cm(alpha,elv)
cnt = cn(alpha,beta) + dnda(alpha,beta)*dail + dldr(alpha,beta)*drdr
```

```
! ! Add damping derivatives !

tvt = 0.5/vt b2v = b*tvt
cq = cbar*q*tvt call damp(alpha,d)
cxt = cxt + cq *d(1)
cyt = cyt + b2v * ( d(2)*r + d(3)*p)
czt = czt + cq * d(4) clt = clt + b2v* ( d(5)*r + d(6)*p)
cmt = cmt + cq * d(7) + czt * (xcgr-xcg)
cnt = cnt +b2v*(d(8)*r + d(9)*p) - cyt*(xcgr-xcg) * cbar/b

! ! Get ready for state equations !

cbta = cos(x(3))
u = vt * cos(x(2))*cbta
v = vt * sin(x(3))
w = vt * sin(x(2))*cbta

sth = sin(theta); cth = cos(theta); sph = sin(phi); cph = cos(phi); spsi = sin(psi)
cpsi = cos(psi); qs = qbar * s; qsb = qs * b; rmqs = qs/mass; gcth = gd * cth
qsph = q * sph; ay = rmqs * cyt; az = rmqs * czt

! ! Force equations !

udot = r*v - q*w - gd * sth + ( qs * cxt + t ) / mass
vdot = p*w - r*u + gcth * sph + ay
wdot = q*u - p*v + gcth * cph + az dum = (u*u + w*w)

xd(1) = (u*udot + v*vdot + w*wdot)/vt
xd(2) = (u*wdot - w*udot)/dum
xd(3) = (vt*vdot - v*xd(1)) * cbta / dum


! ! Kinematics !

xd(4) = p + (sth/cth)*(qpsh + r*cph)
xd(5) = q*cph - r*sph xd(6) = (qpsh + r*cph)/cth

! ! Moments !

roll = qsb*clt
pitch = qs*cbar*cmt
yaw = qsb*cnt
```

```
pq = p*q
qr = q*r
qhx = q*hx
xd(7) = ( xpq*pq - xqr*qr + azz*roll + axz*(yaw+qhx))/gam
xd(8) = ( ypr*p*r - axz*(p**2-r**2) + pitch - r*hx)/ayy
xd(9) = (zpq*pq - xpq*qr + axz*roll + axx*(yaw+qhx))/gam

! ! Navigation !

t1 = sph *cpsi; t2 = cph*sth; t3 = sph*spsi
s1 = cth*cpsi; s2 = cth*spsi; s3 = t1*sth - cph*spsi; s4 = t3*sth+cph*cpsi
s5 = sph*cth; s6 = t2*cpsi + t3; s7 = t2*spsi-t1; s8 = cph*cth

! ! !

xd(10) = u*s1+v*s3+w*s6  ! north speed
xd(11) = u*s2+v*s4+w*s7  ! eash speed
xd(12) = u*sth-v*s5-w*s8 ! vertical speed

! ! Outputs !

an = -az/gd
alat = ay/gd

return
end
```

The aerodynamic and stability derivatives are given below [118]:

```
subroutine damp(alpha,d)
real d(9), a(-2:9,9)
data a/-.267,-.110,.308,1.34,2.08,2.91,2.76,2.05,1.50,1.49,1.83,
1.21,.882,.852,.876,.958,.962,.974,.819,.483,.590,1.21,-.493,
-1.04,-.108,-.108,-.188,.110,.258,.226,.344,.362,.611,.529,
.298,-2.27,-8.80,-25.8,-28.9,-31.4,-31.2,-30.7,-27.7,-28.2,
-29.0,-29.8,-38.3,-35.3,-.126,-.026,.063,.133,.208,.230,.319,
.437,.680,.100,.447,-.330,-.360,-.359,-.443,-.420,-.383,-.375,
-.329,-.294,-.230,-.210,-.120,-.100,-7.21,-.540,-5.23,-5.26,
-6.11,-6.64,-5.69,-6.00,-6.20,-6.40,-6.60,-6.00,-.380,-.363,
-.378,-.386,-.370,-.453,-.550,-.582,-.595,-.637,-1.02,-.840,
.061,.052,.052,-.012,-.013,-.024,.050,.150,.130,.158,.240,.150/
```

```
      s = 0.2*alpha
      k = int(s)
      if(k .le. -2) k = -1
      if(k .ge. 9) k = 8
      da = s - float(k)
      l = k + int( sign(1.1, da))
      do i = 1,9
      d(i) = a(k,i) + abs(da) * (a(l,i) - a(k,i))
      end do

      return
      end



!c  x axis aerodynamic force coefficient
      real function cx(alpha, elv)
      real alpha, elv, a(-2:9,-2:2), s, da, de, t, u, v, w

      data a/-.099,-.081,-.081,-.063,-.025,.044,.097,.113,.145,
      .167,.174,.166,-.048,-.038,-.040,-.021,.016,.083,.127,.137,
      .162,.177,.179,.167,-.022,-.020,-.021,-.004,.032,.094,.128,
      .130,.154,.161,.155,.138,-.040,-.038,-.039,-.025,.006,.062,
      .087,.085,.100,.110,.104,.091,-.083,-.073,-.076,-.072,-.046,
      .012,.024,.025,.043,.053,.047,.040/

      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = elv / 12.0
      m = int(s)
      if (m .le. -2) m = -1
      if (m .ge. 2) m = 1
      de = s - float(m)
      n = m + int(sign(1.1, de))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = u + abs(da) * (a(l, n) - u)
      cx = v + (w - v) * abs(de)
      return
      end

      real function cy(beta, ail, rdr)
!c  sideforce coefficient
      real  beta, ail, rdr
      cy = -0.02*beta + 0.021*(ail / 20.0) + 0.086*(rdr / 30.0)
      return
      end

      real function cz(alpha, beta, elv)
!c  z axis force coefficient
      real  alpha, beta, elv, a(-2:9), s, da
      integer k, l
      data  a/0.770,0.241,-0.100,-0.416,-0.731,-1.053,-1.366,
            -1.646,-1.917,-2.120,-2.248,-2.229/
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = a(k) + abs(da) * (a(l) - a(k))
      cz = s*(1.0 - (beta / 57.3)**2) - 0.19*(elv / 25.0)
      return
      end



      real function cm(alpha, elv)
!c  pitching moment coefficient
      real  alpha, elv, a(-2:9, -2:2), s, da, de, t, u, v, w
      integer k, l, m, n
      data  a/.205,.168,.186,.196,.213,.251,.245,.238,.252,.231,
      .198,.192,.081,.077,.107,.110,.110,.141,.127,.119,.133,
      .108,.081,.093,-.046,-.020,-.009,-.005,-.006,.010,.006,
      -.001,.014,.000,-.013,.032,-.174,-.145,-.121,-.127,
      -.129,-.102,-.097,-.113,-.087,-.084,-.069,-.006,-.259,
      -.202,-.184,-.193,-.199,-.150,-.160,-.167,-.104,-.076,
      -.041,-.005/
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = elv / 12.0
      m = int(s)
      if (m .le. -2) m = -1
      if (m .ge. 2) m = 1
      de = s - float(m)
      n = m + int(sign(1.1, de))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = u + abs(da) * (a(l, n) - u)
      cm = v + (w - v) * abs(de)
      return
      end



      real function cl(alpha, beta)
!c  rolling moment coefficient
      real  alpha, beta, a(-2:9, 0:6), s, da, db, t, u, v, w, dum
      integer k, l, m, n
      data  a/12*0,-.001,-.004,-.008,-.012,-.016,-.019,-.020,-.020,
      -.015,-.008,-.013,-.015,-.003,-.009,-.017,-.024,-.030,-.034,
      -.040,-.037,-.016,-.002,-.010,-.019,-.001,-.010,-.020,-.030,
      -.039,-.044,-.050,-.049,-.023,-.006,-.014,-.027,.000,-.010,
      -.022,-.034,-.047,-.046,-.059,-.061,-.033,-.036,-.035,-.035,
      .007,-.010,-.023,-.034,-.049,-.046,-.068,-.071,-.060,-.058,
      -.062,-.059,.009,-.011,-.023,-.037,-.050,-.047,-.074,-.079,
      -.091,-.076,-.077,-.076/
!c
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.2 * abs(beta)
      m = int(s)
      if (m .eq. 0) m = 1
      if (m .ge. 6) m = 5
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = u + abs(da) * (a(l, n) - u)
      dum = v + (w - v) * abs(db)
      cl = dum * sign(1.0, beta)
      return
      end



      real function cn(alpha, beta)
!c  yawing moment coefficient
      real  alpha, beta, a(-2:9, 0:6), s, da, db, t, u, v, w, dum
      integer k, l, m, n
      data  a/12*0,.018,.019,.018,.019,.019,.018,.013,.007,.004,-.014,
      -.017,-.033,.038,.042,.042,.042,.043,.039,.030,.017,.004,-.035,
      -.047,-.057,.056,.057,.059,.058,.058,.053,.032,.012,.002,-.046,
      -.071,-.073,.064,.077,.076,.074,.073,.057,.029,.007,.012,-.034,
      -.065,-.041,.074,.086,.093,.089,.080,.062,.049,.022,.028,-.012,
      -.002,-.013,.079,.090,.106,.106,.096,.080,.068,.030,.064,.015,
      .011,-.001/
!c
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.2 * abs(beta)
      m = int(s)
      if (m .eq. 0) m = 1
      if (m .ge. 6) m = 5
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = u + abs(da) * (a(l, n) - u)
      dum = v + (w - v) * abs(db)
      cn = dum * sign(1.0, beta)
      return
      end



      real function dlda(alpha, beta)
!c  rolling moment due to ailerons
      real  alpha, beta, a(-2:9, -3:3), s, da, db, t, u, v, w
      integer k, l, m, n
      data  a/-.041,-.052,-.053,-.056,-.050,-.056,-.082,-.059,-.042,-.038,-.027,
      -.017,-.041,-.053,-.053,-.053,-.050,-.051,-.066,-.043,-.038,-.027,-.023,
      -.016,-.042,-.053,-.052,-.051,-.049,-.049,-.043,-.035,-.026,-.016,-.018,
      -.014,-.040,-.052,-.051,-.052,-.048,-.048,-.042,-.037,-.031,-.026,-.017,
      -.012,-.043,-.049,-.048,-.049,-.043,-.042,-.042,-.036,-.025,-.021,-.016,
      -.011,-.044,-.048,-.048,-.047,-.042,-.041,-.020,-.028,-.013,-.014,-.011,
      -.010,-.043,-.049,-.047,-.045,-.042,-.037,-.003,-.013,-.010,-.003,-.007,
      -.008/

      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.1 * beta
      m = int(s)
      if (m .eq. -3) m = -2
      if (m .ge. 3) m = 2
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = t + abs(da) * (a(l, n) - u)
      dlda = v + (w - v) * abs(db)
      return
      end



      real function dldr(alpha, beta)
!c  rolling moment due to rudder
      real  alpha, beta, a(-2:9, -3:3), s, da, db, t, u, v, w
      integer k, l, m, n

      data  a/.005,.017,.014,.010,-.005,.009,.019,.005,-.000,
      -.005,-.011,.008,.007,.016,.014,.014,.013,.009,.012,
      .005,.000,.004,.009,.007,.013,.013,.011,.012,.011,
      .009,.008,.005,-.002,.005,.003,.005,.018,.015,.015,
      .014,.014,.014,.014,.015,.013,.011,.006,.001,.015,
      .014,.013,.013,.012,.011,.011,.010,.008,.008,.007,
      .003,.021,.011,.010,.011,.010,.009,.008,.010,.006,
      .005,.000,.001,.023,.010,.011,.011,.011,.010,.008,
      .010,.006,.014,.020,.000/

      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.1 * beta
      m = int(s)
      if (m .le. -3) m = -2
      if (m .ge. 3) m = 2
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = t + abs(da) * (a(l, n) - u)
      dldr = v + (w - v) * abs(db)
      return
      end



      real function dnda(alpha, beta)
!c  yawing moment due to ailerons
      real  alpha, beta, a(-2:9, -3:3), s, da, db, t, u, v, w
      integer k, l, m, n
      data  a/.001,-.027,-.017,-.013,-.012,-.016,.001,.017,.011,.017,.008,
      .016,.002,-.014,-.016,-.016,-.014,-.019,-.021,.002,.012,.016,.015,
      .011,-.006,-.008,-.006,-.006,-.005,-.008,-.005,.007,.004,.007,.006,
      .006,-.011,-.011,-.010,-.009,-.008,-.006,.000,.004,.007,.010,.004,
      .010,-.015,-.015,-.014,-.012,-.011,-.008,-.002,.002,.006,.012,.011,
      .011,-.024,-.010,-.004,-.002,-.001,.003,.014,.006,-.001,.004,.004,
      .006,-.022,.002,-.003,-.005,-.003,-.001,-.009,-.009,-.001,.003,
      -.002,.001/
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.1 * beta
      m = int(s)
      if (m .le. -3) m = -2
      if (m .ge. 3) m = 2
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = t + abs(da) * (a(l, n) - u)
      dnda = v + (w - v) * abs(db)
      return
      end



      real function dndr(alpha, beta)
!c  yawing moment due to rudder
      real  alpha, beta, a(-2:9, -3:3), s, da, db, t, u, v, w
      integer k, l, m, n
      data  a/-.018,-.052,-.052,-.052,-.054,-.049,-.059,-.051,-.030,-.037,
      -.026,-.013,-.028,-.051,-.043,-.046,-.045,-.049,-.057,-.052,-.030,
      -.033,-.030,-.008,-.037,-.041,-.038,-.040,-.040,-.038,-.037,-.030,
      -.027,-.024,-.019,-.013,-.048,-.045,-.045,-.045,-.044,-.045,-.047,
      -.048,-.049,-.045,-.033,-.016,-.043,-.044,-.041,-.041,-.040,-.038,
      -.034,-.035,-.035,-.029,-.022,-.009,-.052,-.034,-.036,-.036,-.035,
      -.028,-.024,-.023,-.020,-.016,-.010,-.014,-.062,-.034,-.027,-.028,
      -.027,-.027,-.023,-.023,-.019,-.009,-.025,-.010/
      s = 0.2 * alpha
      k = int(s)
      if (k .le. -2) k = -1
      if (k .ge. 9) k = 8
      da = s - float(k)
      l = k + int(sign(1.1, da))
      s = 0.1 * beta
      m = int(s)
      if (m .eq. -3) m = -2
      if (m .ge. 3) m = 2
      db = s - float(m)
      n = m + int(sign(1.1, db))
      t = a(k, m)
      u = a(k, n)
      v = t + abs(da) * (a(l, m) - t)
      w = t + abs(da) * (a(l, n) - u)
      dndr = v + (w - v) * abs(db)
      return
      end
```

The atmosphere is calculated using:

```
!C
      subroutine adc(vt, alt, amach, qbar)
!     input - vt,alt !     output - amach,qbar
      data r0/2.377e-3/
      tfac = 1.0 - 0.703e-5*alt
      t = 519.0*tfac
      if (alt.ge.35000.0) t = 390.0
      rho = r0 * (tfac**4.14)
      amach = vt / sqrt(1.4*1716.3*t)
      qbar = 0.5*rho*vt*vt
      return
      end
```

The aircraft engine and thrust are:

```
    real function tgear (thtl)
    if ( thtl .le. 0.77 ) then
     tgear=64.94*thtl
    else
     tgear=(217.38*thtl)-117.38
    endif
    end function

    real function pdot(p3,p1) !cpdot = rate of change of power
    real p1 ! power command
    real p2
    real p3 ! actual power
```

```
real t
if (p1.ge.50.0) then
    if (p3.ge.50.0) then
        t = 5.0
        p2 = p1
    else
        p2 = 60.0
        t = rtau(p2 - p3)
    endif
else
    if (p3 .ge. 50.0) then
        t = 5.0
        p2 = 40.0
    else
        p2 = p1
        t = rtau(p2 - p3)
    endif
endif
pdot = t * (p2 - p3)
end function

real function rtau(dp)
real dp
if (dp .le. 25.0) then
    rtau = 1.0
else if (dp .ge. 50.0) then
    rtau = 0.1
else
    rtau = 1.9 - (0.036 * dp)
end if
return
end

real function engine_thrust(pow, alt, rmach) !cf16 engine thrust model
real pow, alt, rmach, a(0:5, 0:5), b(0:5, 0:5), c(0:5, 0:5)
real h, dh, rm, dm, cdh, s, t, tmil, tidl, tmax
integer i, m ! idle data
data a/4714.9,2980.2,3914.2,5070.7,6672.0,8273.3,2824.5,
1890.4,3069.1,4492.5,5915.8,7561.6,266.9,111.2,1534.6,
3358.2,5026.2,6783.2,-4537.0,-3158.1,-1334.4,1556.8,
4047.7,6049.3,-12009.6,-8451.2,-5782.4,-1098.7,2668.8,
4892.8,-16012.8,-6227.2,-2646.6,-1521.2,-889.6,3113.6/ ! mil data
```

```
data b/56400.6,40699.2,27577.6,17569.6,10897.6,6227.2,
56400.6,40699.2,28080.2,17969.9,10986.6,6227.2,56089.3,
41419.8,29401.3,19081.9,11564.8,6938.9,56222.7,43763.9,
31536.3,20727.7,12632.3,7383.7,55110.7,45262.8,34472.0,
23663.4,14456.0,8584.6,51952.6,43803.9,35806.4,27132.8, 16902.4,10274.9/ ! max data

data c/88960.0,66720.0,48038.4,31136.0,17792.0,11120.0,
95276.2,69833.6,49928.8,32572.7,19726.9,11564.8,100969.6,
74993.3,54488.0,36269.0,22240.0,12610.1,107819.5,84111.7,
61204.5,41299.7,25353.6,14300.3,115959.4,93741.6,71056.8,
49439.5,30513.3,17569.6,128484.9,103722.9,81398.4,59976.8, 38439.6,22493.5/ !
h = alt / 3048.0
i = int(h)
if (i .ge. 5) i = 4
if (i .lt. 0) i = 0
dh = h - float(i)
rm = 5.0 * rmach
m = int(rm)
if (m .ge. 5) m = 4
if (m .lt. 0) m = 0
dm = rm - float(m)
cdh = 1.0 - dh

s = b(i,m) * cdh + b(i+1,m) * dh
t = b(i,m+1) * cdh + b(i+1,m+1) * dh
tmil = s + (t - s) * dm

if (pow .lt. 50.0) then
  s = a(i,m) * cdh + a(i+1,m) * dh
  t = a(i,m+1) * cdh + a(i+1,m+1) * dh
  tidl = s + (t - s) * dm
  engine_thrust = tidl + (tmil - tidl) * pow * 0.02
else
  s = c(i,m) * cdh + c(i+1,m) * dh
  t = c(i,m+1) * cdh + c(i+1,m+1) * dh
  tmax = s + (t - s) * dm
  engine_thrust = tmil + (tmax - tmil) * (pow - 50.0) * 0.02
endif

return
end
```

The Runge-Kutta integration code:

```
      subroutine rk4(tt,dt,xx,xd,nx,ail,rdr,elv,thtl,ireturn)
      parameter (nn=30)
      real xx(*),xd(*),x(nn),xa(nn)
      call f(tt,xx,xd,ail,rdr,elv,thtl,ireturn)
      if ( ireturn==1005) return
      do 1 m=1,nx
      xa(m)=xd(m)*dt
1     x(m)=xx(m)+0.5*xa(m)
      t=tt+0.5*dt
      call f(t,x,xd,ail,rdr,elv,thtl,ireturn)
      if ( ireturn==1005) return
      do 2 m=1,nx
      q=xd(m)*dt
      x(m)=xx(m)+0.5*q
2     xa(m)=xa(m)+q+q
      call f(t,x,xd,ail,rdr,elv,thtl,ireturn)
      if ( ireturn==1005) return
      do 3 m=1,nx
      q=xd(m)*dt
      x(m)=xx(m)+q
3     xa(m)=xa(m)+q+q
      tt=tt+dt
      call f(tt,x,xd,ail,rdr,elv,thtl,ireturn)
      if ( ireturn==1005) return
      do 4 m=1,nx
4     xx(m)=xx(m)+(xa(m)+xd(m)*dt)/6.0
      return
      end
```

A.2 Generic Air-to-Air Missile

The configuration of a medium range air-to-air missile is given by the following [38]:

Mass (kg)	200 kg
Thrust (N)	6000 N
t_e	8s
$a_{c_{max}}$	30g
τ	3s
N_e	3-6