Optimal Novel Taxiing Navigation of a BOEING-747 Aircraft Using Artificial Intelligence

Abrar Alhindi

Supervised by Prof Mahdi Mahfouf

Submitted to the University of Sheffield for the degree of Doctor of Philosophy in the Faculty of Engineering

Department of Automatic Control and Systems Engineering University of Sheffield

Mappin Street, Sheffield, S1 3JD, UK

August 2023

Acknowledgments

I would like to express my gratitude to everyone who supported me and helped me throughout my years of studies.

I would like first to express my sincere gratitude and appreciation to my supervisor, Prof Mahdi Mahfouf, for giving me the opportunity to work under his wise guidance and for his invaluable support and expert advice during my research and the writing up of this thesis.

Also, I am grateful to Dr. Olusayo Obajemu for all the helpful interactions and the team at QMUL, Stirling University, who are part of TRANSIT for help and assistance.

In addition, I would like to thank all ACSE research students for the constructive, interesting and useful discussions I have benefited from immensely.

Finally, I take this opportunity to thank my family for supporting and encouraging me continuously.

Abstract

Aircraft ground movement coordination plays a key role in improving airport efficiency, as it acts as a link to all other ground operations. Finding novel approaches to coordinate the movements of a fleet of aircraft at an airport in order to improve system resilience to disruptions with increasing autonomy is at the centre of many key studies for airport airside operations. Moreover, autonomous taxiing is envisioned as a key component in future digitalized airports. However, state-of-the-art routing and scheduling algorithms for airport ground movements do not consider high-fidelity aircraft models at both the proactive and reactive planning phases.

The majority of such algorithms do not actively seek to optimize fuel efficiency and reduce harmful greenhouse gas emissions. This thesis proposes new approaches using Artificial Intelligence (AI) for optimal taxiing navigation of a high-fidelity aircraft model, working in conjunction with a routing and scheduling algorithm that determines the taxi route, waypoints, and time deadlines. The proposed approaches used in this thesis are: PID controller, artificial neural networks controller, Fuzzy Inference System (FIS) model and an online controller using reinforcement learning.

The proposed approaches integrate a MATLAB-Simulink model of the BOEING-747 aircraft with artificial intelligence based control that successfully generate fuel-efficient four-dimensional trajectories 4DTs in real time, while taking constraints on operations into account. The proposed methodologies are realistic and simple to implement. Moreover, simulation studies show that the proposed approaches are capable of providing a reduction in the fuel consumed during the taxiing of a large Boeing 747-100 jumbo jet.

Table of Contents

1 INTRODUCTION 14

1.1 A Review of the Research Area 14

1.2 Issues and Shortcoming in the State of the Art 29

1.3 Aims and Objectives 29

1.4 Research Methodology 30

1.5 Thesis Overview 31

2 AIRCRAFT MODEL DEVELOPMENT 33

2.1 Introduction 33

2.2 The Boeing 747 Aircraft Model 35

2.3 Axis Systems 36

2.4 State Equations 37

2.4.1 Rotational acceleration function ($f1$) 38

2.4.2 Translational acceleration function ($f2$) 38

2.4.3 Attitude rates 39

2.4.4 Earth relative velocity 39

2.4.5 Pre-thrust rate 40

2.5 Output Equations 40

2.6 Force and Moment Derivations 43

2.6.1 Aerodynamic Forces and Moments 43

2.6.2 Engine Forces and Moments 45

2.6.3 Undercarriage Forces and Moments 52

2.6.4 Gravity Model 58

2.7 Simulation Plots 58

2.8 Summary 83

3 PID CONTROLLER FOR AIRCRAFT TAXIING 84

3.1 Introduction 84

3.2 PID Controller Design Process 87

3.2.1 Aircraft navigation optimization 87

3.3 The Limitation of PID Controller 95

3.4 The Clustering Solution 95

3.5 Results of Experiments 97

3.5.1 Taxiing run along a circle path *97*
3.5.2 Taxiing run along a rectangular path *98*
3.6 SUMMARY 99
4 ARTIFICIAL NEURAL NETWORKS FOR AIRCRAFT TAXIING 101
4.1 INTRODUCTION 101
4.2 NEURAL NETWORKS EQUATIONS 103
4.3 NEURAL NETWORK DESIGN PROCESS 105
4.3.1 Data collection *105*
4.3.2 Build the network *105*
4.3.3 Initialize the weights and biases *106*
4.3.4 Train the network *106*
4.3.5 Use the network *106*
4.4 RESULTS OF EXPERIMENTS 108
4.4.1 Taxiing run along a circle path *108*
4.4.2 Taxiing run along a rectangular path *109*
4.5 SUMMARY 109
5 ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM FOR AIRCRAFT TAXIING 111
5.1 INTRODUCTION 111
5.2 WHAT IS ANFIS? 112
5.3 FUZZY INFERENCE SYSTEM STRUCTURE AND PARAMETER ADJUSTMENT 112
5.4 KNOW THE DATA 112
5.5 FUZZY INFERENCE SYSTEM DESIGN PROCESS 113
5.5.1 Data Collection *113*
5.5.2 Designing the fuzzy inference system *113*
5.6 SUMMARY 123
6 REINFORCEMENT LEARNING FOR AIRCRAFT TAXIING 124
6.1 INTRODUCTION 124
6.2 REINFORCEMENT LEARNING AND TRADITIONAL CONTROLS 126
6.3 THE WORKFLOW OF REINFORCEMENT LEARNING 127
6.3.1 Formulate Problem *127*
6.3.2 Create the Environment Interface *128*
6.3.3 Define Reward *129*
6.3.4 Create the Critic *131*

6.3.5 Create the Actor *134*
6.3.6 Create the DDPG Agents *141*
6.3.7 Train the Agents (online training) *142*
6.3.8 Validate Trained Agents *144*
6.3.9 Test Trained Agents *147*
6.4 SUMMARY 150
7 CONCLUSIONS AND FUTURE WORK 151
7.1 CONCLUSIONS 151
7.2 FUTURE WORK 156
REFERENCES 161

Publications

Abrar Alhindi, Mahdi Mahfouf, Olusayo Obajemu and Jun Chen, **An Artificial Neural Networks (ANN) Based System for Optimal Taxiing Navigation of a BOEING-747 Aircraft**, 2022 21st UK Workshop on Computational Intelligence (UKCI). Springer, 2022 (in Press).

Olusayo Obajemu, Mahdi Mahfouf, Lohithaksha M. Maiyar, **Abrar Alhindi**, Michal Weiszer and Jun Chen, **Real-Time Four-Dimensional Trajectory Generation Based on Gain-Scheduling Control and a High-Fidelity Aircraft Model,** Engineering, 2021. 7(4): pp.495-506.

List of Abbreviations and Symbols

4DTs	Four-Dimensional Trajectories
AAS	Aircraft Arrival Scheduling
ACS	Ant Colony System
AI	Artificial Intelligence
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
CPS	Constrained of Position Shifting
DDPG	Deep Deterministic Policy Gradient
EPR	Engine Pressure Ratio
FCFS	First Come First Serve
FIS	Fuzzy Inference System
GA	Genetic Algorithm
GRP	Ground Routing Problem
GSA	Ground Slope Angle
ICAO	International Civil Aviation Organization
IE	Implicit Enumeration
PLA	Power Lever Angle
PID	Proportional-Integral-Derivative controller
PV	Process Variable
RHC	Receding Horizon Control
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RSP	Runway Sequencing Problem
SP	Desired Setpoint
TMA	Traffic Management Advisor
F_{s_i}	Side force

F_{μ_i}	Tire drag force
$\mathscr{s}_{F.R.L}$	Horizontal stabilizer angle relative to the fuselage reference line (degrees)
a_x	Acceleration along the "x" axis ft/sec^2
a_y	Acceleration along the "y" axis ft/sec^2
a_z	Acceleration along the "z" axis ft/sec^2
$C_{D,basic}$	Basic drag coefficient for the rigid airplane
C_D	Airplane drag coefficient
$C_{L,basic}$	Basic lift coefficient for the rigid airplane
C_L	Airplane lift coefficient
C_l	Airplane rolling moment coefficient
$C_{M,basic}$	Basic pitching moment for the rigid airplane
C_M	Airplane pitching moment
C_n	Airplane yawing moment coefficient
$C_{Y,basic}$	Basic side force coefficient for the rigid airplane
C_Y	Airplane side force coefficient
$\bar{c}$	Wing mean aerodynamic chord, m
$\frac{dC_D}{dM}M$	Change in drag coefficient due to Mach number
$\frac{dC_L}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V}$	Change in lift coefficient due to rate of change of angle of attack
$\frac{dC_L}{dn}n$	Change in lift coefficient due to aeroelastic inertia relief caused by normal load factor
$\frac{dC_l}{dp}\frac{pb}{2V}$	Rolling moment coefficient due to roll rate about the stability axis
$\frac{dC_L}{dq}\frac{q\bar{c}}{2V}$	Change in lift coefficient due to pitch rate
$\frac{dC_l}{dr}\frac{rb}{2V}$	Rolling moment coefficient due to yaw rate about the stability axis

$\frac{dC_L}{d\alpha}\alpha$	Change in basic lift coefficient due to the aeroelastic effect on the rigid airplane basic lift coefficient curve slope
$\frac{dC_l}{d\beta}\beta$	Rolling moment coefficient due to angle of sideslip
$\frac{dC_M}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V}$	Change in pitching moment coefficient due to rate of change of angle of attack
$\frac{dC_M}{dn}n$	Change in pitching moment coefficient due to aeroelastic inertia relief caused by normal load factor
$\frac{dC_M}{dq}\frac{q\bar{c}}{2V}$	Change in pitching moment coefficient due to pitch rate
$\frac{dC_n}{d\dot{\beta}}\frac{\dot{\beta}b}{2V}$	Yawing moment coefficient due to rate of change of sideslip angle
$\frac{dC_n}{dp}\frac{qs}{2V}$	Yawing moment coefficient due to roll rate about the stability axis
$\frac{dC_n}{d\beta}\beta$	Yawing moment coefficient due to angle of sideslip
$\frac{dC_Y}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V}$	Change in side force coefficient due to rate of change of angle of attack
$\frac{dC_Y}{dn}n$	Change in side force coefficient due to aeroelastic inertia relief caused by normal load factor
$\frac{dC_Y}{dq}\frac{q\bar{c}}{2V}$	Change in side force coefficient due to pitch rate
$\frac{dC_Y}{d\beta}\beta$	Side force coefficient due to angle of sideslip
$\frac{d()}{dt} = (\dot{})$, $\frac{d^2()}{dt^2} = (\ddot{})$	Time derivative operation
F_{G_i}	Tire vertical oleo strut force
F_z	Vertical force

K_B	Braking constant
K_{BM}	Maximum value of braking constant
V_G	Aircraft ground speed, knots
$w_{ij}(t+1)$	Neural network hidden weights
$w_{ki}(t+1)$	Neural network output weights
Y_k	Neural network output
$\dot{\alpha}$	Rate of change of angle of attack
$\dot{\beta}$	Rate of change of angle of sideslip
δ_{t_i}	Tire deflection
$\delta_i^h(t)$	Local gradients of the neural network hidden neurons
$\delta_k(t)$	Local gradient of the neural network output
δ_S	Nose wheel steering angle, degrees
δ_{SP}	Spoiler deflection angle, degrees
ΔS_{T_i}	Landing gear oleo strut compression, in
$\Delta \dot{S}_{T_i}$	Landing gear oleo strut compression rates
θ_B	Body pitch angle
$\dot{\theta}$	Pitch rate
μ_B	Coefficient of rolling friction
ϕ_B	Body bank angle
$\dot{\phi}$	Roll rate
$\dot{\psi}$	Yaw rate
$\Delta C_{D,\ Landing\ gear}$	Change in drag coefficient due to landing gear extension
$\Delta C_{D,\ rudders}$	Change in drag coefficient due to rudder deflection
$\Delta C_{L,\ Landing\ gear}$	Change in lift coefficient due to landing gear extension

$\Delta C_{l,\ rudders}$	Rolling moment coefficient due to rudder deflection
$\Delta C_{M,\ Landing\ gear}$	Change in pitching moment coefficient due to landing gear extension
$\Delta C_{M,\ rudders}$	Change in pitching moment coefficient due to rudder deflection
$\Delta C_M \frac{dC_M}{d\alpha}\alpha$	Change in pitching moment coefficient due to the aeroelastic effect on the rigid airplane basic pitching moment coefficient curve slope
$\Delta C_{n,\ rudders}$	Yawing moment coefficient due to rudder deflection
$\Delta C_{Y,\ Landing\ gear}$	Change in side force coefficient due to landing gear extension
b	Wing span, m
bof	Break out force
c.g.	Airplane centre of gravity position as a fraction of the wing mean aerodynamic chord
CO	Carbon monoxide
e(t)	Error value
$F.R.L$	Fuselage Reference Line (anybody water line)
h	Pressure altitude of the airplane c.g., ft
I	Moments of inertia
J	Product of inertia
M	Mach number
m	Mass of the aircraft
p	Roll rate
q	Pitch rate

r	Yaw rate
S	Wing area, m^2
u, v, w	Body fixed linear velocity
V	Velocity vector
α	Angle of attack relative to fuselage reference line.
β	Angle of sideslip
δA_D	Outboard aileron deflection angle, degrees
δA_I	Inboard aileron deflection angle, degrees
δE_D	Outboard elevator deflection angle, degrees
δE_I	Inboard elevator deflection angle, degrees
δR_L	Lower rudder deflection angle, degrees
δR_U	Upper rudder deflection angle, degrees
θ	Pitch angle
ϕ	Roll angle
$\varphi(v)$	Sigmoid activation function
ψ	Yaw angle

1 Introduction

1.1 A Review of the Research Area

Congestion is considered to be one of the most problematic issues in international airspace. In the twenty-first century, European airports experience many challenges, including those relating to capacity and the environment. Airports and their transportation systems can avoid the occurrence of huge bottleneck problems via appropriate expansion plans or efficient utilization of existing resources. The investment in infrastructure can help in expanding the airport surface area, leading increased capacity of the airport. However, this is expensive in most cases and can lead to harmful effects on the environment, which may lead to noise and pollution as well as increasing the overall complexity of operations which thus add to more workload.

Airports work almost closer to their maximum capacity. The continued increase in airport surface area can be difficult and costly; it is argued here that research should focus on finding solutions that use the existing space in more efficient ways rather. Decision support systems, planning and scheduling have to be increasingly and continuously improved and advanced. Improving the efficiency of the airports is considered to be one of the most important issues of aircraft ground management, because it is a link to all other ground operations which involve the coordination of machines and humans.

The attempt to taxi an aircraft in an optimal and efficient manner using automatic systems is increasingly deployed across major airports across the world. However, many of these systems do not use adequate aircraft models, nor do they continuously seek to optimize objective functions such as minimizing fuel consumption and minimizing harmful greenhouse gas emissions.

Recent studies have shown that for efficient operations and to deal with the increase of traffic of passenger and aircraft at airports, the whole guidance of each flight from gate to gate is required [1], [2]. This whole guidance is called the four Dimensional Trajectory (4DT) which

consists of 3 coordinates of position and 1 for time [3]. The 4DT is the aircraft navigation from departure to arrival gates (pushback and taxiing paths are included) [4].

The 4DT guidance system not only provides opportunities for optimizing the various stages of ground movement, but also coordinates the movements of a fleet of aircraft. Indeed, the 4DT guidance system has the potential to reduce taxi delays by as much as 55% [5]. It is therefore no surprise that many new investigations of new technologies are currently conducted by air traffic managers in order to deal with the problem of airports congestion [1].

Furthermore, aircraft emissions, which account for 3% of global greenhouse emissions [6], [7], provide a significant motivation for ground movement optimization. In 2002 in the London Heathrow airport alone, it was calculated that 56% of the overall NOx was produced from aircraft ground traffic [8]. It is cited in [9] that fuel consumed for taxi and ground operations of the entire mission in European flights was about 5% to 10%. These provide the significant motivations for optimisation of the ground movement. With 1.5 times increasing numbers of passengers by 2035 [1], [7] and to manage the challenge of aircraft traffic task at many airports the need to implement the 4DT algorithms has become more urgent and important. Investigating the feasibility of scheduling systems and airport ground navigation [10]–[11] has been carried out via a plethora of studies.

An aircraft accurate model is important for accurate guidance, as this model will be used by the scheduling and optimisation algorithms that provide the 4DT guidance. Some external constraints would be used by these algorithms such as an aircraft performs a manoeuvre by using the least amount of emissions and fuel in a specific time deadline.

In particular, the most important objectives are fuel consumption and CO2 emissions because of the increase in congestion and traffic are as these will directly affect the consumption of aircraft's fuel and emissions, which will therefore cause increase the running costs and will significantly contribute to greenhouse gas emissions and environmental damage. Thus, decisions via optimisation/scheduling algorithms are crucial in order to meet the aircraft time constraints. These optimisation/scheduling algorithms will contribute in eliminating the airport

traffic and congestion which will in turn reduce the delay time and therefore result in minimising the fuel consumption and emissions.

These studies are broadly classified into two categories. The first category relates to algorithms which are concerned with scheduling the airport ground operations. These are more concerned with airport-wide operations rather than being related to a specific aircraft. For example, see [7] and [12], where an active routing schedule algorithm was developed to this effect.

Ravizza et al. [13], have developed a new model to examine the trade-off between two different conflict objectives, which are fuel consumption during taxiing and taxi time. The model combines two sophisticated algorithms, which are developing a simulation platform for different datasets, while the other is to do with the analysis work. The objective functions consist of the taxi time and the aircraft's engine force function during taxiing. Historic data that were used for the analysis were taken from the Zurich Airport. The results show that the integrated algorithms proved their ability to tackle the trade-off problem in an effective way. The study needs more research in understanding the fuel usage and practices of pilots during taxiing.

Guépet et al. [14], have focused on integrating the Runway Sequencing Problem (RSP) and Ground Routing Problem (GRP), that aimed to simultaneously reduce taxi times and increase runway efficiency. This integration improved the synchronization of take-offs and ground movements. An IP formulation based heuristic sequential algorithm was proposed for the RSP. Real data was used for testing the proposed method that was taken from Copenhagen Airport (CPH). The results confirm that this algorithm is able to reduce the taxi time significantly.

Brownlee et al. [15], have proposed a model of a permutation-encoded genetic algorithm with a rolling window approach to keep aircrafts apart in a safe distance and find aircraft's shortest path. In this approach, allocating routes depend on the order of choosing aircraft. When applied to Beijing, Hong Kong and Doha and Capital International Airports, the proposed method was able to efficiently find better routings than First Come First Serve method or extensive search over small rolling windows.

Brownlee et al. [16], have estimated taxi times and their related uncertainties by using a proposed approach which is an adaptive Mamdani fuzzy rule based system. Moreover, the fuzzy taxi time estimates were used with the existing Quickest Path Problem with Time Windows (QPPTW) algorithm. Taxi movements were simulated at Manchester Airport in the UK and the results from this experiments proved that the proposed approach was able to find robust routes and the delays were reduced by 10–20% over the original QPPTW approach.

Psaraftis [17], has developed three versions of dynamic programming algorithm for Aircraft Sequencing problem. The objective of these algorithms was how to land all of airplanes in the airport in a quick way in order to decrease the total waiting time of the passenger. The cases that were studied are (a) The single runway with no constraint (b) The single runway with Constraint of Position Shifting (CPS) (c) The two runway with no constraint. All the airplanes in their system were categorized into a small number of different types. Therefore, there was a huge savings in computational effort. Some ideas were suggested on how this work can be extended in the future.

Bayen et al. [18], have proposed a model that each aircraft is set in holding loops, a specified amount of time every loop will take to be completed before they go to the queue. All aircraft are assumed to belong to a single class. The objective is to the minimize the idle time made by traversing an integer number of loops. They consider this problem as scheduling problem of a single machine, where there is N of jobs and each job has a deadline and release time. A holding time and a processing time were also previously provided. The study aims to achieve two important objectives by creating efficient schedules which are minimizing both the makespan (the time at which all jobs are finished) and the sum of the starting times of all jobs. To achieve this, approximation algorithms are presented that is able to approximate two objectives with factors of 3 and 5. Their algorithms combine dynamic programming and the rounded solution of a linear program relaxation.

Lee and Balakrishnan [19], have studied how to minimize the sum of landing times of arriving aircraft, fuel cost and total delay by using a dynamic programming technique. The strategy of speeding up some aircraft at the expense of burning extra fuel was examined. They investigated

the trade-off between minimizing the average delay and maximizing the throughput by using 30-aircraft sequences and generating 1000 problem instances with a Poisson distribution. The study show that speeding up the aircrafts enable the them to land a up to 3 minutes earlier than usual. The results of the study show that the average delay significantly improved (up to 50%) through resequencing with the throughput decrease, thus that the delay will be small.

Lieder et al. [20], have contributed in presenting a new optimization algorithm using dynamic programming to create optimal landing schedules with general assumptions (limited time windows, positive target times and multiple runways). The algorithm aims to minimize the delay costs of all aircraft while simultaneously adhering to separation constraints, incorporating a cost function that considers factors such as exhaust emissions, fuel burn, maintenance, and missed flights. The study shows that the proposed algorithm is able to produce schedules notably faster within seconds instead of hours than standard MIP solvers such as Cplex with up to 100 aircraft.

Craig et al. [21], have used a dynamic programming approach for solving the problem of sequencing the aircraft departure at holding points at London Heathrow airport based on a flexible cost function. Some possible ways are considered that the algorithm could be extended to cope with a long sequence of departures.

Branch-and-bound is a known method for combinatorial optimization problems. One of the early work using this method was by Brinton [22] to optimize the sequences and schedules of the arrival aircraft on the runway. The scheduling algorithm, referred to as the Implicit Enumeration (IE), is employed in the Traffic Management Advisor (TMA). To decrease the required search of tree branches, various methods are employed in this study, including dynamic, static, and depth strategies. The developed IE algorithm works in a dynamic feedback environment. Combination of multiple weighted costs are included in the objective function, but the proposed (IE) algorithm does not depend on which costs are included. The algorithm also works efficiently in the situation at congested airports with several simultaneous landing runways. When considering both runway and sequence assignments, IE algorithm shows significant enhancements in the performance.

Beasley et al. [23], have proposed tree search approaches using linear programming used to develop branch-and-bound algorithms for tackling the problem of single and multiple landing runways. However, they propose extra constraints to strengthen the LP relaxation and reduce the formulation of zero-one space of the mixed integer. The objective function is assigning penalties for aircraft landing in case before and after target times. The aircraft landing problem is optimally tackled for the problem cases including four runways and fifty aircrafts.

Sölveling et al. [24], have solved the stochastic scheduling problem of the airport runway using stochastic branch and bound methodology to find optimal solutions and investigate the impact of the uncertainty level. Minimizing the total makespan of sequence of aircrafts on single or multiple runways was the main objective of this scheduling algorithm. The obtained sequence from the proposed algorithm has shorter makespans by 5–7% than sequence generated from deterministic sequencing models. Additionally, the proposed algorithm proves its ability to solve fourteen aircrafts instances in less than one minute. The trade-off between solution quality and runtime for different upper and lower bound models was not investigated in this study.

D'Ariano et al. [25], have developed two creative branch and bound algorithms and proved their ability to solve the scheduling problem in seconds with solutions better than other scheduling heuristics algorithms. More than half of the instances can be optimality scheduled in less than two minutes. Scheduling algorithms are implemented and evaluated in the Fiumicino airport. On-line decision support system for air traffic control at TMAs is currently in ongoing research plan.

Bianco et al. [26], have considered the scheduling jobs problem is a single machine scheduling problem with sequence dependent processing times and release dates in order to decrease the total completion time. They indicate that the problem of a Traveling Salesman with additional time constraints is equivalent to this problem. Therefore, a dynamic programming algorithm was developed with three lower bounds and also. Moreover, two heuristic algorithms are proposed. The first one is a construction procedure of O($n^2 logn$) that adding jobs to the current

partial sequence to create a schedule, whereas the second one is an insertion approach based on O(n4).

Performance analyses is conducted for both heuristics and lower bounds using random instances. Furthermore, the algorithms were tested and analysed on real aircrafts sequencing problem of 44 commercial aircrafts of four different classes in congested airports. Then they were evaluated based on the average and maximum landing delay and the total aircraft landing time. The total scheduling process takes about fourtee minutes to be completed. Computational results prove the effectiveness heuristic solutions in practice. Some aircrafts are noticed to have extreme delay as CPS was not included in their model.

Shi et al. [27], have proposed a CGIC novel heuristic approach to solve aircraft landing scheduling problems. The CGIC includes 4 parts: a chunk improvement heuristic, a landing subsequence generation rule, a chunking rule based on costs and a connection rule. By breaking down the aircraft landing scheduling problem into more than two subproblems and using fewer amounts of aircrafts, its complexity will be highly reduced. Results show that cost based chunking rule outperforms weight or time based chunking rules in static instances which contains many aircrafts. The proposed CGIC is able to optimally solve the landing scheduling problems with up to 500 aircrafts. CGIC can efficiently attain great solutions in dynamic instances case, and it is good in real-time execution because its low computation time.

Cheng et al. [28], have developed 4 different genetic formulations for many runways sequencing, assignment and scheduling of aircraft landing problem. The first formulation uses two separate chromosomes to encode the runway assignment and the sequence of flight. The second formulation uses only one chromosome to describe the priority list or overall sequence. The third formulation differs from the second one with adding a fitness-based probabilistic selection process. The fourth formulation differs from the first three with chromosomes defined as functions and mathematical operations to describe a metric, that is used as part of a sequencing, scheduling and runway assignment. They evaluate the four approaches using 12 aircraft and three runways.

Hu et al. [29], have proposed a solution for landing aircrafts using binary representation based GA. Each pair of aircraft has a neighbouring relationship which result in is a 0-1 valued matrix chromosome. The proposed algorithm is also integrated with the strategy of receding horizon control (RHC). This binary representation is considered an efficient uniform crossover operator, and usually not applicable to those permutation representations. The results from the study show that binary representation based GA outperforms the permutation based GA.

Randall [30], has applied the ACO algorithm for the aircraft landing problem. In this application the objective is the minimization of the total aircraft landing penalty. In addition, they use time-window constraints. The basic ACO algorithm has been modified with some novel modifications. The proposed algorithm generated competitive solutions and tested on 50 aircrafts.

Wu et al. [31], have proposed two-level Ant Colony System (ACS) approach based on Receding Horizon Control (RHC) in order to tackle the multi-runways Aircraft Arrival Scheduling (AAS) problem. By dividing the problem using RHC and reducing the number of aircrafts in each RHC stage, a less computational complexity will ACS algorithm have. To deal with the difficulty of multi run ways scheduling, a strategy of two level scheduling is suggested to assist the ACS. In first level scheduling, a single runway only was only used to schedule aircrafts. In second level scheduling, multi-runways used to schedule aircrafts which based on scheduling of first level and the a runway occupancy. The experiment results of the study was compared with the results of first come first serve (FCFS) approach. The proposed ACS-MRAAS-RHC approach shows its higher performance as well as feasibility and efficiency. Although the proposed approach was able to solve the multi-runways AAS problem, but it faces a challenge to deal with dynamical environment.

Bauerle et al. [32], have considered the ALP as a queueing system with the incoming aircraft corresponds to different types of customers and separation times between aircrafts corresponds to customer service times. First, they uses M/SM/1 queueing system to model a single runway, with semi-Markov service times. Moreover, the aircrafts average waiting time and the stability condition were given. Several routing heuristic strategies are considered such as type splitting,

Round Robin, coin flipping, and variants of the join-the-least-load rule. These strategies were numerically compared with their the average delay.

Atkin et al. [33], have tackled the static version of the ATP by proposing a hybrid metaheuristic approach which uses different search methodologies and a heuristic method. This approach decrease some future problems and assists runway controller by recommending schedules because of its ability to deal with huge amount of aircrafts. Their objective function contains a reordering cost, a weighted sum of aircraft delay, scheduling a take-off near to a boundary of the window or for a non-linear cost for violation of a CTOT time window and an penalty cost for excessive delay to an aircraft. Data from London Heathrow airport was used to evaluate the model and the result shows the effectiveness this system when compared against real-world schedules.

Van Leeuwen et al. [34], have treated the ATP problem as a constraint satisfaction problem, using ILOG Solver. He considers flights as activities and the runways, exit points and taxiways as resources. Moreover, different types of constraints like separation, time-slot and take-off order were listed as resource constraints or temporal in the environment of ILOG. Real data from Prague airport was used to evaluate the model with up to 12 aircrafts and the result was discussed. The model fails to find a solution in realistic time when problem size gets larger.

The second category is more specific to the aircraft and is related to control algorithms where the scheduled outputs were implemented and the effect of human factors were studied. The study by Haus et al in [10], is a prime example where they investigated the importance of automation during taxiing and how can this aids the human pilots in decision making. To implement the 4DT successfully, It is crucial to integrate these two categories to work seamlessly together. Particularly, the waypoints and the optimal time deadline should be determined by the scheduler based on specific scheduling approach [35], while this optimal schedule is followed by the specific aircraft in an optimal manner. Although the scheduler determines the schedule based on high level information such as reducing number of turns, taxi times, and number of accelerations during taxiing, it is imperative that the scheduling algorithm determines the schedule based on realistic aircraft parameters and constraints [36]. This schedule is then realistically followed by the aircraft where the pilot is involved in the loop as

the controller. Sometimes, the scheduler provides the optimal speed profile (such as in [7] and [37] [38]). Typically, as pilots have high workload and have to complete a many tasks from the checklist during taxi rolls, it is not practical to ask them to follow a strict speed profile. For this reason, [39] has studied taxiing with fully automated systems. However, using auto-taxiing has many challenges in surface operations, so it is not widely favoured to use full auto-taxiing modes [39]. Nevertheless, as discussed in [1] and [2], to meet the strict 4DT requirements, it is needed to employ full auto-taxi systems. However, employing partial auto-taxiing modes will circumvent this challenge. For example, the study in [39] has focused on the so-called human-centred 4D surface navigation system where the taxiway lighting elements are used to aid the pilot in in decision making, hence the pilot still remains in charge. This approach is called the follow-the-greens approach and now several major airports around the world employ such approach, including Singapore Changi International and Heathrow [10]. Unfortunately, many deficiencies have accrued when using partial auto-taxiing mode. It is difficult to meet the required 4DT as full auto-taxiing will not be employed. Also, many unrealistic assumptions are usually made when designing the optimal control system, for example pilots using idle throttle lever position during taxiing [40]. As said in [41], control depends on the pilot's behaviour to determine the lever position setting during taxiing. Furthermore, many of taxiway guiding algorithms that guide the pilot do not consider minimising the engine emissions or fuel burn. For example, [10] introduced a guidance system that is only concerned with following the trajectory in a strict way.

Also many of the systems do not use adequate aircraft models. For example the study in [39] established the mathematical model of a taxiing UAV. The force on the landing gear is represented by a linear mass spring damper which may not be realistic for the TRANSIT project. However, the paper is clear and easy to understand. The study in [42] describes a robust control algorithm for controlling an aircraft represented by a non-linear detailed system of equations. The equations are very comprehensive but lack undercarriage effects. The work presented is a stochastic robust control algorithm that may be very useful in the latter stages of the TRANSIT project.

The simulation of a large jet transport – Volumes I and II in [43, 44] are the de facto manual for flight simulation engineers. The document was developed in 1971 by NASA and details the

equations of motion as well as extensive testing and validation data for the Boeing 747. The undercarriage model was represented by a non-linear spring damper system. As mentioned in the AGARD document in 1998, the document is old but still very relevant for analysing the aircraft on the ground movement. It is a very good resource from which data for TRANSIT's preliminary model will be derived.

Allerton [42], has written a book of principles of flight simulation that gives a systematic approach to developing the flight simulator. Aspects such as hardware and instrument systems are extensively discussed. Sample codes are provided as well. The model is based on the NASA documents.

Krüger et al. [45], have given a summary of the requirements for a good landing gear and the possible defects of the landing gear (such shimmy) as well as the dynamics during the ground roll. The paper also presents three software packages for the aircraft ground dynamics. Controlled landing gear as well as test cases scenarios are presented. The paper contains some important articles from which may for advanced modelling such as aeroelasticity and shimmy. May be too advanced for the preliminary model.

Young [46], have analysed the requirements in detail of the aircraft undercarriage. The requirements include position, absorption of energy to within 3.02m/s, smooth ride during taxiing, horizontal energy dissipation, reverse thrust/brakes, extreme weather. Aircrafts usually must meet the compromises between the conflicting demands of good ground ride, operating to safe life and absorption of energy leading to nonlinear strut characteristics. The paper details the energy dissipation requirements of a good landing gear as well as the rationale behind the design changes in the landing gear.

Allerton [47], has focused on advances underpinning flight simulation technologies, mathematical modelling, real time issues, actuators, visualization and projection systems. Talked about flight simulation standardisation through the help of the RAES group as well as learnt in the last 30 years. Also noted that Ed Link is the founder of modern flight simulation. Talked about the future of flight simulators where cost will continue to fall and simulations

will become more pervasive to become a major discipline. Paper provides a good summary of simulation influences especially in the UK. Simulators incorporating AI methods were not discussed.

Pritchard [48], has documented the last 10 years of research on approaches to solving the shimmy and brake-induced vibrations problem in aircrafts. Findings from the survey reveal that a variety of method are used in analyses, testing, modelling and simulation and well as recommendations for the solving these problems. The paper presents a good survey of landing gear problems. The paper however did not adequately analyse the importance of shimmy in a taxiing aircraft.

Boril et al. [49], have discussed simulation centres available to the Czech Air Force Personnel. The paper contains neither mathematical models nor simulation results. Unfortunately, the paper is not immediately relevant to TRANSIT.

Wood et al. [50], have proposed a low order model for the aircraft tyre, which is designed to run on aircraft based on the industry standard multi-body systems program MSc.ADAMS. The low parameter model is similar to Pacejka's magic formula where a set of input data is used to empirically predict the tyre forces and moments. The paper presents a new angle to tyre force modelling based on empirical data. Further investigation is needed to ascertain suitability for the TRANSIT project.

Kapadoukas et al. [51], have investigated the modes required to flight simulator validation to allow a simulator-to-flight match. High non-linearities were identified in the strut model.

Barnes et al. [52], have summarised advances made in flight simulation as at 1997. The documents details the modelling requirements of the simulator such as bandwidth, accuracy, realism and validation. Appendix two contains mathematical models for the friction coefficients as a function of several factors including condition of taxiway surface, yaw angle braking (with and without ABS). These simple mathematical expressions facilitate their fast

implementations on a computer. This is a very good document, which provide an alternative to the Hanke model (especially undercarriage model). Data is rather limited though.

Daniels [53], has provided very detailed modelling and simulation of the undercarriage system. The main gear of the intruder were analysed to account for the inherent non- linearities (such as nonlinear spring and damping, polytrophic gas law, velocity squared damping, stick-slip friction effects). Paper is worth consulting when a more detailed undercarriage model is required.

Hajiloo et al. [54], have proposed methods for controlling shimmy vibrations in aircrafts using a stochastic MPC algorithm. The paper claims shimmy normally occurs at speed greater than 10 knots. The stochastic MPC approach used in the paper allows for scenarios of uncertainty. This paper may come in handy in the latter part of TRANSIT when one wants to handle uncertain models/data/measurements.

Esmailzadeh et al. [55], have detailed different mathematical models for shimmy analysis. It was reported in the paper, lower taxiing speed leads to higher stability and the landing gear becomes stable at speeds lower than 10 knots (5.144 m/s). It is also reported that the shimmy vibrations increase with increased velocity. Before the aircraft touches down, forward velocity is supposed to be lower than 150 knots (77.2 m/s). Based on a nonlinear model of the mechanics of the landing gear and tyre elasticity (based on the elastic string theory), some well-known linear and nonlinear mathematical methods are applied to the shimmy analysis of a simple model of a nose gear instead of complicated functions. The paper is worth investigating further what modelling changes in the preliminary model is needed so as to be able analyse Shimmy i.e. how detailed should the model be and if adding this level of detail necessary?

Krüger et al. [56], have used multibody modelling or a mixture of multibody and finite element modelling including time domain simulation to understanding the landing gear dynamics. Numerical methods on how vibration problems in landing gears can be treated are also discussed.

Somieski [57], has provided detailed mathematical models of the landing gear for understanding Shimmy and other vibrations.

Currey [58] has written a standard textbook for understanding landing gear design principles. The books cover issues ranging from tyre selection, suspension design and location.

Raymer [59], has dedicated a few pages on global landing gear parameters like placement and overall mass, without going into detail on landing gear design and dynamics. The book is useful for design engineers.

Ladda et al. [60], have collected studies from several leading authors studying the variation in landing gear load during the course of an aircraft operation. The document is very useful as it provides data and algorithms for analysing the loads on the landing gear. In the preliminary model, the load is assumed fixed. This assumption may not be tenable.

Denti et al. [61], have presented studies on shimmy prediction and brake modelling. Discusses the effect of different tyre models and brake on the longitudinal dynamics of aircraft landing gear.

Vaishnav [62], has compared the costs and benefits of using fuel and cost saving taxiing methods such as using electric, diesel or gasoline tugs to tow aircraft on the tarmac. Potentially, this paper provides baseline with which we can compare our fuel savings. Potential for fuel reduction through electric taxiing. In the study, simple models were used to investigate the fuel savings when using electric taxiing method. Detailed statistical analyses show that the electric taxiing can provide fuel savings. The methods used in the paper to establish this fact can be adapted to our work later on when justifying the fuel savings advantage of TRANSIT.

Zhang et al. [63], have established the mathematical model of a taxiing UAV. The force on the landing gear is represented by a linear mass spring damper which may not be realistic for the TRANSIT project. However, the paper is clear and easy to understand.

Bo et al. [64], have investigated the yawing of a taxiing aircraft in a unideal condition (such as asymmetry of brake devices, side wind, tyre and course coefficient differences). It is proposed to use a type of differential braking based on the slip ratio for aircraft dynamics, which contains an inbuilt side wind model.

Khadilkar et al. [41], have proposed a model for estimating the fuel consumption on the ground given the aircraft surface trajectory. It was found in the study the proposed method includes several acceleration modes which play a major role in the amount of fuel consumed by a taxiing aircraft.

Ferguson [65], has created a 500 page manual containing mathematical modelling steps and data for the tilt rotor aircraft. As with the NASA reports may be more relevant to the TRANSIT project, this article may be redundant.

Gillespie [66], has created a good resource for understanding the mathematics of aircraft dynamics.

Wong [67], has provided a theoretical approach to the mathematical modelling of ground vehicles. Detailed analyses of the interface between the vehicles and the ground are investigated.

Khapane [68], has presented a multibody system model for studying aircraft ground dynamics. Provides very good equations of the forces which may be adapted for the TRANSIT Project.

Wang [69], have provided a robust control algorithm for controlling an aircraft represented by a non-linear detailed system of equations. The equations are very comprehensive but lacks undercarriage effects. The work presented is a stochastic robust control algorithm that may be very useful latter stages of the TRANSIT project.

1.2 Issues and Shortcoming in the State of the Art

The gaps and shortcoming in research include:

- Many of taxiway guiding algorithms do not consider minimising the engine emissions or fuel burn.
- Many of the systems do not use adequate aircraft models.
- Pilots using idle throttle lever position during taxiing.

This project will provide new approaches towards aircraft taxiing around airports. Additionally, this project will study taxiing the aircraft in an optimal manner using a high fidelity aircraft model of a large Boeing 747 jumbo jet. The overarching research questions are:

- How to use artificial intelligence for optimal aircraft taxiing at airports?
- How to improve the taxiing of a large Boeing 747 jumbo jet aircraft model through design of artificial intelligence based control models to taxi around airports for minimal fuel consumption and minimum CO2 emissions?

1.3 Aims and Objectives

The research topic of this project is aircraft taxiing using artificial intelligence (AI). AI, has proved its success in many control applications today. Very recently, researchers from the University of Sheffield have built a powerful aircraft model for efficient aircraft taxiing at airports. AI can be used for designing algorithms for efficient and optimal aircraft taxiing. The research primary aims to further improve aircraft taxiing through designing AI models to taxi around airports for minimal fuel consumption and minimum CO2 emissions.

The achievable objectives in this thesis include:

- Simulating a non-linear Boeing 747-100 aircraft model in MATLAB Simulink.
- Designing aircraft navigation system.
- Designing a generalized PID controller for aircraft taxiing.
- Designing artificial neural networks controller for aircraft taxiing

- Designing a Fuzzy Inference System (FIS) model for aircraft taxiing.
- Designing an online controller using reinforcement learning for aircraft taxiing.
- Validating all the proposed systems using several taxiing scenarios.

1.4 Research Methodology

The methodologies to achieve each objective are discussed below:

I. Using the NASA reports to drive the non-linear Boeing 747-100 aircraft model simulated in MATLAB Simulink for taxiing only (excluding flying) and teste it for validation of the control algorithms.

II. Designing a MATLAB Simulink navigation system that allows aircraft to move from one point to the other on the ground in an optimal manner.

III. Designing a generalized PID controller with clustering of taxiing scenarios for aircraft taxiing. Building three sets of PID controllers that manipulate the throttle, brakes, and rudder respectively.

IV. Designing backpropagation offline artificial neural networks controller for aircraft taxiing with two inputs (speed error, change in speed error) and two outputs (throttle value, brake value).

V. Designing a fuzzy inference system (FIS) model for aircraft taxiing and using ANFIS to tunes (adjusts) the membership function parameters. Building two fuzzy inference systems, one for the throttle and one for the brake using two ways: grid partition and fuzzy c-mean clustering.

VI. Designing an online controller for aircraft taxiing by using reinforcement learning and using the Deep Deterministic Policy Gradient (DDPG) for the agents. Building two DDPG agents, one for the throttle and one for the brake and using the reward to measure the success of the action in achieving the goal of the task.

VII. Validating all the proposed systems using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout with simulation results.

1.5 Thesis Overview

The thesis's contents are divided into seven chapters as follows:

Chapter two provides a comprehensive overview of the aircraft model used in this project. The chapter begins by introducing the non-linear Boeing 747-100 aircraft which is used in this study and the state-space model formulation. Thereafter, the chapter outlines the three axis system of the aircraft, and then discusses the aircraft state equations. Then, it is carefully defining and explaining the derivations of aircraft force and moment. Furthermore, some simulation experiments are conducted and the output responses are plotted.

Chapter three is concerned with the aircraft navigation system which consists of several blocks that allows aircraft to move from one point to the other on the ground in an optimal manner. Then, the outer loop controller that determines the speed and heading references is explained with a detailed example of moving the aircraft from along a straight line of distance 500 m in 50s". The turning segments is also outlined. Afterward, it reports on a PID controller for aircraft taxiing and introduces the three sets of PID controllers existed in the inner loop control. Then, the occurrence of control problems when one is confronted with different taxiing scenarios is explained. Thereafter, it outlines the solve of this problem by using the clustering strategy on the lists of scenarios. Then, the PID controller is validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout with simulation results.

Chapter four describes the proposed artificial neural networks control for aircraft taxiing. It starts by a detailed explanation of the concepts of backpropagation artificial neural networks with its layers. Thereafter, The equations of each step in artificial networks has been presented. The steps for designing the neural network model for this project has been provided. Then, the artificial neural networks controller is validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout with simulation results.

Chapter five reviews the fuzzy inference system and its structure and parameter adjustment, also explains the ANFIS (adaptive neuro-fuzzy inference system) and its main usage in the fuzzy inference system. Thereafter, it describes the built of proposed fuzzy inference system for aircraft taxiing using the MATLAB ANFIS toolbox with grid partition and fuzzy c-mean clustering ways. Then, the fuzzy inference system controller with both grid partition and fuzzy c-mean clustering is validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout with simulation results.

Chapter six reports on the way of designing the online controllers for aircraft taxiing using reinforcement learning specifically the Deep Deterministic Policy Gradient (DDPG) Agents with two DDPG agents, one for the throttle and one for the brake. The chapter starts by an introduction to the reinforcement learning and its applications. Then clarifies the difference between the reinforcement learning and the traditional control. Thereafter. it lists and explains the workflow of reinforcement learning. Then, the reinforcement learning controller is validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout with simulation results.

Chapter seven concludes and summarises the conducted work in the thesis and outlines recommendations for future work in the aircraft taxiing area.

2 Aircraft Model Development

2.1 Introduction

The aircraft model used in this study is the non-linear Boeing 747-100 aircraft simulated in MATLAB Simulink. A state-space model formulation is the mathematical model that is used which is derived from the general equation of motion of a rigid body and are presented in this section [70]. According to Newtonian mechanics, for a rigid body in motion, the rates of change of the linear and angular velocities are related to the total forces and moments acting on the body, as given by the following equation:

$$\begin{aligned} \mathrm{F} &= m\left(\frac{\partial V}{\partial t} + \Omega \times \mathrm{V}\right) \\ \mathrm{M} &= \frac{\partial(\mathrm{I}\cdot\Omega)}{\partial t} + \Omega \times (\mathrm{I}\cdot\Omega) \end{aligned} \tag{2.1}$$

Where $\mathrm{F} = [F_x\, F_y\, F_z\,]^T$ and $M = [L\; M\; N]^T$ are the force and moment vectors along the x, y, and z axes in the body axis system, respectively; $V = [u\; v\; w]^T$ is the linear velocity vector, where u, v, and w are the velocity along the x, y, and z axes, respectively; $\Omega = [p\; q\; r]^T$ is the angular velocity vector, where p is roll rate, q is pitch rate, and r is yaw rate; m is the mass of the aircraft; t is time; and I is its constant inertia tensor (which are moments and products of inertia of the aircraft) [59].

If the frame of reference is fixed to the vehicle these values are constant, regardless of the attitude of the vehicle. Therefore I is given as follows:

$$\mathrm{I} = \begin{bmatrix} I_{xx} & -J_{xy} & -J_{xz} \\ -J_{yx} & I_{yy} & -J_{yz} \\ -J_{zx} & J_{zy} & -I_{zz} \end{bmatrix} \tag{2.2}$$

where the elements indicated as I represent moments of inertia about the axis defined in the subscript and the components indicated as J represent the product of inertia about the axis defined in the subscript.

State-space formulation

It is often useful to derive the rigid body equation in the state space form. Doing so immediately offers the possibility of applying a plethora of theoretically matured control algorithm. Equation (2.1) can be rewritten into a state-space formulation according to the following equation:

$$\begin{aligned} \frac{\partial V}{\partial t} &= \frac{F}{m} - \Omega \times \mathrm{V} \\ \frac{\partial \Omega}{\partial t} &= \mathrm{I}^{-1}(\mathrm{M} - \Omega \times \mathrm{I} \cdot \Omega) \end{aligned} \tag{2.3}$$

It is easily seen that equation (2.3) is of the standard state-space form and can be written in the general form as follows:

$$\dot{x} = f(x(t), u(t), v(t), t) \tag{2.4}$$

Where $x \in R^K$ are the states with the corresponding time derivatives denoted as $\dot{x}$; R represents real number; K represents the total number of states; $u \in R^W$ represents the time-variant inputs of dimension W; and $v \in R^Z$ is the disturbance vector (e.g., atmospheric disturbances) of dimension Z. It should be noted that equation (2.4) represents the nonlinear time variant state equation in which the forces and moments exerted on aircraft are implicitly modelled.

It should also be noted that the nonlinear time-invariant equivalent of equation (2.4) generally suffices to model any realistic physical dynamical systems, including aircraft. Therefore, a nonlinear time-invariant model was assumed in this study. For example, although the moment of inertia may depend on many other factors such as the mass of aircraft, it is not dependent on time (i.e., age of aircraft). In the Boeing 747-100 aircraft model utilized in this study, the number of state variables is 13. In particular, the states include the standard 12-dimensional

variable (which governs the body rates and accelerations) as well as an additional variable that has been called ‘‘pre-thrust” and represents the dynamics of the engine. The state equations are briefly described in Section 2.4.

2.2 The Boeing 747 Aircraft Model

The Boeing 747-100 is a four-fanjet jumbo transport aircraft that first came into operation in 1967. For taxiing, the Boeing 747-100 is equipped with a tiller and a rudder that facilitate directional control. Thrust is assumed to be generated using the two inner engines. The Boeing 747-100 is also equipped with inward and outward Krueger flaps and a movable stabilizer with four ailerons, but these are not included in the modelling as they are assumed to not be in use during taxiing. The datasets utilized for modelling the components of the Boeing 747-100 used in this study were obtained from [43, 44].

The convention of the forces and moments acting in the body frame on the aircraft is shown in Figure 2.1. The summary of the Boeing 747 dimensions and areas are illustrated in Table 2.1. It should be noted that the engine and fuel systems are modelled based on the International Civil Aviation Organization (ICAO) database.

Table 2.1:Summary of the Boeing 747 Dimensions and Areas [70].

Item	Symbol	Value
Wing Area	S	$510.97m^2$
Wing Mean Aerodynamic Chord	$\bar{c}$	$8.32m$
Wing Span	b	$59.65m$

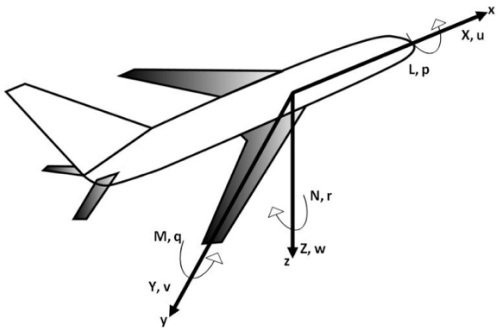


Figure 2.1:The aircraft Body Axis with force and moment conventions [70].

2.3 Axis Systems

The variables and data sets of the aircraft are given in the three axis systems listed below:

I. The Body Axis System: the Newtonian mechanics variables equation (2.1) are derived in the body axis. Therefore, if a force or moment is in any other coordinate system apart from the body axis system, it is transformed into the body axis integration is performed. The origin of the body axis system is the centre of gravity of the aircraft. The x-axis is directed towards to nose of the aircraft, the y-axis towards the right wing and the z-axis bottom (towards the gear). This axis system is shown in Figure 2.1 with the arrows indicating positive conventions.

II. The Earth Reference System: This is also called the topodetic reference system or the earth- fixed reference system. This axis is considered fixed in space it is from here that the aircraft is observed to move. The vehicle's position (x,y) as well as the vehicle's height (h) are all measured from the centre of the earth reference axis. It is worth noting that it is sometimes convenient to introduce a new axis system called the vehicle carried vertical axis system. This system is obtained by translating (no rotation!) the earth reference system so that its origin coincides with the centre of gravity of the aircraft. The angles between the earth reference axis and the vehicle carried axis is referred to as the attitude of the aircraft (called roll, pitch and yaw).

III. The Wind Axis System: This axis is aligned with the velocity vector of the aircraft. The angle of attack (α) and the angle of sideslip (β) define the orientation of the wind axis with the body axis. The aerodynamic coefficients (for lift, drag, side force, pitching moments, yawing moments and rolling moments) are all given in the wind axis. As such, it is necessary that when the aerodynamic forces and moments are calculated, that they are transferred into the body axis system. Undercarriage Axis System: During taxi, the undercarriage exerts significant forces and moments on the body frame of the aircraft. These forces and moments were derived in the respective local frames of each gear and is needed to be transformed into the body axis. The stability axis can be thought of as the body axis rotated by the angle α.

The relationship between the various axis systems utilised in developing the Boeing model is shown in Figure 2.2.

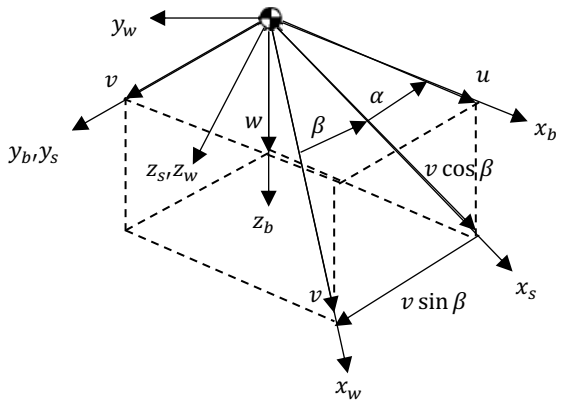


Figure 2.2: Axis systems utilized in developing the Boeing Model (undercarriage not included)[44].

It is important to emphasize that a mixed reference system is utilised in this thesis as evidenced by the states chosen in the next section.

2.4 State Equations

The aircraft rigid body equations are converted into state space form. It consists of 13 functions of nonlinear state equations which are:

$$\mathrm{x} = [\mathrm{x}_1{}^T\ \mathrm{x}_2{}^T\ \mathrm{x}_3{}^T\ \mathrm{x}_4{}^T\ \mathrm{x}_5{}^T]$$

where

- $\mathrm{x}_1 = [p\ q\ r]^T$ represents the rotational velocity
- $\mathrm{x}_2 = [\mathrm{V}\ \alpha\ \beta]^T$ represents the translational velocity
- $\mathrm{x}_3 = [\phi\ \theta\ \psi]^T$ represents the attitude
- $\mathrm{x}_4 = [h\ x\ y]^T$ represents the position
- $\mathrm{x}_5 = \mathrm{prethrust}$ is the pre-thrust sub-vector

x_1 depends on the roll rate p, pitch rate q, and yaw rate r. x_2 is a function of velocity vector V, angle of attack α, and sideslip angle β. The attitude vector x_3 is a function of roll angle ϕ, pitch angle θ, and yaw angle ψ. The position vector x_4 consists of the altitude h and the x and y positions. It is worth emphasizing that the rotational and translational velocities are defined

within the body/wind axis of the aircraft, while the attitude and position are defined in the earth axis frame. The last variable represents the thrust, which is also given in the body axis of the aircraft. The vector function f , which relates the vector states derivatives to the vector states and the inputs, is given by the following equation:

$$f(x(t), \dot{x}(t), u(t)) = [f_1^T \ f_2^T \ f_3^T \ f_4^T \ f_5^T \ ]^T \tag{2.5}$$

Where f_1, f_2, f_3, f_4 and f_5 represent the vector sub-functions that relate the rotational, translational, attitude, position, and pre- thrust vectors to the state variables, respectively. Each of these sub-functions are developed separately in following sections:

2.4.1 Rotational acceleration function (f_1)

The rotation acceleration functions relate the states to the derivatives of the rotational velocities in the body frame. The derivation of the equations is extensively presented in [71] and is summarized by the following set of equations:

$$\begin{aligned} \dot{p} &= \frac{1}{I_x}[L - (I_z - I_y)qr + I_{xz}(pq + \dot{r})] \\ \dot{q} &= \frac{1}{I_y}[M - (I_x - I_z)rp + I_{xz}(p^2 - r^2)] \\ \dot{r} &= \frac{1}{I_z}[N - (I_y - I_x)pq + I_{xz}(qr - \dot{p})] \end{aligned} \tag{2.6}$$

where $p, q,$ and r are the rotational rates along the $x, y,$ and z body axes, respectively; $I_x, I_y, I_z,$ and I_{xz} are moments of inertia along the x, y, z and $x - z$ body axes, respectively; and L, M, N are the body axis moments along the $x, y,$ and z body axes, respectively.

2.4.2 Translational acceleration function (f_2)

The translational acceleration function relates the states to the derivatives of the body velocities. It should be noted however, that such velocities are better suited in the wind axis but the respective body velocities (u, v and w) and rates ($\dot{u}$, $\dot{v}$ and $\dot{w}$) can easily be derived as

will be shown during derivations for output equations (section 2.5). The derivations of the translational acceleration function is given as follows:

$$\begin{aligned}
\dot{V} &= \frac{1}{m}[-D\cos\beta + Y\sin\alpha + X_{\mathrm{T}}\cos\alpha\cos\beta + Y_{\mathrm{T}}\sin\beta + \\
&\quad Z_{\mathrm{T}}\sin\alpha\cos\beta - mg(\cos\alpha\cos\beta\cos\theta)] \\
\dot{\alpha} &= \frac{1}{mV\cos\beta}[-L + Z_{\mathrm{T}}\cos\beta - X_{\mathrm{T}}\sin\alpha + \\
&\quad mg(\cos\alpha\cos\beta\cos\theta + Z_{\mathrm{T}}\sin\alpha\sin\theta)] + q - \tan\beta\,(p\cos\beta + r\sin\alpha) \\
\dot{\beta} &= \frac{1}{mV}[D\sin\beta + Y\cos\beta - X_{\mathrm{T}}\cos\alpha\sin\beta + \\
&\quad mg(\cos\alpha\cos\beta\cos\theta + Z_{\mathrm{T}}\sin\alpha\sin\theta)] + p\sin\alpha - r\cos\alpha
\end{aligned} \tag{2.7}$$

where V, α, and β are the total velocity, angle of attack and angle of sideslip, respectively; $m, g,$ and θ are aircraft mass, acceleration due to gravity at sea level, and pitch angle, respectively; D is the total aerodynamic drag; Y is the total aerodynamic side force; and X_{T}, Y_{T}, and Z_{T} are the total thrust forces along the x, y, and z axes in the body axis system.

2.4.3 Attitude rates

The attitude of the aircraft defines the angles the inertia frame makes with the body frame. The angles are the widely known roll (ϕ), pitch (θ) and yaw angles (ψ). It should be noted that the angles ($\phi\ \theta\ \psi$) are also called the Euler angles. The rates of the attitudes are given below. The rates of the attitudes are given as follows:

$$\begin{aligned}
\dot{\phi} &= p + q\sin\phi\tan\theta = r\cos\phi\tan\theta \\
\dot{\theta} &= q\cos\phi - r\sin\phi \\
\dot{\psi} &= q\sin\phi\sec\theta - r\cos\phi\sec\theta
\end{aligned} \tag{2.8}$$

2.4.4 Earth relative velocity

The Earth relative velocities $\dot{x}, \dot{y},$ and $\dot{h}$ along the x, y, and z earth axes, respectively, can be expressed by the following equation:

$$
\begin{aligned}
\dot{h} &= V(\cos\alpha\cos\beta\sin\theta - \sin\beta\sin\phi\cos\theta - \sin\alpha\cos\beta\cos\phi\cos\theta) \\
\dot{x} &= V[\cos\alpha\cos\beta\cos\theta\cos\psi + \sin\beta(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + \\
&\quad \sin\alpha\cos\beta(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)] \\
\dot{y} &= V[\cos\alpha\cos\beta\cos\theta\sin\psi + \sin\beta(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi) + \\
&\quad \sin\alpha\cos\beta(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)]
\end{aligned}
\tag{2.9}
$$

2.4.5 Pre-thrust rate

In the Boeing 747 model, the engine pressure ratio (EPR) henceforth called the prethrust determines the thrust produced by the aircraft. However, this prethrust variable is represented by a first order dynamical system with a time constant of 5 seconds. It is necessary to convert this to the state-space form to make it compatible with other state equations and facilitate developing the model. Given the static EPR (steady state value) which is dependent on the throttle setting, the prethrust rate is given by the following equation:

$$
\dot{prethrust} = -0.5\, prethrust + 0.5g\,(throttle, states) \tag{2.10}
$$

where g is a function which relates the EPR to the throttle setting and represents the steady state EPR. Given the prethrust, the thrust generated by the engines can be calculated as will also be shown in section 2.5. It should be noted that the state equations are applicable to any aircraft. However, the forces and moments data are typically peculiar to the aircraft. The derivations for the force and moment components as well as the dataset utilized are described in Section 2.6.

2.5 Output Equations

The following equations have also been utilised so that some the required outputs can be observed in the simulator. The output equation is generally derived from the states and their derivations are discussed as follows:

a- The body axis rates are given by the following equation:

$$
u = V\cos\alpha\cos\beta
$$

$$v = \sin\beta$$
$$w = V \sin\alpha \cos\beta \tag{2.11}$$

b- The body axis accelerations are given by the following equation:

$$\dot{u} = \frac{X_T - gm\sin\theta - \mathrm{D}\cos\alpha + L\sin\alpha}{m} + rV\sin\beta - \mathrm{qV}\sin\alpha\cos\beta$$
$$\dot{v} = \frac{Y_T - gm\sin\phi\cos\theta + Y}{m} + pV\sin\alpha\cos\beta - rV\cos\alpha\cos\beta \tag{2.12}$$
$$\dot{w} = \frac{Z_T - gm\cos\phi\cos\theta - D\sin\alpha - L\cos\alpha}{m} + qV\cos\alpha\cos\beta - pV\sin\beta$$

It should be noted that the equations above represent the generic aircraft equations. Hence, the developed approach is generic and is easily extensive to include other aircraft models as has been done with the Boeing A320 Model. What differentiates different aircraft is how the forces and moments are derived as well as their respective data. The derivations for the force and moment components as well as the data set utilised as described in section 2.6. The maximum control deflections are listed in Table 2.2 . To aid in scaling, a list of maximum values is given in Table 2.3.

Table 2.2:Maximum control surface deflections of the Boeing 747 as listed in the NASA reports [43, 44].

Control Surface	Symbol	Maximum Displacement (deg)	Normal operation (Full Bost) Rate (deg/sec)	One Hydraulic System Failure Rate (deg/sec)
Elevators:				
Inboard	δE_I	$\begin{cases}+17\\-23\end{cases}$	37 down 37 up	30 down 26 up
outboard	δE_D	$\begin{cases}+17\\-23\end{cases}$	37 down 37 up	30 down 26 up
Stabilizer	$\delta_{F.R.L}$			
Pilots Thumb Switch		$\begin{cases}+0.5\\-10\end{cases}$	0.5-0.2	0.25-0.1
Control Stand Levers		$\begin{cases}+3\\-12\end{cases}$	0.5-0.2	0.25-0.1
Ailerons:				
Inboard	δA_I	$\begin{cases}+20\\-20\end{cases}$	40 down 45 up	27 down 35 up
Outboard	δA_D	$\begin{cases}+15\\-25\end{cases}$	45 down 55 up	22 down 45 up
Spoilers:	δ_{SP}			
Panels 1,2,3,4,9,10,11,12		45	75	
Panels 5,8		20	75	
Panels 6,7 (Speedbrakes only)		20	25	
Rudder:				
Upper	δR_U	$\begin{cases}+25\\-25\end{cases}$	50	40
Lower	δR_L	$\begin{cases}+25\\-25\end{cases}$	50	40

Table 2.3:Maximum value of the aircraft rates as listed in the NASA reports [43, 44].

Parameter	Maximum value	Units
a_x	30	ft/sec^2
a_y	25	ft/sec^2
a_z	120	ft/sec^2
$\ddot{\theta}$	50	deg/sec^2
$\ddot{\phi}$	130	deg/sec^2
$\ddot{\psi}$	30	deg/sec^2
$\dot{\theta}$	60	deg/sec
$\dot{\phi}$	75	deg/sec
$\dot{\psi}$	30	deg/sec
$\dot{\alpha}$	40	deg/sec
$\dot{\beta}$	30	deg/sec
Altitude	45000	ft
Rate of Climb	15000	ft/min
Rate of Descent	15000	ft/min
Centre of Gravity Range	0-40	%MAC

2.6 Force and Moment Derivations

The equations of forces and moments acting on the aircraft model are described in this section which involve aerodynamic, engine, undercarriage and gravity components. Data of the equations were obtained from [43] and [44].

2.6.1 Aerodynamic Forces and Moments

At relatively high speeds taxiing, the aerodynamic forces and moments can be significant. The aerodynamic forces and moments components are all calculated in the wind axis and are then transformed into the body axis [70]. It should be noted that the aerodynamic primary inputs (mainly ailerons, rudder, and elevator) influence these aerodynamic components through their respective aerodynamic coefficients. The data for calculating the aerodynamic coefficients is given in [44]. The six aerodynamic coefficients utilized for eliciting the aircraft model are listed as follows:

a. Lift (Aerodynamic lift coefficient C_L): The aerodynamic lift coefficient is given as follows:

$$C_L = C_{L,basic} + \frac{dC_L}{d\alpha}\alpha + \frac{dC_L}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V} + \frac{dC_L}{dq}\frac{q\bar{c}}{2V} + \frac{dC_L}{dn}n + \Delta C_{L,\ Landing\ gear} \tag{2.13}$$

where C_{L} is the airplane lift coefficient, $C_{\mathrm{L,basic}}$ is the basic lift coefficient for the rigid airplane, $\overline{c}$ is the wing mean aerodynamic chord, α is the angle of attack relative to fuselage reference line, n is the airplane normal load factor along the z-axis, and $\Delta C_{\mathrm{L,\ landing\ gear}}$ is the change in lift coefficient due to landing gear extension.

b. Drag

$$C_D = C_{D,basic} + \frac{dC_D}{dM}M + \Delta C_{D,\ Landing\ gear} + \Delta C_{D,\ rudders} \tag{2.14}$$

Where C_{D} is the airplane drag coefficient, $C_{\mathrm{D,basic}}$ is the basic drag coefficient for the rigid airplane, M is the Mach number, $\Delta C_{\mathrm{D,landing\ gear}}$ the change in drag coefficient due to landing gear extension, and $\Delta C_{\mathrm{D,rudders}}$ is the change in drag coefficient due to rudder deflection.

c. Side force

$$C_Y = C_{Y,basic} + \frac{dC_Y}{d\beta}\beta + \frac{dC_Y}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V} + \frac{dC_Y}{dq}\frac{q\bar{c}}{2V} + \frac{dC_Y}{dn}n + \Delta C_{Y,\ Landing\ gear} \tag{2.15}$$

where C_{Y} is the airplane side force coefficient, $C_{\mathrm{Y,basic}}$ is the basic side force coefficient for the rigid airplane, and β is the airplane sideslip angle.

d. Pitching moment

$$C_M = C_{M,basic} + \Delta C_M \frac{dC_M}{d\alpha}\alpha + \frac{dC_M}{d\dot{\alpha}}\frac{\dot{\alpha}\bar{c}}{2V} + \frac{dC_M}{dq}\frac{q\bar{c}}{2V} + \frac{dC_M}{dn}n + \Delta C_{M,\ Landing\ gear} + \Delta C_{M,\ rudders} \tag{2.16}$$

where C_{M} is the airplane pitching moment, $C_{\mathrm{M,basic}}$ is the basic pitching moment for the rigid airplane, and α is the airplane angle of attack relative to the wing design plane.

e. Rolling moment

$$C_l = \frac{dC_l}{d\beta}\beta + \frac{dC_l}{dp}\frac{pb}{2V} + \frac{dC_l}{dr}\frac{rb}{2V} + \Delta C_{l,\ rudders} \tag{2.17}$$

where C_{l} is the airplane rolling moment coefficient, b is the wing span, and $\Delta C_{\mathrm{l,rudders}}$ is the change in rolling moment coefficient due to rudder deflection.

f. Yawing moment

$$C_n = \frac{dC_n}{d\beta}\beta + \frac{dC_n}{d\dot{\beta}}\frac{\dot{\beta}b}{2V} + \frac{dC_n}{dp}\frac{qs}{2V} + \Delta C_{n,\ rudders} \tag{2.18}$$

where C_{n} is the airplane yawing moment coefficient.

The convention for positive and negative values for the aerodynamic surface deflections as utilised in the developed Boeing model is given in [43].

2.6.2 Engine Forces and Moments

The developed Boeing 747 aircraft model is equipped with four (4) Pratt and Whitney JT9D-3 engines with a take-off thrust of 43500 pound force (193500 N) [70]. Only two engines are typically utilized for taxiing. As such, although it is easy to ‘switch on’ the other two engines

in the developed MATLAB aircraft model. In modelling the aircraft engines the reverse thrust has not been used as it is not typically utilised during taxiing.

A series of steps is necessary to calculate the final thrust generated by the engine. The steps are described as follows:

a) Get the throttle lever angle and then use this angle to calculate the engine power lever angle (PLA). Note that in the Boeing 747 model developed in Sheffield, this step has been somewhat modified so that the input to the system is a throttle setting normalised between zero and 1. Such a normalisation makes sense as one can see that original forward thrust angle is linearly related to the engine power lever angle. In particular, this means that the PLA of between 60° and 130° degrees is transformed to be between 0 and 1 throttle in the developed Boeing 747-100 aircraft mode. Such a transformation is shown in Figure 2.3. It is also worth noting that the ambient temperature utilised is 59°F.

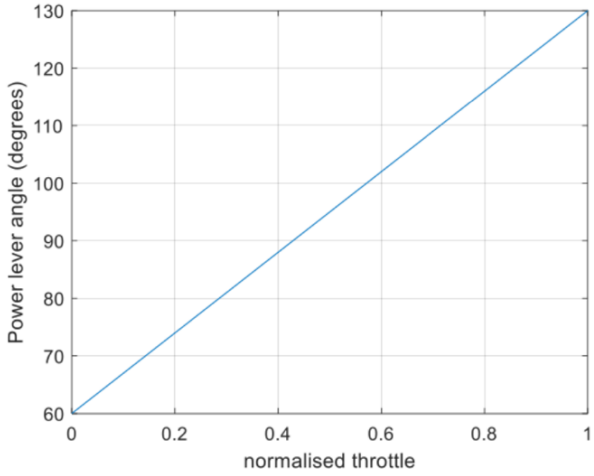


Figure 2.3: Normalized throttle between 0 and 1 [72].

The following formula is used for the normalisation:

$$thr = \frac{PLA - \min(PLA)}{\max(PLA) - MIN(PLA)} \tag{2.19}$$

where PLA is the power lever angle and thr is the throttle. The throttle setting is then related to the engine pressure ratio (EPR) using a non-linear polynomial fit as discussed next.

b) Given the engine power level angle (or throttle normalised to between 0 and 1 as performed in the Simulink model), the static engine pressure ratio (EPR) is determined. The data for such a determination of static EPR is given in [44]. The polynomial model is given by the following equation:

$$epr = k_0 + k_1 * thr + k_2 * thr^2 + k_3 * thr^3 \quad (2.20)$$

The parameters k_0 , k_1, k_2 and k_3 are the coefficients and given in Table 2.4. The plot of the EPR as a function of throttle position is shown in Figure 2.4.

Table 2.4:Function coefficients used for calculating the static engine pressure ratio [70].

Coefficients	k_0	k_1	k_2	k_3
Values	1.0067	0.15098	0.3757	-0.0291

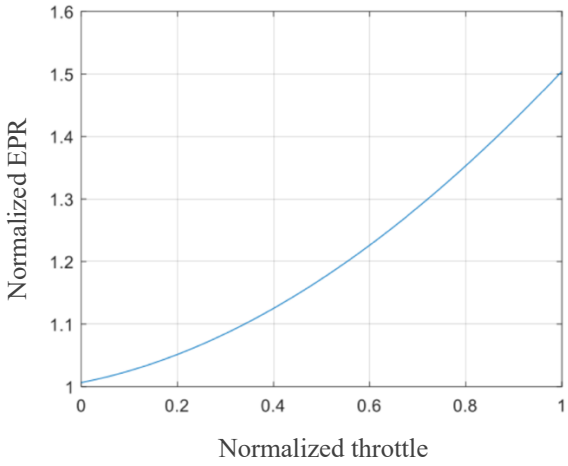


Figure 2.4: Plot of the EPR as a function of throttle position [72].

The EPR obtained so far is referred to as the static EPR which crudely represents the steady state EPR without taking into account EPR changes due to changes in Mach number. This static EPR must be represented using a dynamical system so as to include the engine transients. This process is described next.

c) As already stated, the engine includes a transient, which is represented by a first-order dynamic system with a 5 s time constant. The static EPR calculated in the last step represents the reference level/steady state of the EPR. It is worth emphasizing that the transients have been represented in a state-space format, hence the need to add an additional state pre-thrust, as stated earlier. In essence, the pre-thrust variable represents the dynamic EPR.

d) The National Aeronautics and Space Administration (NASA) reports in [43, 44, 73] provide plots for adding incremental EPR to the dynamic EPR as the Mach number of the aircraft increases. However, this has not been included in the simulation, as our taxiing speeds are typically less than 40 knots (21 m/s), which would result in negligible incremental EPR due to the speed increase.

e) When EPR is known (which is of course dynamic in nature), the final step involves calculating the net thrust per engine thrust. The net thrust is calculated as a function of the engine EPR as well as the ambient conditions. The equation for deriving the net thrust is given as follows:

$$thrust\ (LB) = (\ slope * EPR + intercept) * 1000 \tag{2.21}$$

where the slope and intercept are given in Table 2.5. The plot of the thrust in (thousand LB) as a function of the EPR is shown in Figure 2.5.

Table 2.5:Linear Slope and intercept of the net-thrust per engine as a function of EPR [72].

Intercept	-90
Slope	90

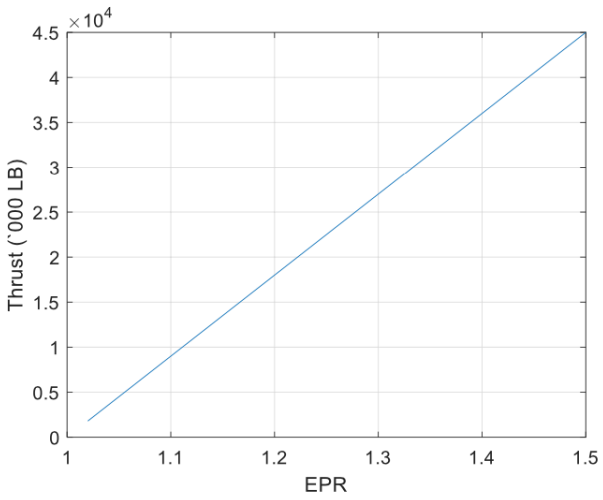


Figure 2.5:Thrust as a function of EPR [72].

As calculations were performed using S.I. units in the developed model, the thrust was converted into Newtons by multiplying by the factor 4.44822. It is worth emphasizing that the thrust derived in the preceding steps is for one engine only. If more than one engine is used in taxiing, then the total thrust must be amended accordingly. In summary the process of calculating the thrust developed per engine is shown in Figure 2.6.

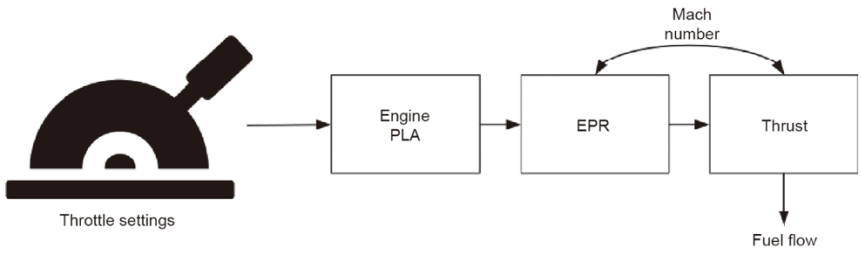


Figure 2.6:Steps for calculating the net thrust [70].

Idle thrust:

It is very important to note that at a throttle setting of zero, the EPR has an approximate value of 1.02, which makes the engine produce positive net thrust. It is also worth noting that the engines generate a moment about the centre of gravity of the air- craft. While the net moment

due to engine thrust is zero during the two-engine taxiing mode (in which inner engines are used and outer engines are switched off), for the single-engine taxiing mode, a moment will be produced that will make the aircraft yaw. This moment is easily calculated as the product of the force and the distance of the engine from the aircraft's centre of gravity. This distance is shown in [43, 44] as being equal to 39.167 ft (11.938 m).

Fuel flow and emissions:

Since one of the goals of this study is to find more efficient methodologies for aircraft taxiing, an accurate measure of engine fuel consumption and emissions is crucial. Consequently, an engine fuel consumption model based on the ICAO database of fuel and emissions [74] was developed. The ICAO database has been used extensively in the literature for studying the fuel consumption and emissions of engines at various thrust settings [41]. The database provides a comprehensive list of approximately 500 engines manufactured after 1980 as well as their fuel consumption and emissions at four distinct thrust settings. The distribution of the rated thrust of all the engines present in the database is shown in Figure 2.7.

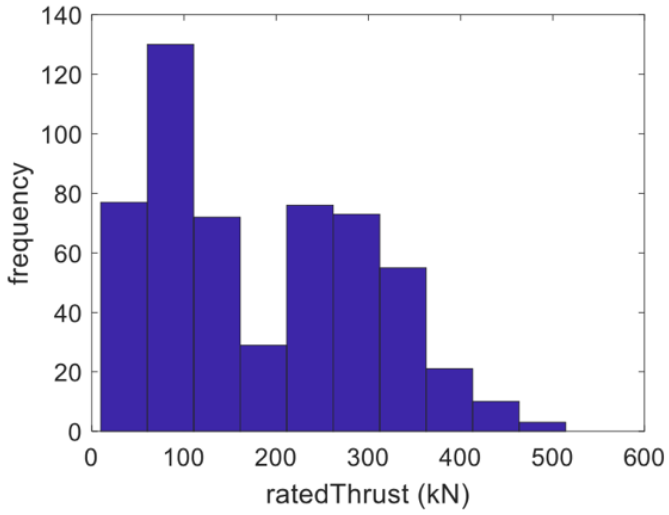


Figure 2.7:Distribution of rated thrust of the engines contained in the ICAO database [70].

In addition to the emissions and fuel statistics shown in [74], the ICAO database includes other important information relating to the engine, such as out-of-production status, type of fuel, engine manufacturer, and whether or not the engine is out of service. The JT9D-3 engine was

not present in the database list, which may be due to the fact that this engine was manufactured before 1980. To facilitate the calculations of fuel burn and emissions for the JT9D- 3 engine, a comprehensive statistical analysis of the database was undertaken. A fuzzy logic model has previously been utilized to analyse such data [3, 75, 76].

This analysis found that the rated thrust and out-of-production status of the engine were significant for predicting the fuel flow. Therefore, these two variables have been used for predicting the fuel flow of an arbitrary engine whose rated thrust and out-of-production status are known, as in the case of the JTD9-3 engine. The fuel flows at different values of thrust (four per engine) were arranged into a vector along with the engine out-of- production status. A polynomial was then utilized to identify a function that can be used for fuel flow and emission predictions. The data was divided into training and testing datasets (60% and 40%, respectively). The results of the polynomial regression are shown in Figure 2.8 (a) which is training data results and (b) which is testing data results.

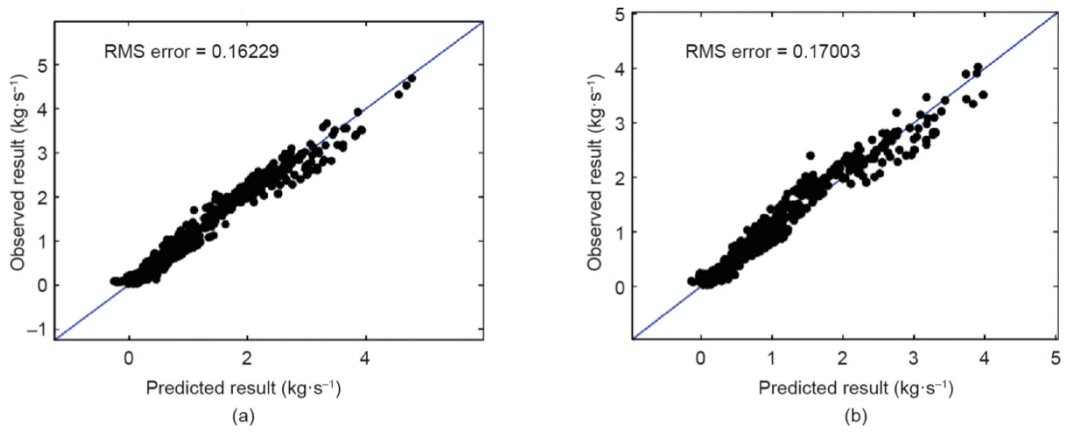


Figure 2.8:Result from predicting fuel flow given the rated thrust and the out-of-production status. (a) Training data results; (b) testing data results. RMS: root mean square [70].

The rated thrust of the JT9D-3 engine is approximately 193.5 kN and the engine is currently out of production. These two variables were inputs into the polynomial regression model to predict the fuel flow and emissions. The results of such predictions are shown in Figure 2.9 (a) and (b). The fuel flow rate f_{fuel} in Figure 2.9 (a) is calculated using a polynomial equation with

the coefficients shown in the equation below. The carbon monoxide (CO) emission index f_{CO} in Figure 2.9 (b) is calculated as below.

$$f_{fuel} = -0.0308 + 0.0120thrust + (-1.2314) \times 10^{-5}thrust^2 + 9.6211 \times 10^{-9}thrust^3 \quad (2.22)$$

$$f_{CO} = 28.2743 + (-0.5024)thrust + 0.0025thrust^2 + (-3.5476) \times 10^{-6}thrust^3 \quad (2.23)$$

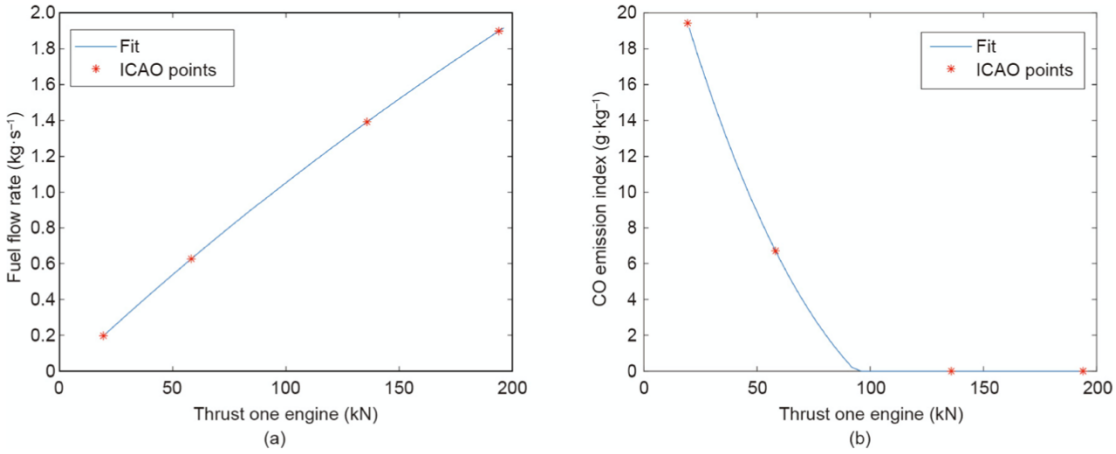


Figure 2.9:Prediction for the JT9D-3 engine utilized in the Boeing 747-100 aircraft model. (a) Fuel flow; (b) carbon monoxide (CO) emission [70].

2.6.3 Undercarriage Forces and Moments

The undercarriage equations as well the data for calculating the moments and force are given in [43, 44]. A tricycle model is used for the aircraft gears. It is assumed that the aircraft is having three gears in the tricycle model and each gear is modelled as non-linear oleo-strut. There are two main gears (left and right) and one nose gear. The main gears apply braking while the nose gear is utilised for steering [70].

As the aircraft moves on the ground, the complex interaction between the ground, tires, and oleo-strut create forces and moments that affect the aircraft motion. The forces and moments due to the undercarriage are each calculated in the local axis of the gear and are each transformed into the body axis. As always, these forces and moments are dependent on the aircraft states. The steps by which these body forces and moments due to the undercarriage are calculated are as follows:

a) Calculate the Oleo Strut Compression and Rates: for main gear, right gear, left gear and nose gear, Figure 2.10 shows how the compression and compression rates are derived.

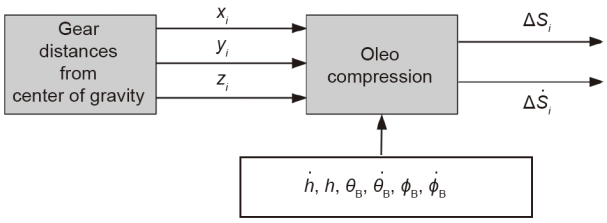


Figure 2.10: Determination of the oleo strut compression and compression rates [70].

$\Delta \dot{S}_{T_i}$ and ΔS_{T_i} represent the oleo strut compression rates and oleo strut compression respectively, at the *i*th gear for *i*=1, 2, 3 (tricycle model assumed). x_i, y_i, and z_i are the distances of the *i*th gear to the centre of gravity in body axes. ϕ_B and θ_B are the body bank angle and the pitch angle respectively [70].

b) Calculate Vertical Force: Once the oleo-strut compression and rates are determined, the three components of the forces (each in the local frame of the gears) are calculated. The first component involves calculating the vertical force F_z. The vertical force carries the weight of the aircraft and provides comfort for the passengers during taxiing. Calculating the vertical force in each gear involves determining two forces per gear (the damping force and the spring force) because the gear is represented by a nonlinear mass–damper–spring system. The spring force can be obtained from [43, 44].

As in the case of the engine data used for calculating the static EPR, a polynomial fit was performed on the undercarriage data as obtained from the NASA reports [43, 44]

(for both the spring force and the damping constants for the main and nose gears). Particularly, strategic points (shown as red points in Figure 2.11, Figure 2.12, Figure 2.13 and Figure 2.14) were selected from the NASA data and a third order polynomial regression was performed. The polynomial coefficients obtained are shown in Table 2.6.

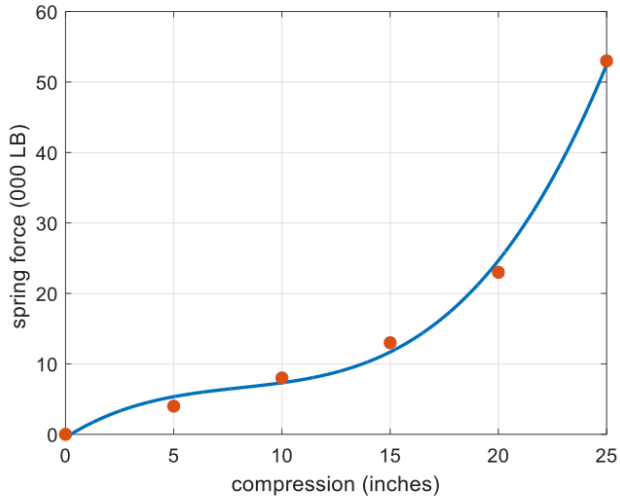


Figure 2.11:Spring force as a function of compression for the nose gear [72].

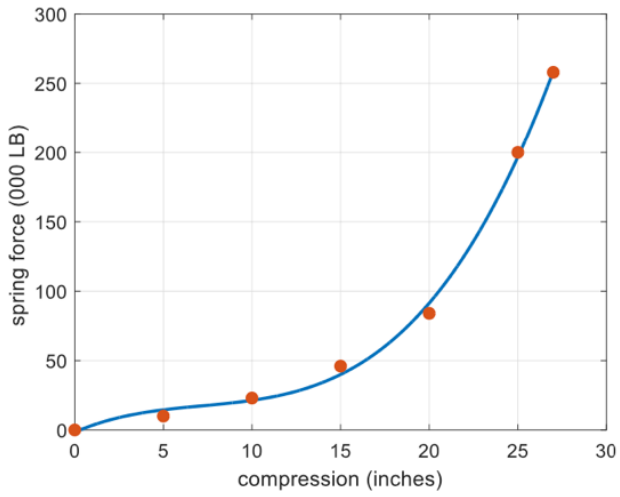


Figure 2.12:Spring force as a function of compression for the left and right main gears [72].

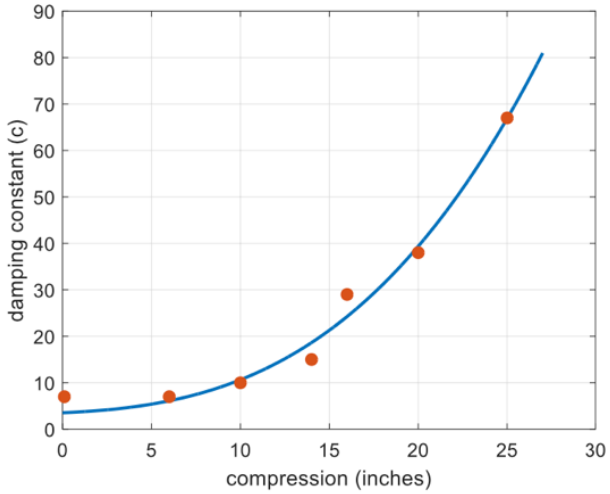


Figure 2.13:Damping constants as a function of compression for the nose gear [72].

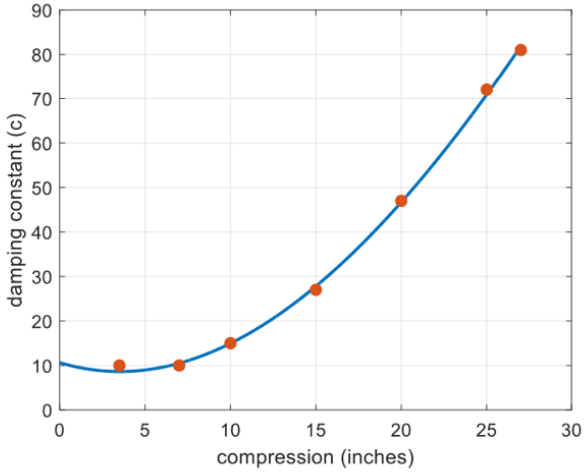


Figure 2.14:Damping constants as a function of compression for the left and right main gears [72].

Table 2.6:Polynomial coefficients for determining the landing gear spring force and damping constants given the oleo deflection [72].

	ao	a1	a2	a3
Nose (spring)	-468.25	1960.1	-201.11	8.2963
Left/Right Main (spring)	-1621.5	5557.8	-606.88	28.078
Nose (c)	3.5474	0.1644	0.027692	0.0026823
Left/Right Main (c)	10.639	-1.1494	0.16872	-0.0010539

c) **Calculate the side force:** The aircraft taxiing movement requires a series of maneuvers that include turning. The turning motion is a consequence of the side forces acting from different components of the aircraft. Calculating the landing gear side force involves a series of steps. The first step involves determining the tire deflection variable (δ_{t_i}) based on the vertical force acting on the tire. This vertical force acting on the tire will be dependent on the attitude of the aircraft. The next step involves calculating the total side force. The total side force is the product of the angle between the tire and the direction of motion and the tire deflection constant [70]. The total side force is limited to 60% of the total vertical force acting on the tires.

d) **Calculate the drag force:** The braking forces are result of the drag force which are applied through the main gears only in the Boeing 747 model and are used to decelerate or stop the aircraft during taxiing. The drag force at a particular gear depend on a both frictional forces and braking. The braking is dependent on the aircraft mass, the braking pedal deflection and a constant K_B= 0.263. The braking force maximum amount is dependent on the rolling friction (μ_B= 0.4 for a dry taxiway), the aircraft mass, as well as a maximum braking constant (K_{BM} = 0.834 + 4.167 $*$ μ_B). The frictional force combines two components: break out force (bof) and a constant rolling friction term. The bof is dependent on the ground speed (V_G) of the aircraft and is determined as follows [70]:

$$\text{bof} = \begin{cases} 0.014 - V_G * 0.0028 & \text{if } V_G < 5m/s, \\ 0 & \text{if otherwise.} \end{cases} \tag{2.24}$$

It is worth noting that the deflection of the braking pedal has been normalised to between 0 and 1.

e) **Transform the forces into the body axes:** The forces derived above in each of the gears are then transformed into the body axis so that they can be included with the relevant body forces and moments. It was shown in [44] that the transformation equation is given by the following equation and is only valid for small angles:

$$
\begin{aligned}
F_{x_{2,3}} &= F_{\mu_{2,3}} - F_{G_{2,3}}\theta_B \\
F_{x_1} &= F_{\mu_1} - F_{G_1}\theta_B - F_{s_1}\delta_S \\
F_{y_{1,2,3}} &= F_{s_{1,2,3}} - F_{G_{1,2,3}}\phi_B \\
F_{z_{1,2,3}} &= F_{\mu_{1,2,3}}\theta_B - F_{s_{1,2,3}}\phi_B + F_{G_{1,2,3}}
\end{aligned}
\tag{2.25}
$$

Where ϕ_B and θ_B are the bank angles and body pitch respectively, and the indices 1, 2, and 3 indicate the tire index. F_{G_i}, F_{μ_i} and F_{s_i} are the tire vertical oleo strut force, tire drag force and side force for the *i*th tire (note that *i*th tire is directly linked to the *i*th gear), respectively. δ_S is the nose wheel steering angle [70]. By summing across the three gears we obtain the total undercarriage force which as follows:

$$
\begin{aligned}
F_x &= \sum_{i=1}^{3} F_{x_i} \\
F_y &= \sum_{i=1}^{3} F_{y_i} \\
F_z &= \sum_{i=1}^{3} F_{z_i}
\end{aligned}
\tag{2.26}
$$

The moment equations are given by the following equation:

$$
\begin{aligned}
M_x &= \sum_{i=1}^{3} F_{z_i} Y_{L_i} - F_{Y_i} h_B \\
M_y &= \sum_{i=1}^{3} F_{z_i} X_{L_i} - F_{X_i} h_B \\
M_z &= \sum_{i=1}^{3} F_{Y_i} Y_{L_i} - F_{X_i} Y_{L_i}
\end{aligned}
\tag{2.27}
$$

where X_L and Y_L are the distances from the centre of gravity to the end of the fully extended landing gear. h_B = 17+ ΔS_i and is the vertical distance from the centre of gravity of aircraft to the side force, drag force and normal force created by the tires in

contact with the runway. ΔS_i is oleo compression at the *i*th gear for *i*=1, 2, 3 (tricycle model assumed) [70].

2.6.4 Gravity Model

it is worth noting that there are two approaches by which the forces due to gravity can be incorporated into the aircraft model. The first relates to implicitly including it in the translational acceleration function earlier discussed. The second relates to explicitly stating the gravitational forces and then including such force into the total forces and moments acting in the body frame of the aircraft. The latter has been used in modelling the aircraft. As the gravity forces act directly at the aircraft's centre of gravity, it does not generate any moment. The forces in the $x(F_{gx})$, $y(F_{gy})$, and $z(F_{gz})$, coordinates of the body frame are given by the following equation:

$$\begin{aligned} F_{g_x} &= -mg\sin\theta + \sin gsa \\ F_{g_y} &= -mg\cos\theta\sin\phi \\ F_{g_z} &= -mg\cos\theta \end{aligned} \tag{2.28}$$

where GSA is the ground slope angle and ϕ and θ are the roll angle and the pitch angle respectively. An important consideration involves the way in which the ground slope of the taxiway has been included in the gravity model. If an aircraft is traveling downhill (a positive slope angle), then gas will cause a positive force along the x axis of the aircraft. It should be noted that the banking slope has been neglected.

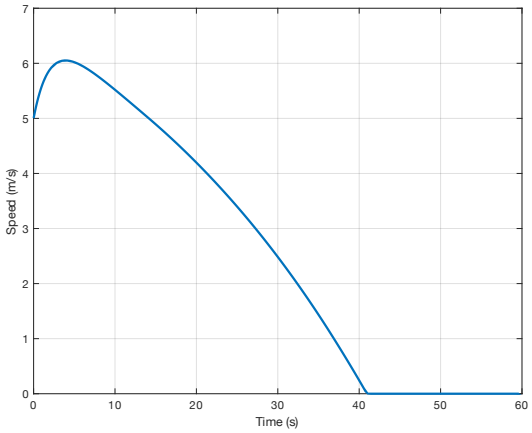
2.7 Simulation Plots

Simulation experiments are conducted in this section and output responses are plotted with respect to changes in state variables and inputs for the open loop model of Boeing 747 aircraft. The duration of simulation is set for 60s. The following types of analysis are conducted in this regard:

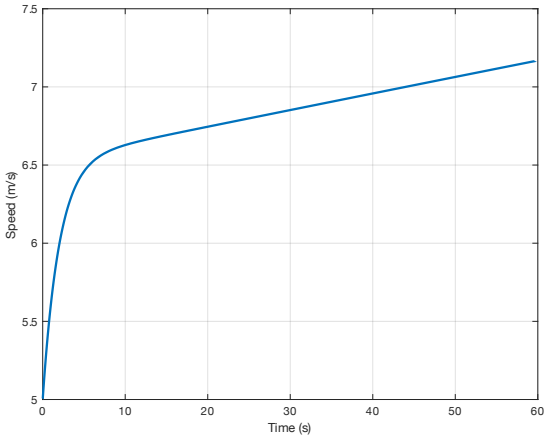
a. Speed/acceleration at varying levels of throttle.
b. Deceleration at varying levels of braking and at varying levels of initial speed.
c. Engine responses to varying levels of initial speed at varying levels of throttle.
d. Yaw at varying levels of rudder.

a. Speed/acceleration at varying levels of throttle

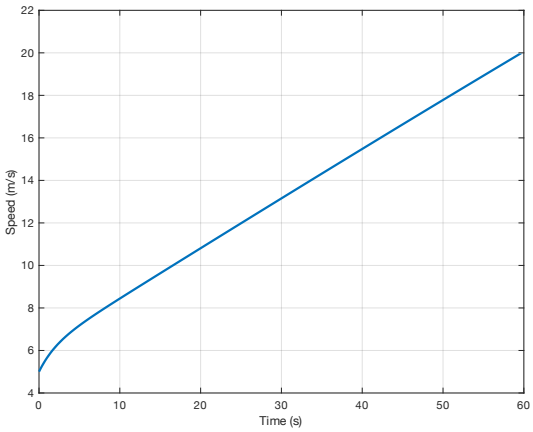
The output of speed profile observed for increasing levels of throttle from [0 1] are plotted as shown in Figure 2.15. The initial speed is set at 5m/s. It can be seen that the rate of change of speed of aircraft increases over time as the throttle value increases [43, 44].

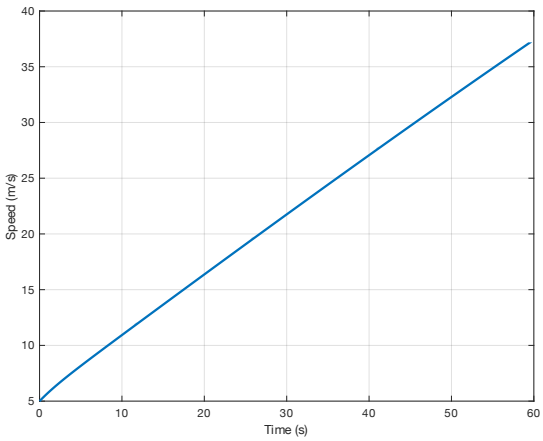


(a) Throttle input = 0

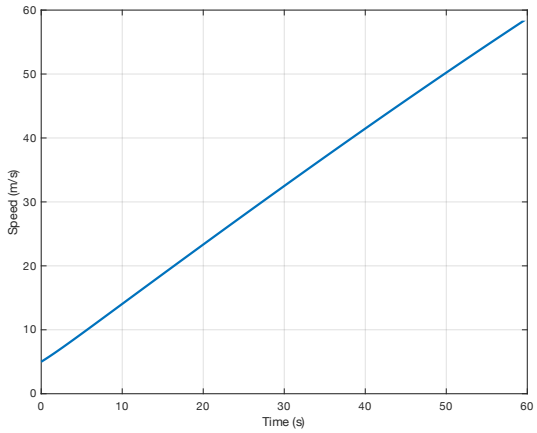


(b) Throttle input = 0.2

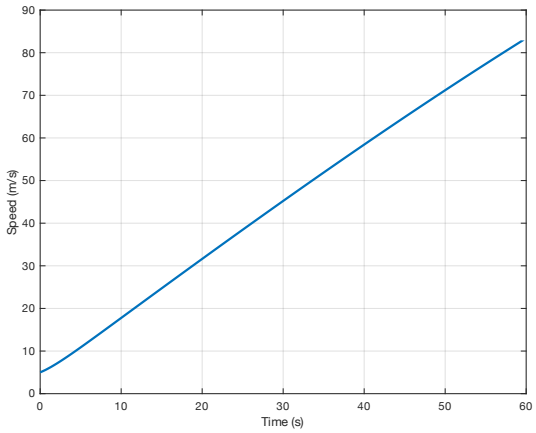




(c) Throttle input = 0.4



(d) Throttle input = 0.6

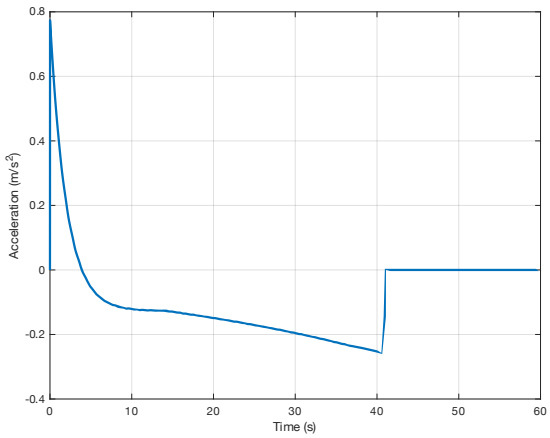


(e) Throttle input = 0.8

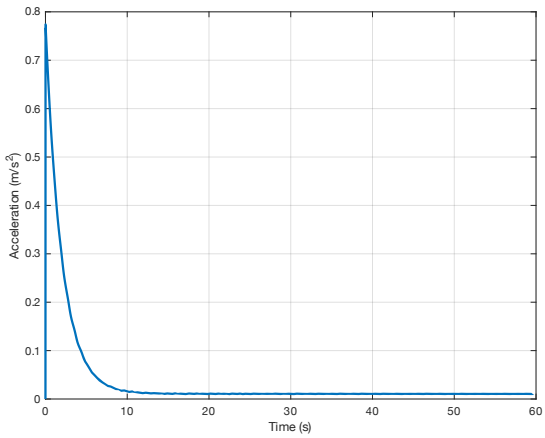
(f) Throttle input = 1

Figure 2.15:Speed profile at varying levels of throttle.

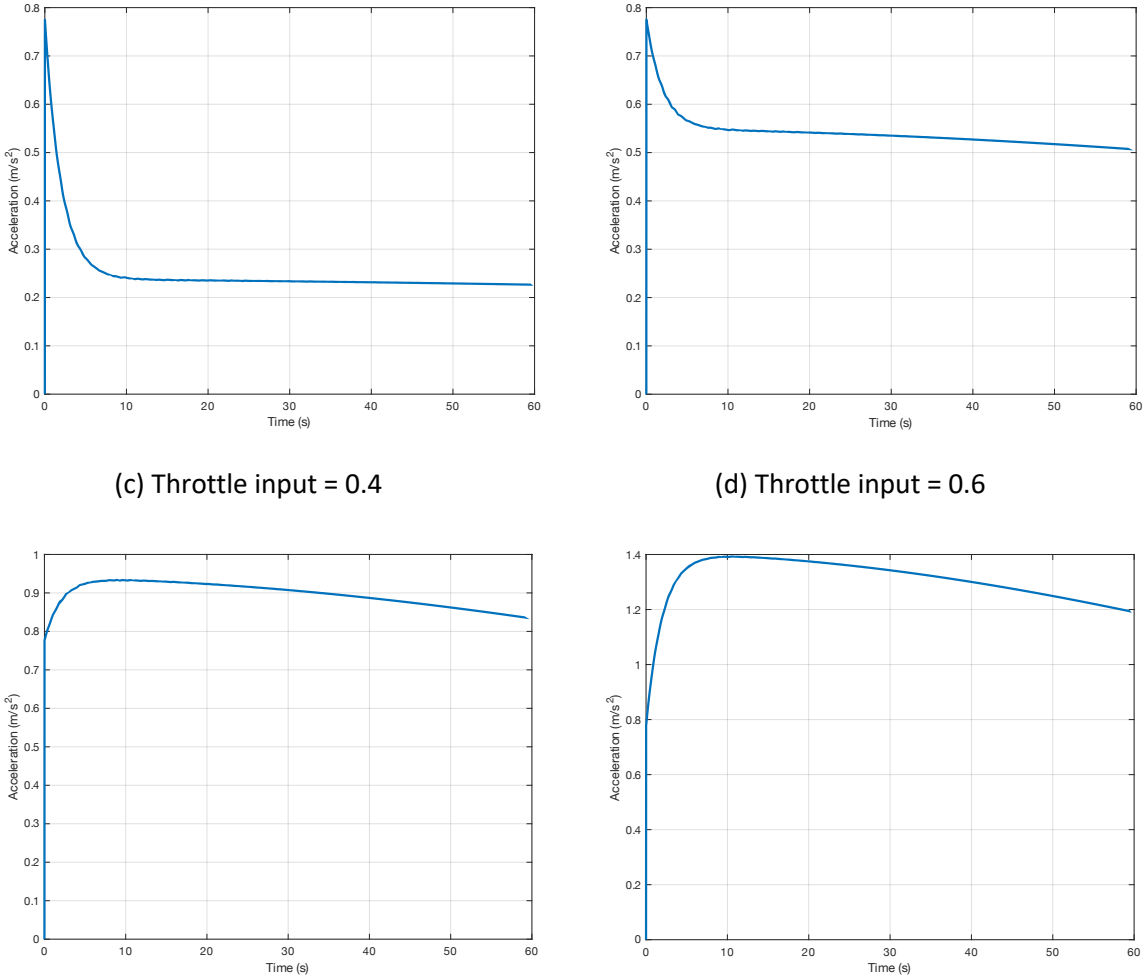
The output of acceleration profile observed for increasing levels of throttle from [0 1] are plotted as shown in Figure 2.16. The initial speed is set at 5m/s. It is noticed that the acceleration increases as the throttle value increases, and then decreases after about 10 seconds from the beginning [43, 44].



(a) Throttle input = 0



(b) Throttle input = 0.2



(c) Throttle input = 0.4

(d) Throttle input = 0.6

(e) Throttle input = 0.8

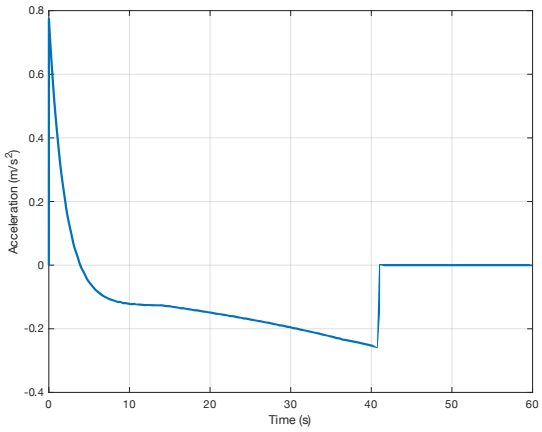
(f) Throttle input = 1

Figure 2.16:Acceleration Profile at varying levels of throttle.

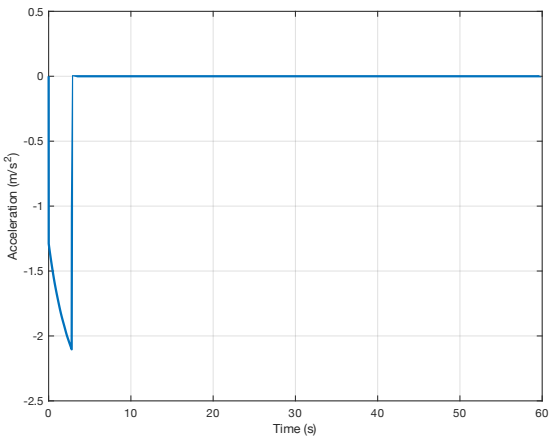
b. Deceleration at varying levels of braking and at varying levels of initial speed

Assuming that there is no throttle input, the deceleration responses for increasing levels of braking varying from [0 1] are plotted considering three different levels of initial speed. Figure 2.17, Figure 2.18 and Figure 2.19 present the acceleration plots obtained from which the deceleration (negative acceleration) can be observed for initial speeds set at 5, 10, and 15 m/s, respectively. It can be seen that the deceleration decreases as the brake value increases and at some point the speed approaches a constant value of zero in all cases, which means that the

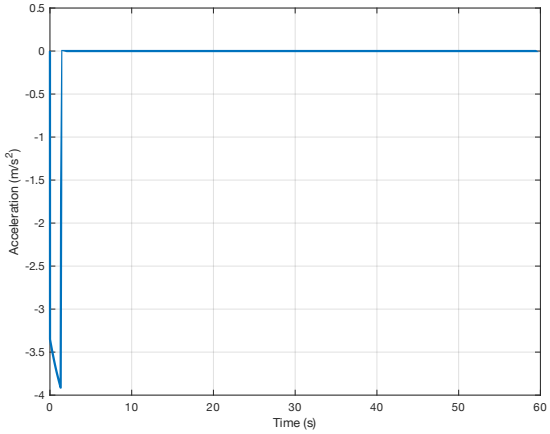
aircraft stop moving. The different initial speeds have not a big effect on the deceleration value [43, 44].



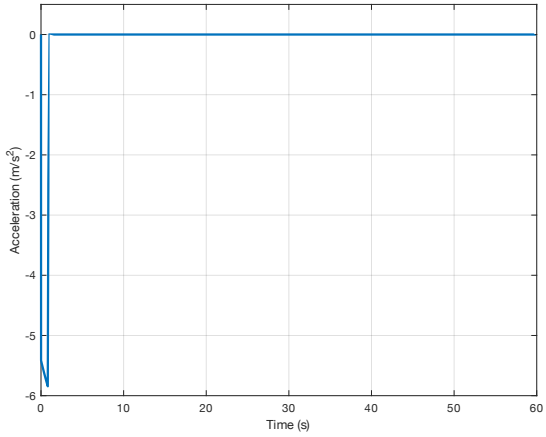
(a) Breaking input = 0



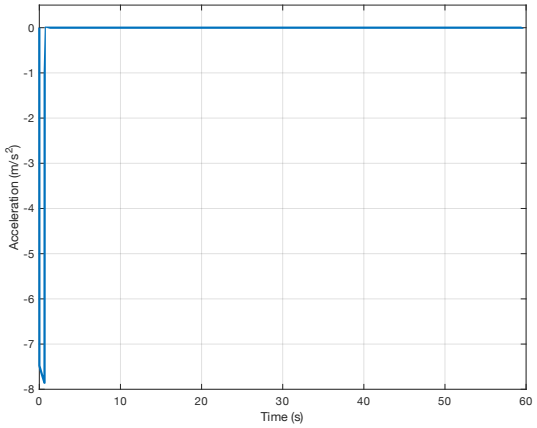
(b) Breaking input = 0.2

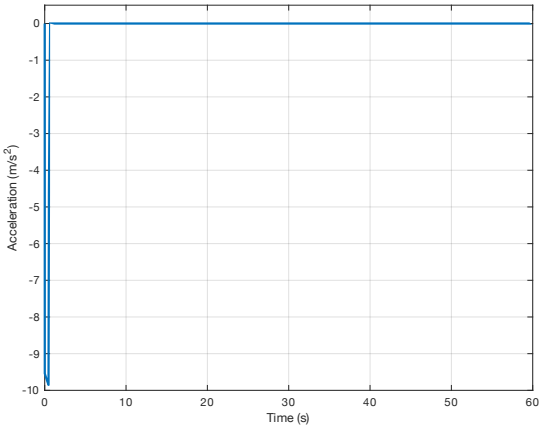


(c) Breaking input = 0.4



(d) Breaking input = 0.6

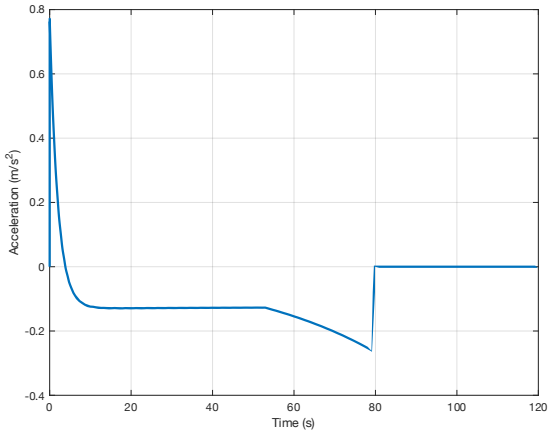




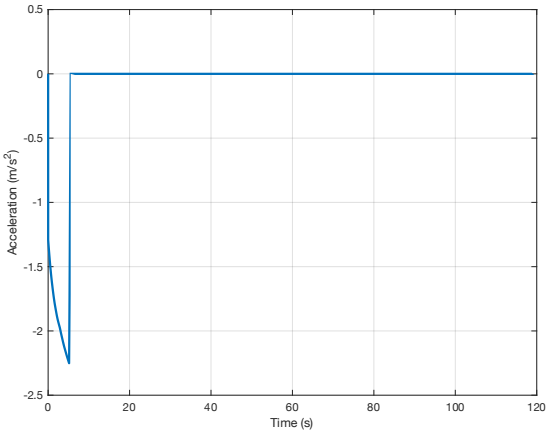
(e) Breaking input = 0.8

(f) Breaking input = 1

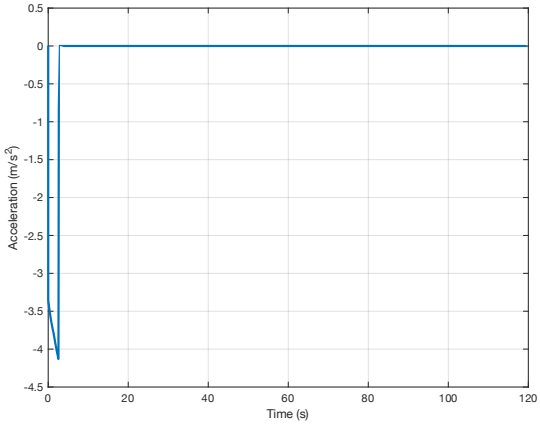
Figure 2.17:Deceleration at initial speed = 5 m/s



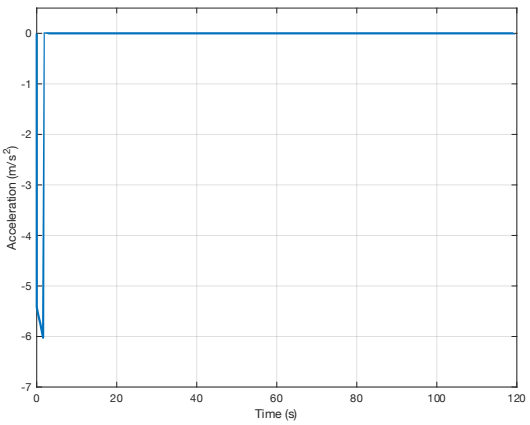
(a) Breaking input = 0



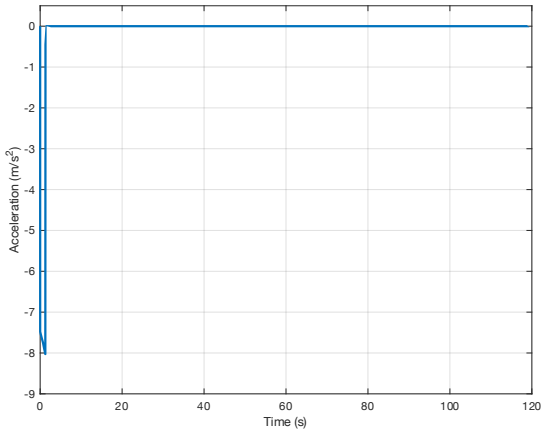
(b) Breaking input = 0.2



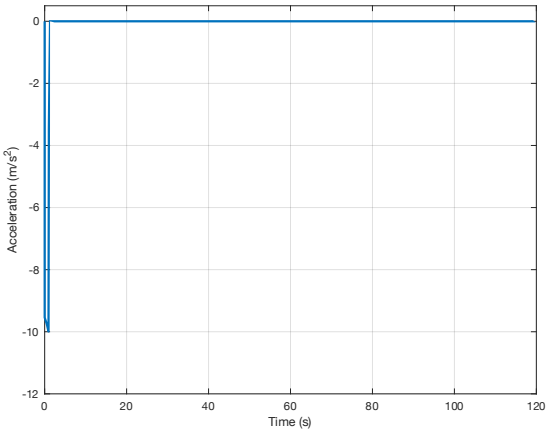
(c) Breaking input = 0.4



(d) Breaking input = 0.6

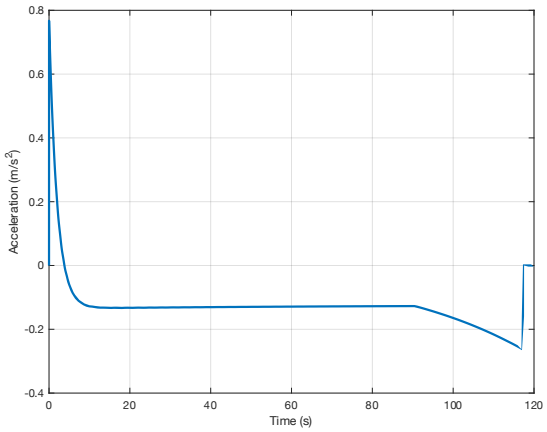


(e) Breaking input = 0.8

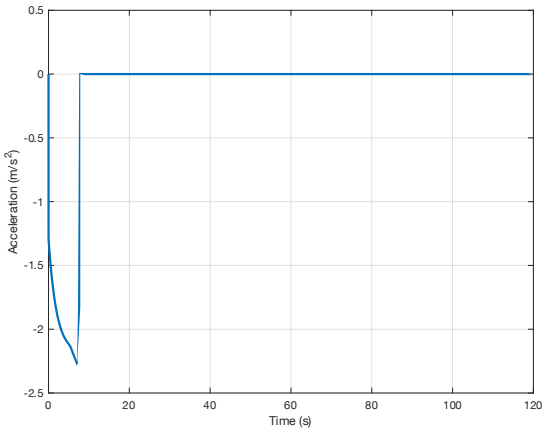


(f) Breaking input = 1

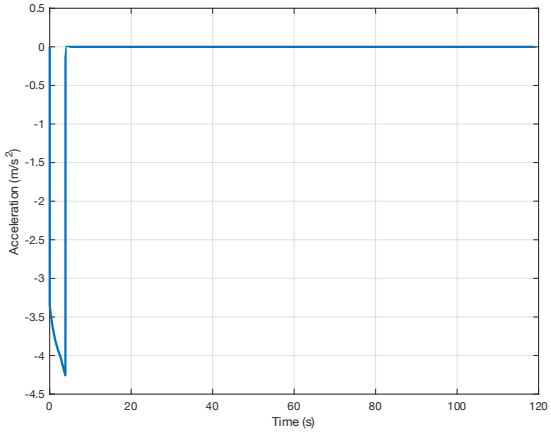
Figure 2.18:Deceleration at initial speed = 10 m/s

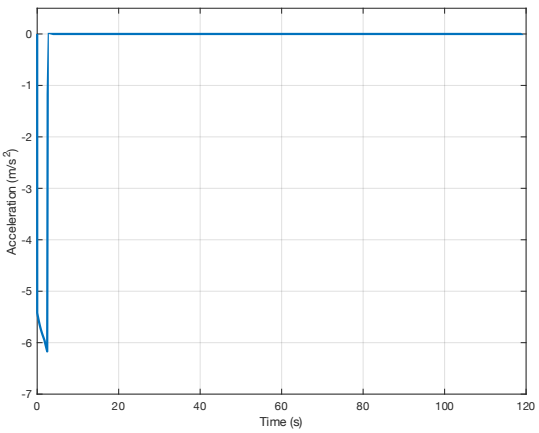


(a) Breaking input = 0

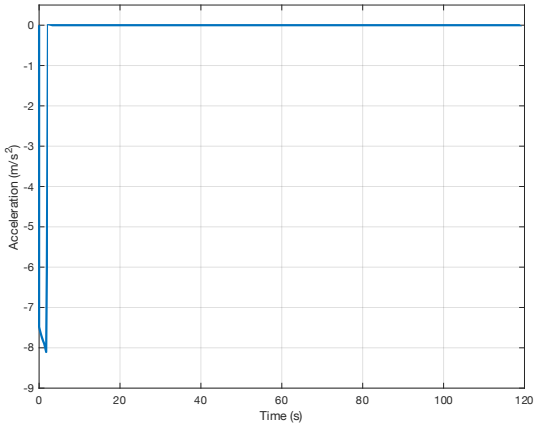


(b) Breaking input = 0.2

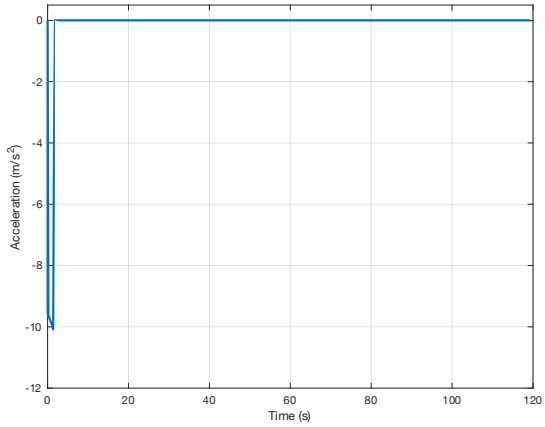




(c) Breaking input = 0.4



(d) Breaking input = 0.6



(e) Breaking input = 0.8

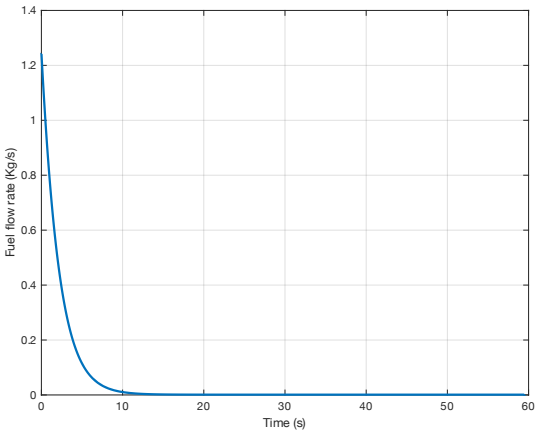
(f) Breaking input = 1

Figure 2.19:Deceleration at initial speed = 15 m/s

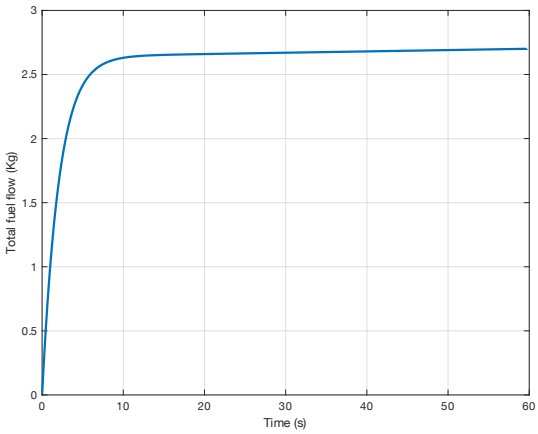
c. Engine responses to varying levels of initial speed at varying levels of throttle

The responses corresponding to emissions, fuel consumption and thrust forces for three initial speed levels (5, 10 and 15 m/s) and three levels of throttle inputs (0, 0.5 and 1) are plotted in Figure 2.20, Figure 2.21,

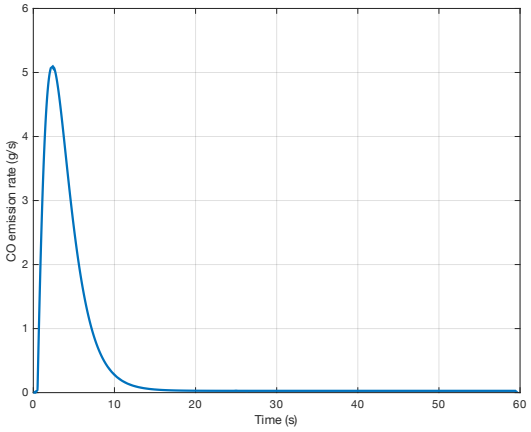
Figure 2.22, Figure 2.23, Figure 2.24, Figure 2.25, Figure 2.26, Figure 2.27 and Figure 2.28. The simulation run time is set to 60s. In all figures it has been noticed that the fuel flow rate and the total fuel flow are increasing when the throttle value increases, also the same thing with the NO emission rate, total NO emission and engine pressure ratio while CO emission rate and total CO emission are increasing when the throttle value increases except when the throttle value is equal to one, the result will be equal to zero. Also it can be seen that the different initial speeds have no effect on all the results [43, 44].



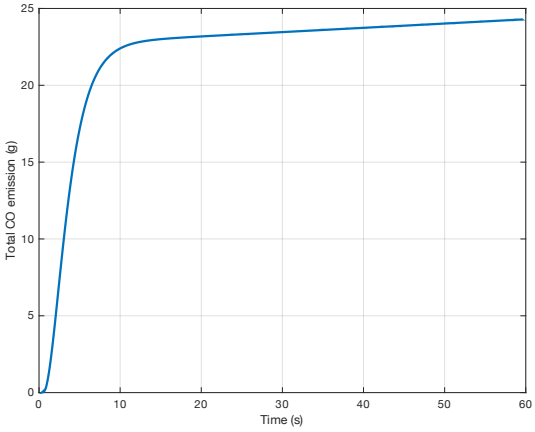
(a) Fuel flow rate at initial speed = 5m/s and throttle input = 0



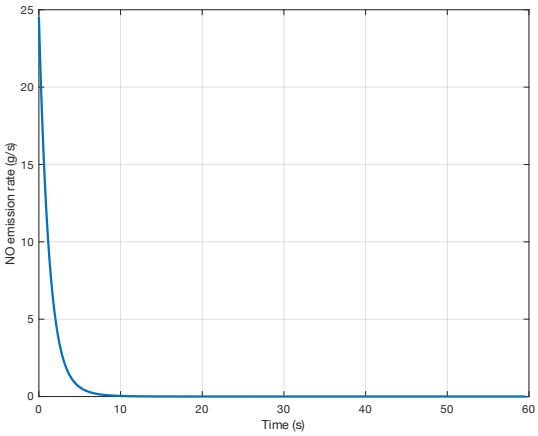
(b) Total fuel flow at initial speed = 5m/s and throttle input = 0

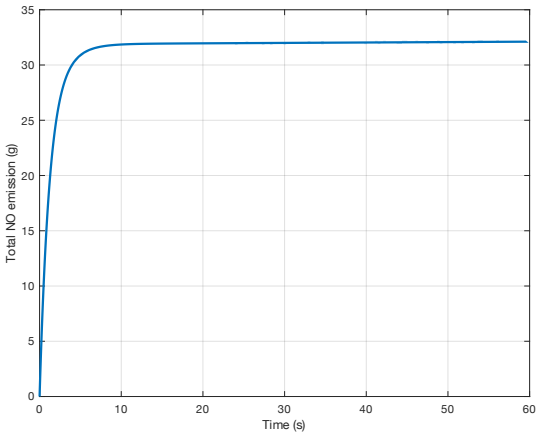


(c) CO emission rate at initial speed = 5m/s and throttle input = 0



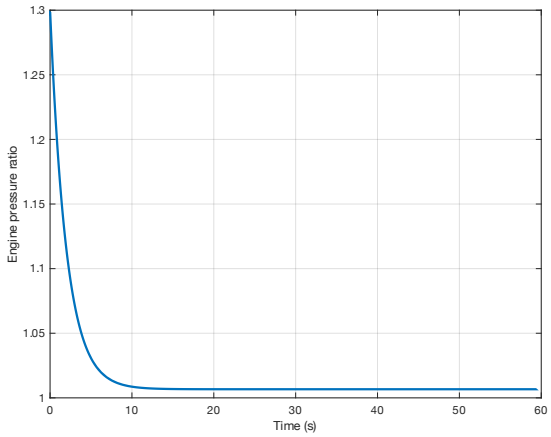
(d) Total CO emission at initial speed = 5m/s and throttle input = 0





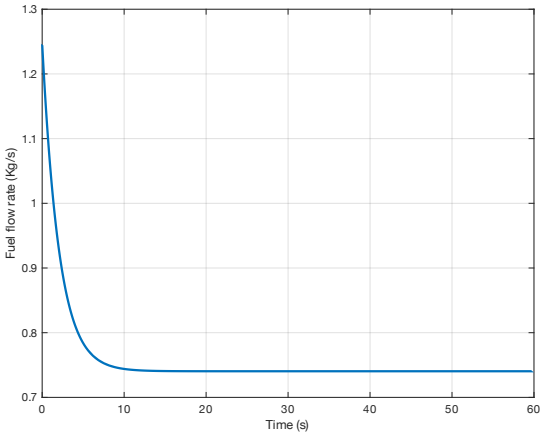
(e) NO emission rate at initial speed = 5m/s and throttle input = 0

(f) Total NO emission at initial speed = 5m/s and throttle input = 0

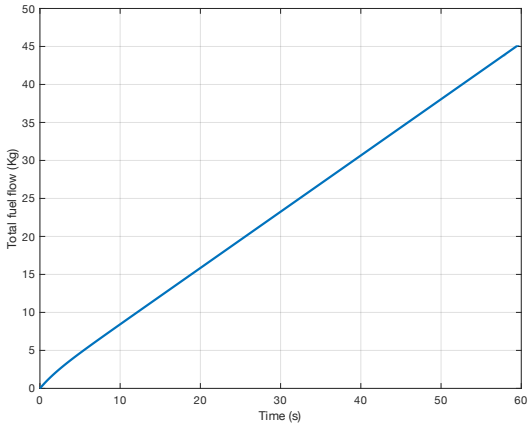


(g) Engine pressure ratio at initial speed = 5m/s and throttle input = 0

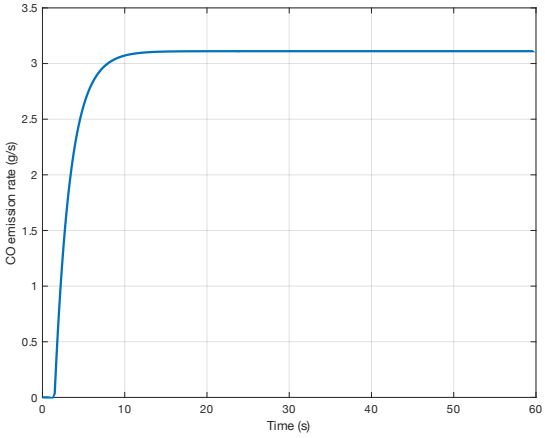
Figure 2.20:Engine responses at initial speed = 5m/s and throttle input = 0



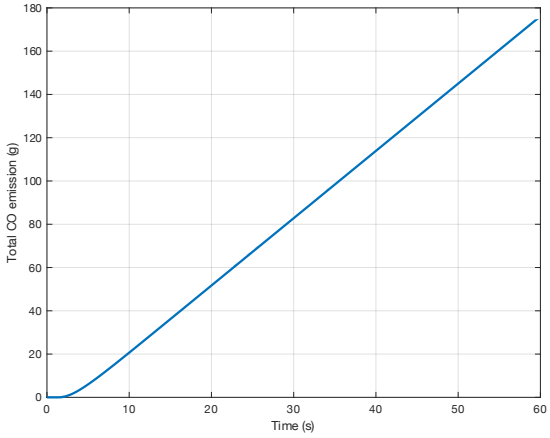
(a) Fuel flow rate at initial speed = 5m/s and throttle input = 0.5



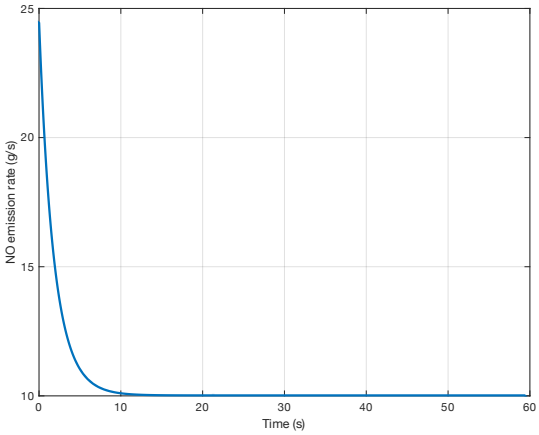
(b) Total fuel flow at initial speed = 5m/s and throttle input = 0.5



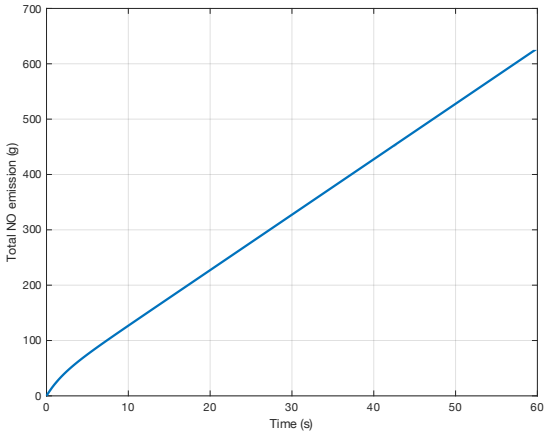
(c) CO emission rate at initial speed = 5m/s and throttle input = 0.5



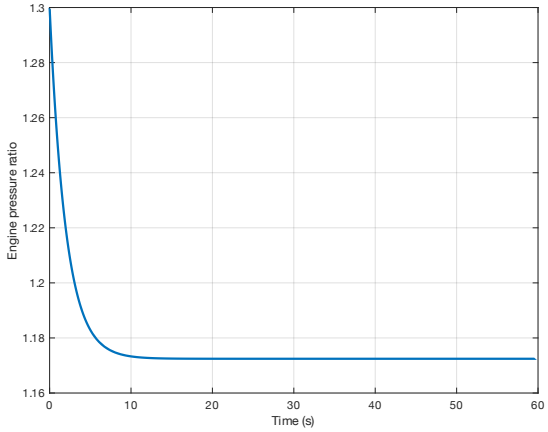
(d) Total CO emission at initial speed = 5m/s and throttle input = 0.5



(e) NO emission rate at initial speed = 5m/s and throttle input = 0.5

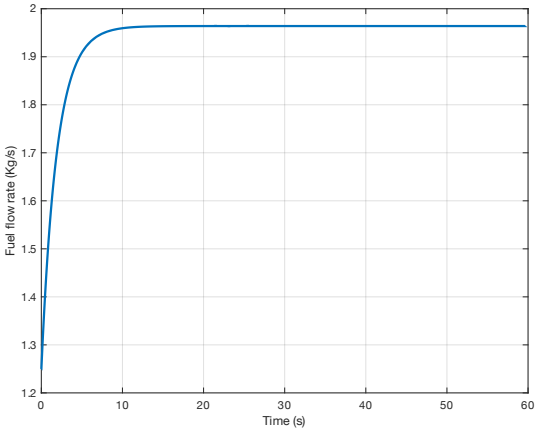


(f) Total NO emission at initial speed = 5m/s and throttle input = 0.5

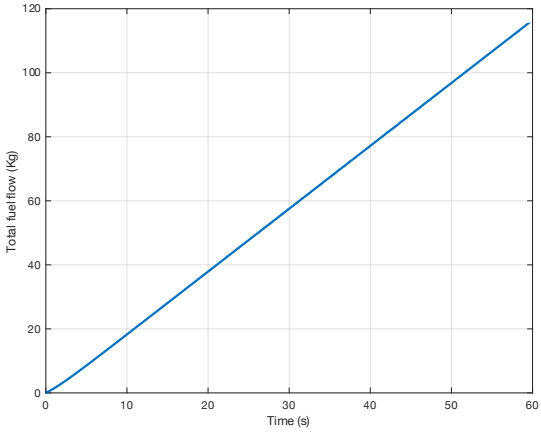


(g) Engine pressure ratio at initial speed = 5m/s and throttle input = 0.5

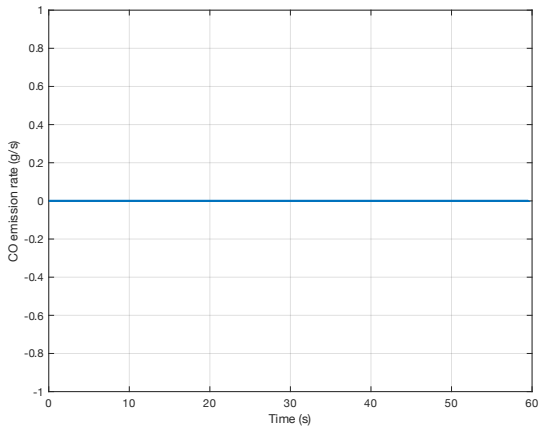
Figure 2.21:Engine responses at initial speed = 5m/s and throttle input = 0.5



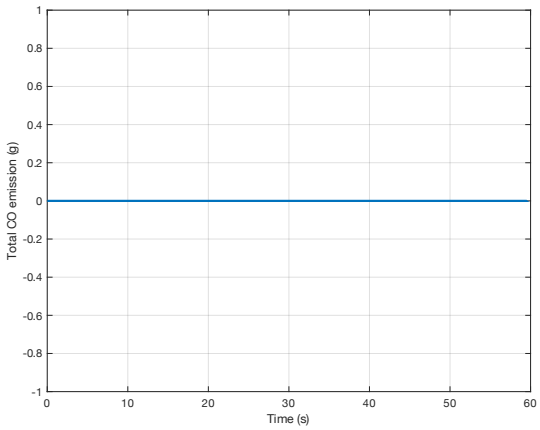
(a) Fuel flow rate at initial speed = 5m/s and throttle input = 1



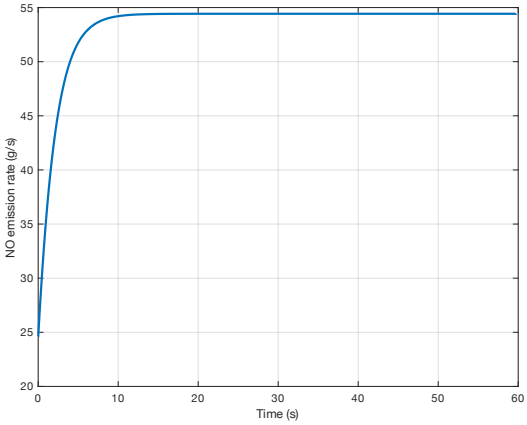
(b) Total fuel flow at initial speed = 5m/s and throttle input = 1



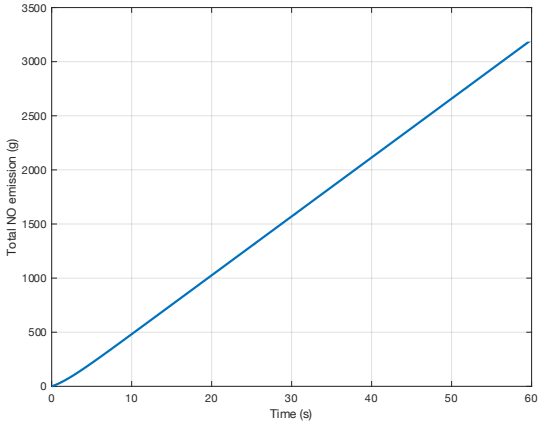
(c) CO emission rate at initial speed = 5m/s and throttle input = 1



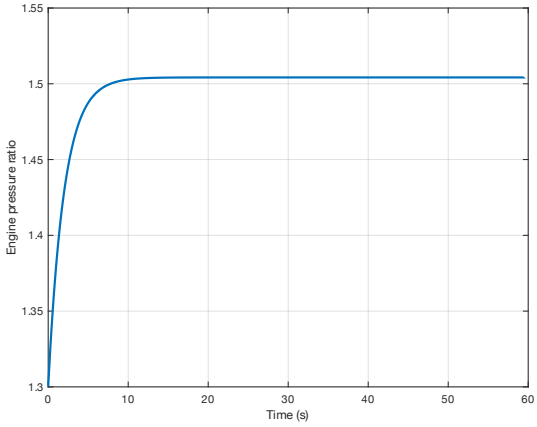
(d) Total CO emission at initial speed = 5m/s and throttle input = 1



(e) NO emission rate at initial speed = 5m/s and throttle input = 1

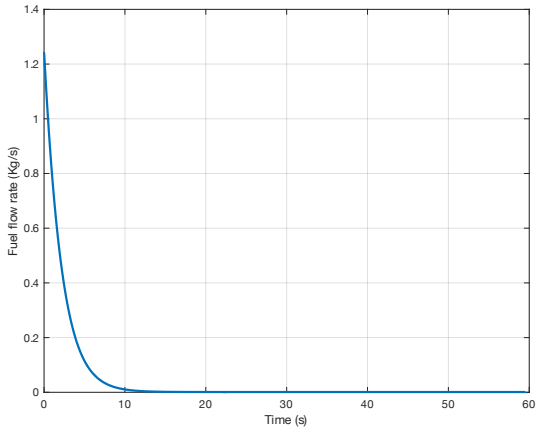


(f) Total NO emission at initial speed = 5m/s and throttle input = 1

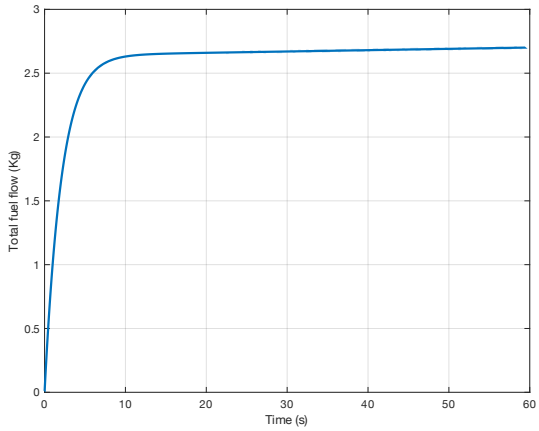


(g) Engine pressure ratio at initial speed = 5m/s and throttle input = 1

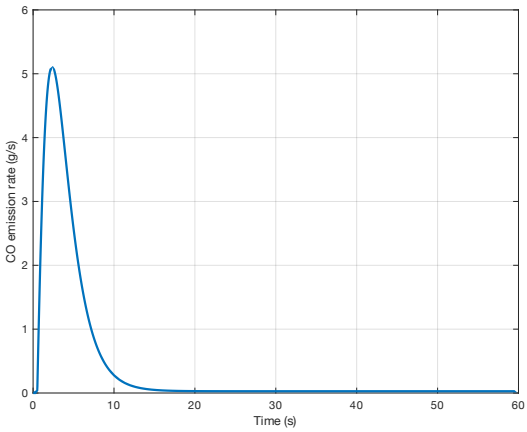
Figure 2.22:Engine responses at initial speed = 5m/s and throttle input = 1



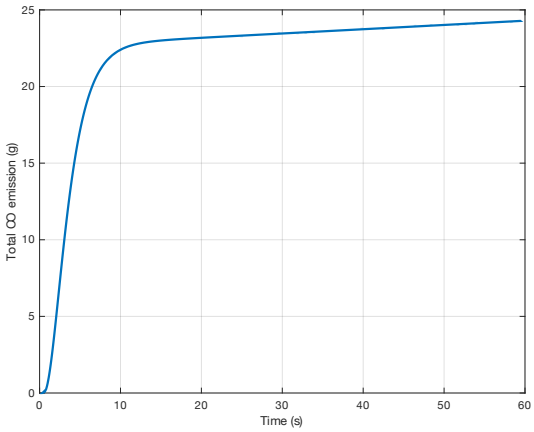
(a) Fuel flow rate at initial speed = 10m/s and throttle input = 0



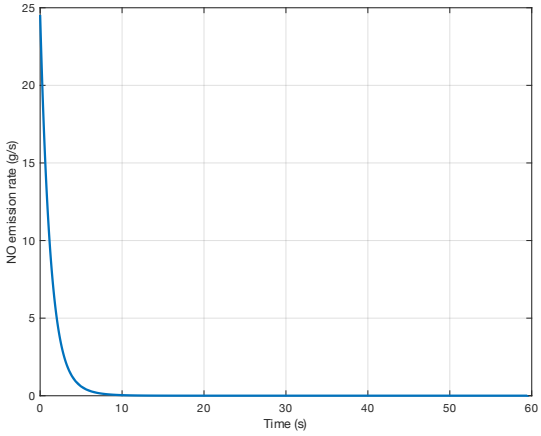
(b) Total fuel flow at initial speed = 10m/s and throttle input = 0



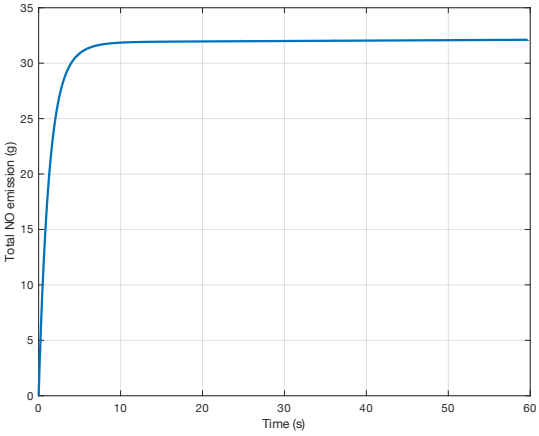
(c) CO emission rate at initial speed = 10m/s and throttle input = 0



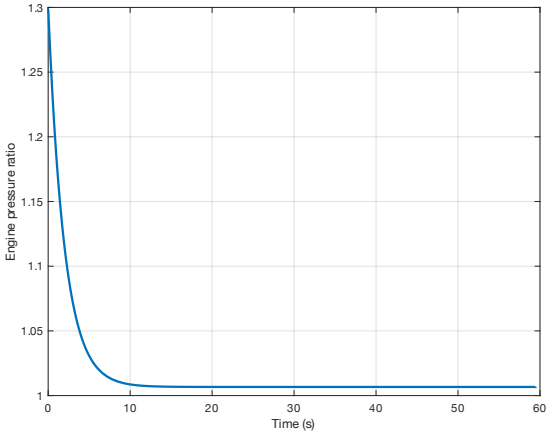
(d) Total CO emission at initial speed = 10m/s and throttle input = 0



(e) NO emission rate at initial speed = 10m/s and throttle input = 0

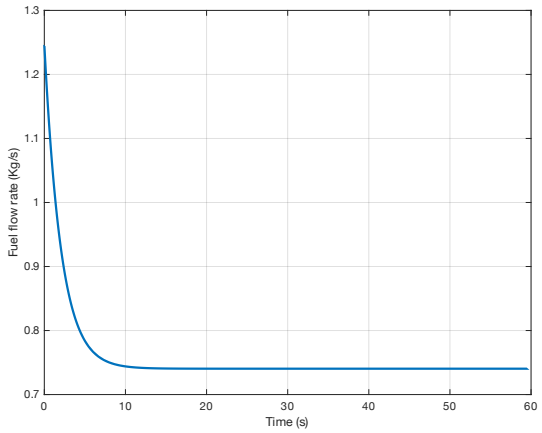


(f) Total NO emission at initial speed = 10m/s and throttle input = 0

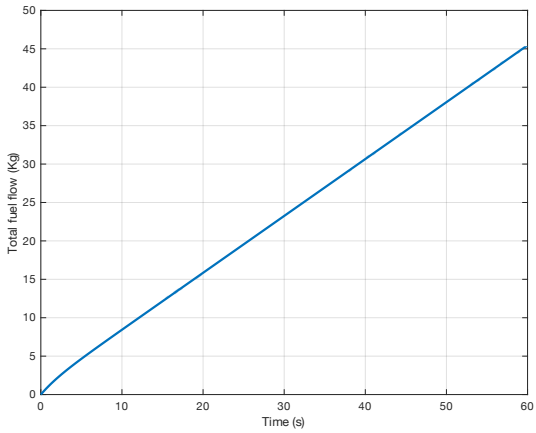


(g) Engine pressure ratio at initial speed = 10m/s and throttle input = 0

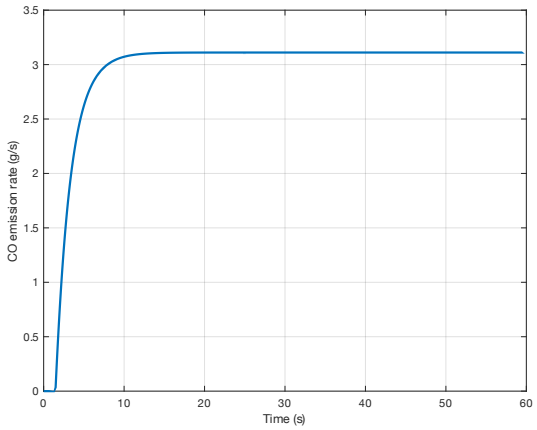
Figure 2.23:Engine responses at initial speed = 10m/s and throttle input = 0



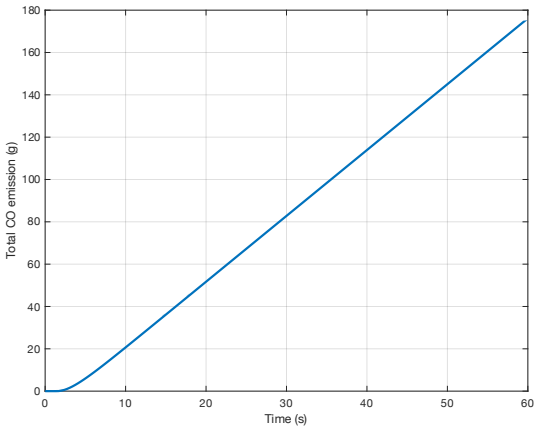
(a) Fuel flow rate at initial speed = 10m/s and throttle input = 0.5



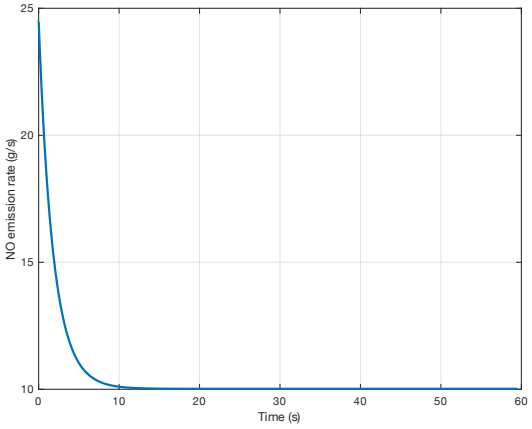
(b) Total fuel flow at initial speed = 10m/s and throttle input = 0.5

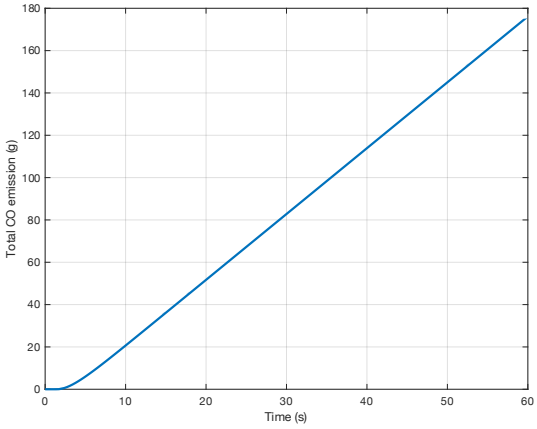


(c) CO emission rate at initial speed = 10m/s and throttle input = 0.5



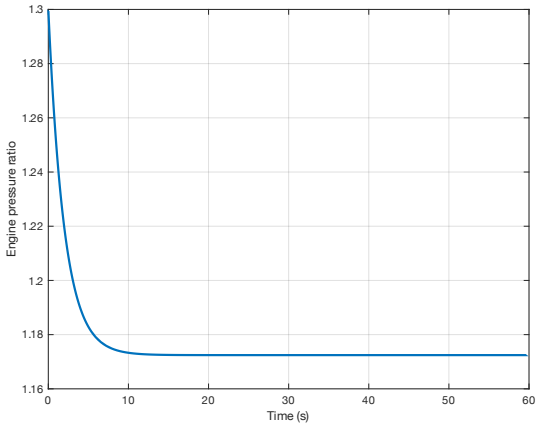
(d) Total CO emission at initial speed = 10m/s and throttle input = 0.5





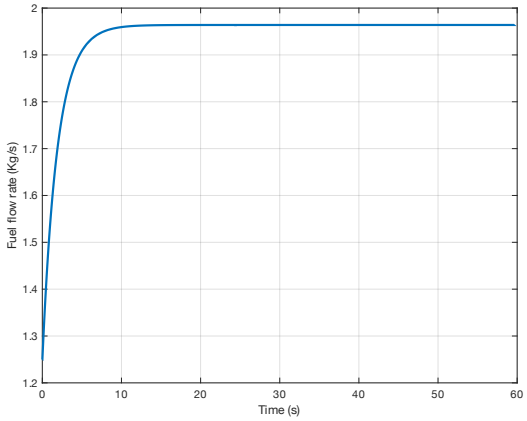
(e) NO emission rate at initial speed = 10m/s and throttle input = 0.5

(f) Total NO emission at initial speed = 10m/s and throttle input = 0.5

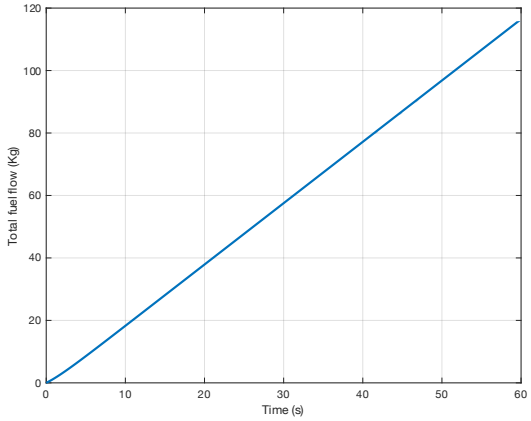


(g) Engine pressure ratio at initial speed = 10m/s and throttle input = 0.5

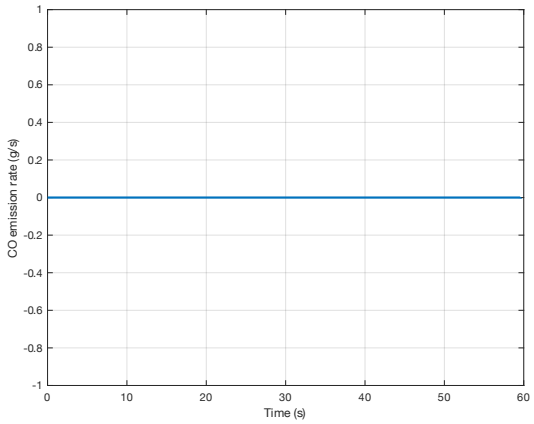
Figure 2.24:Engine responses at initial speed = 10m/s and throttle input = 0.5



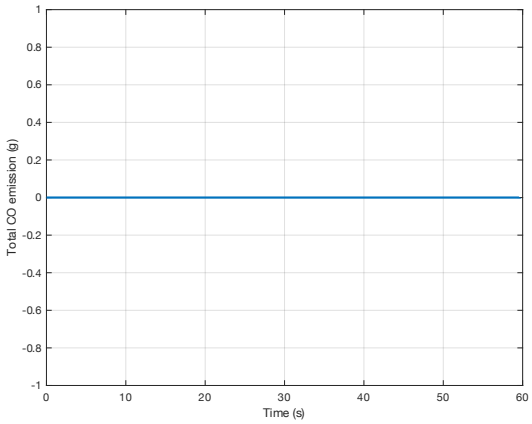
(a) Fuel flow rate at initial speed = 10m/s and throttle input = 1



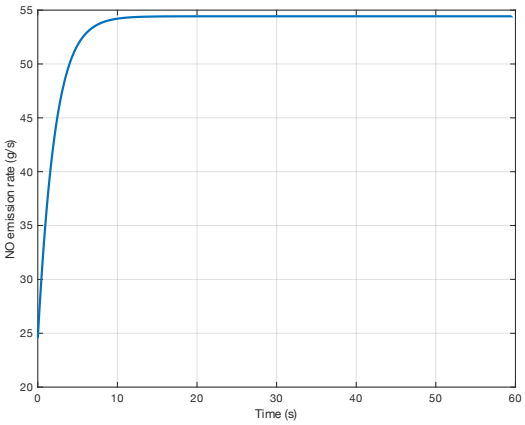
(b) Total fuel flow at initial speed = 10m/s and throttle input = 1



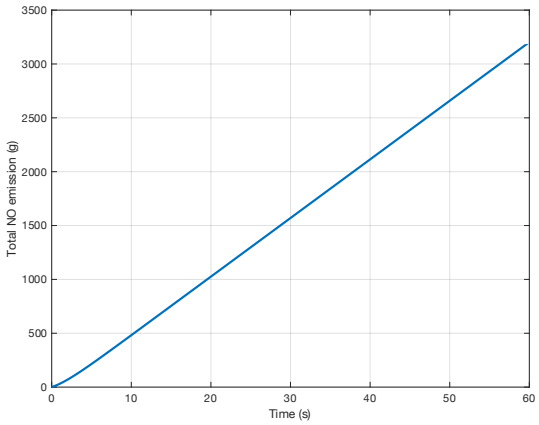
(c) CO emission rate at initial speed = 10m/s and throttle input = 1



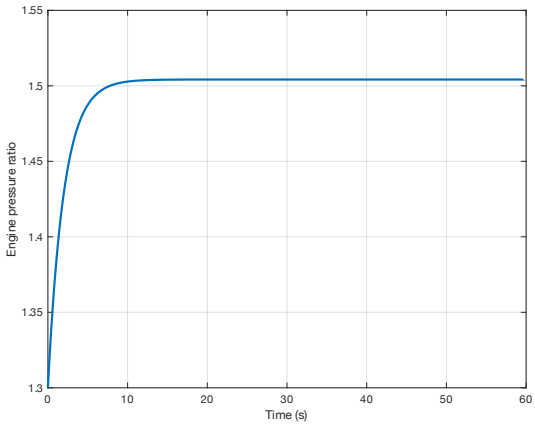
(d) Total CO emission at initial speed = 10m/s and throttle input = 1



(e) NO emission rate at initial speed = 10m/s and throttle input = 1

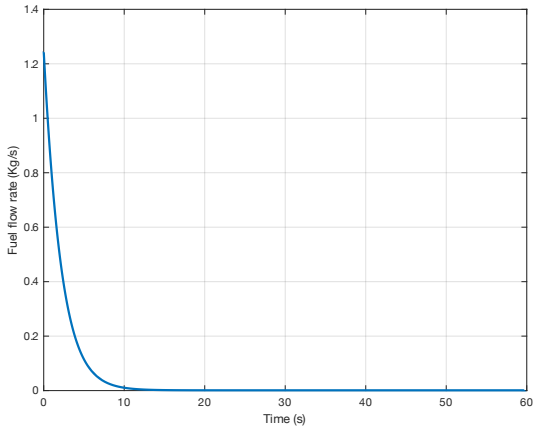


(f) Total NO emission at initial speed = 10m/s and throttle input = 1

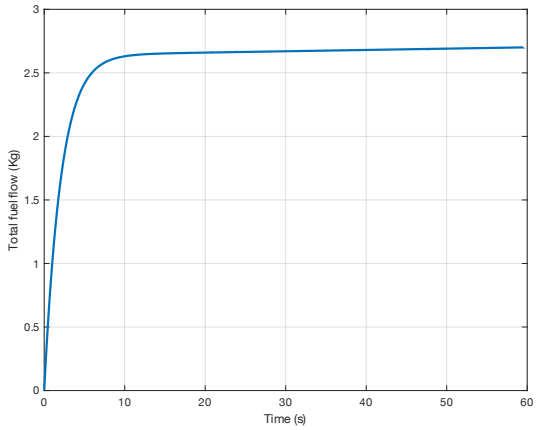


(g) Engine pressure ratio at initial speed = 10m/s and throttle input = 0.5

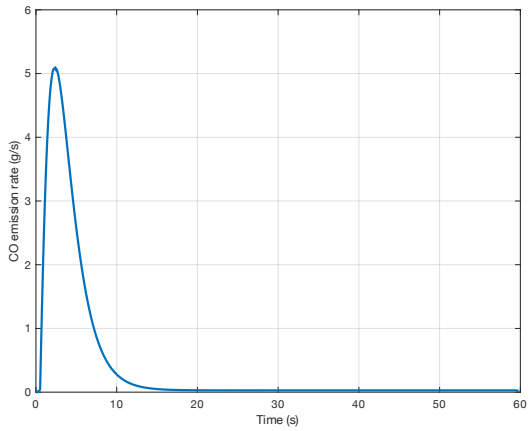
Figure 2.25:Engine responses at initial speed = 10m/s and throttle input = 1



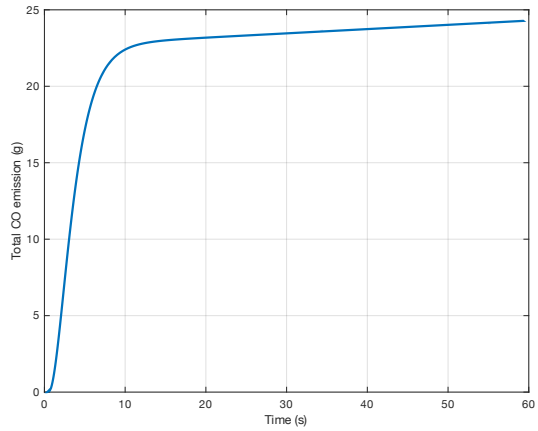
(a) Fuel flow rate at initial speed = 15m/s and throttle input = 0



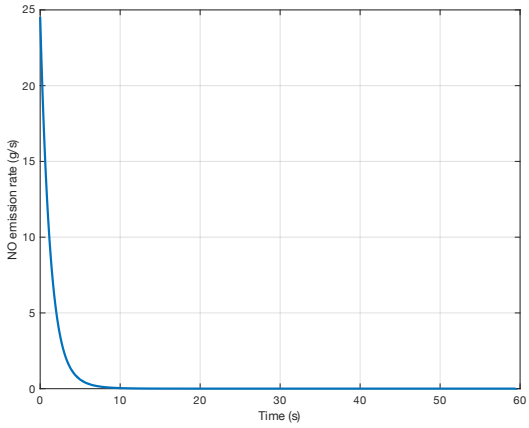
(b) Total fuel flow at initial speed = 15m/s and throttle input = 0



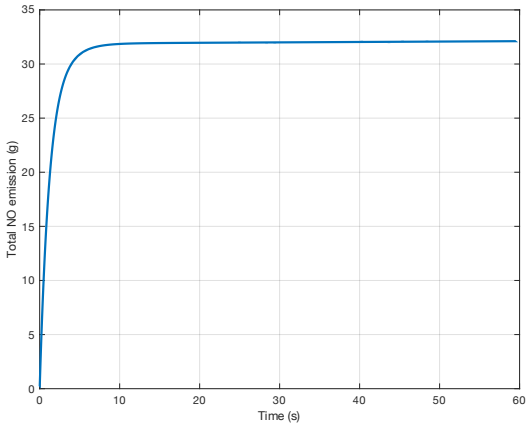
(c) CO emission rate at initial speed = 15m/s and throttle input = 0



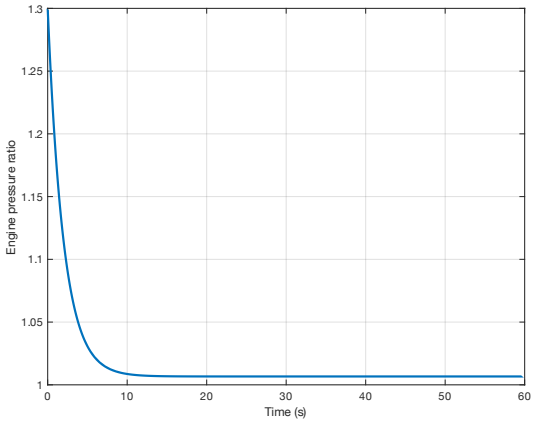
(d) Total CO emission at initial speed = 15m/s and throttle input = 0



(e) NO emission rate at initial speed = 15m/s and throttle input = 0

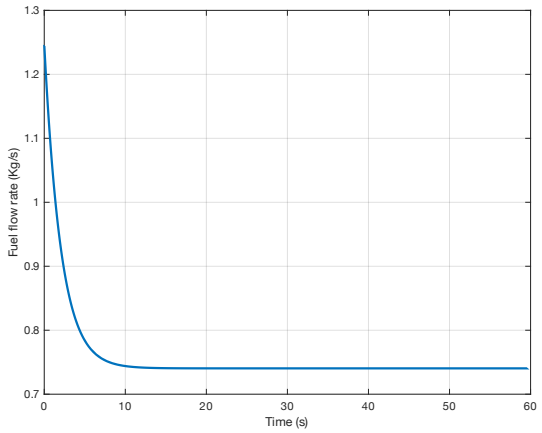


(f) Total NO emission at initial speed = 15m/s and throttle input = 0

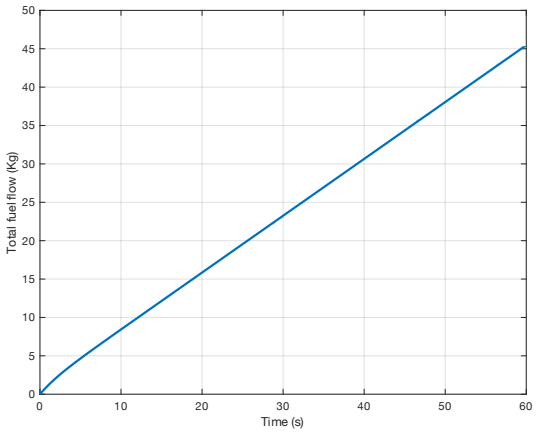


(g) Engine pressure ratio at initial speed = 15m/s and throttle input = 0

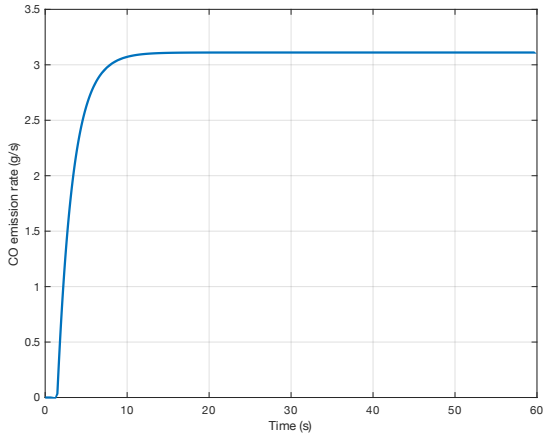
Figure 2.26:Engine responses at initial speed = 15m/s and throttle input = 0



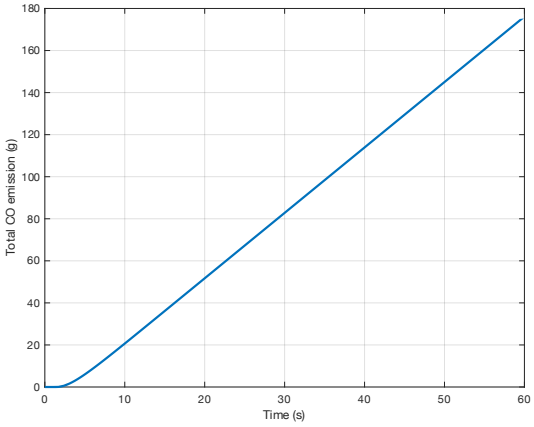
(a) Fuel flow rate at initial speed = 15m/s and throttle input = 0.5



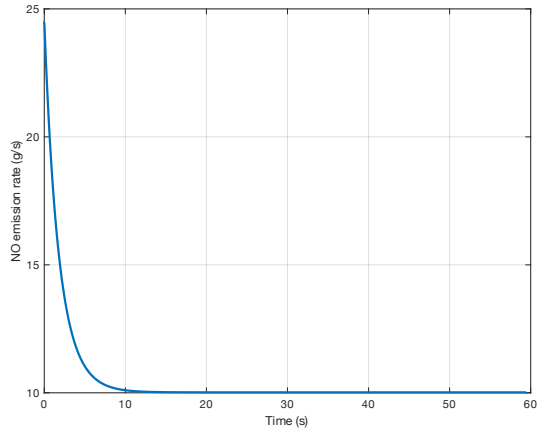
(b) Total fuel flow at initial speed = 15m/s and throttle input = 0.5

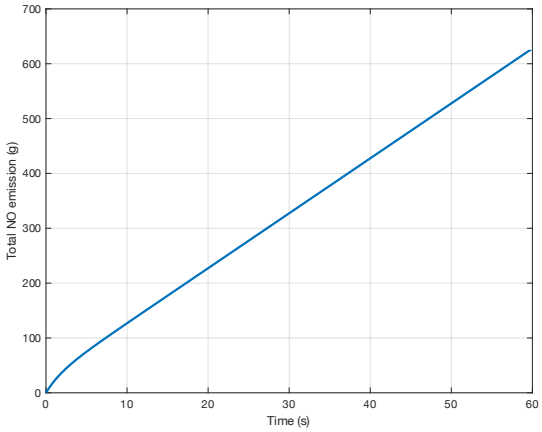


(c) CO emission rate at initial speed = 15m/s and throttle input = 0.5



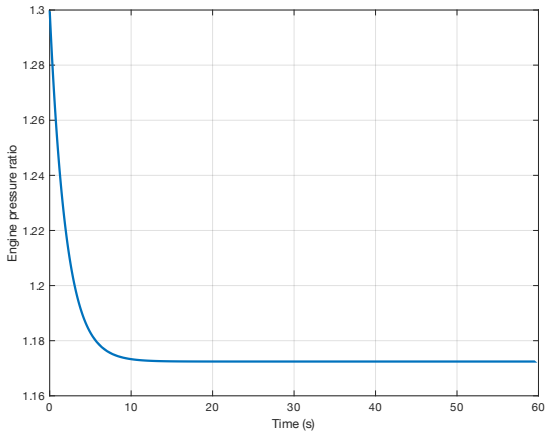
(d) Total CO emission at initial speed = 15m/s and throttle input = 0.5





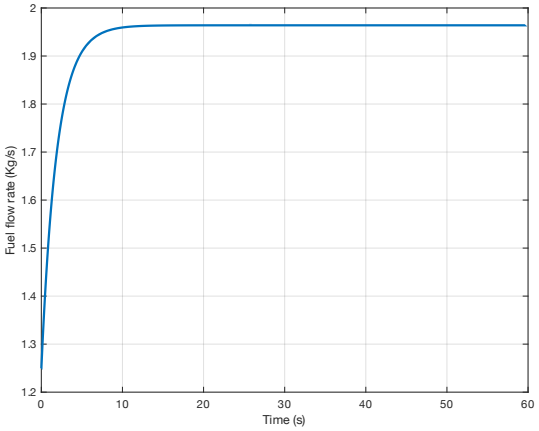
(e) NO emission rate at initial speed = 15m/s and throttle input = 0.5

(f) Total NO emission at initial speed = 15m/s and throttle input = 0.5

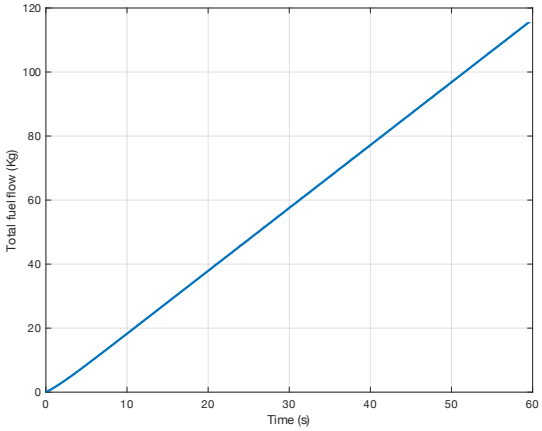


(g) Engine pressure ratio at initial speed = 15m/s and throttle input = 0.5

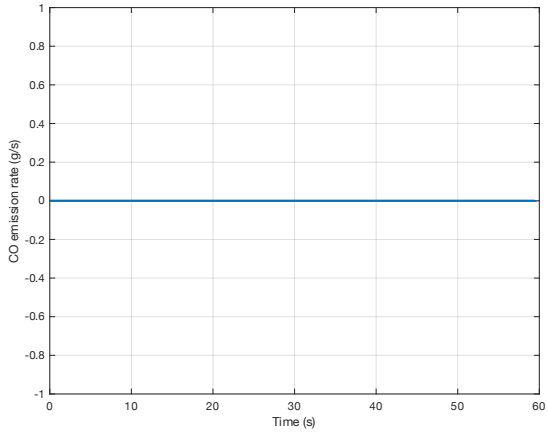
Figure 2.27:Engine responses at initial speed = 15m/s and throttle input = 0.5



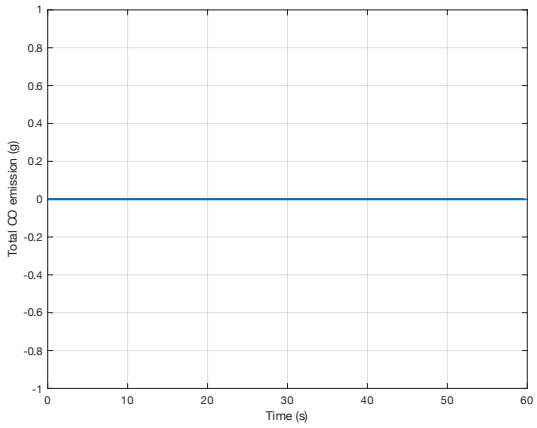
(a) Fuel flow rate at initial speed = 15m/s and throttle input = 1



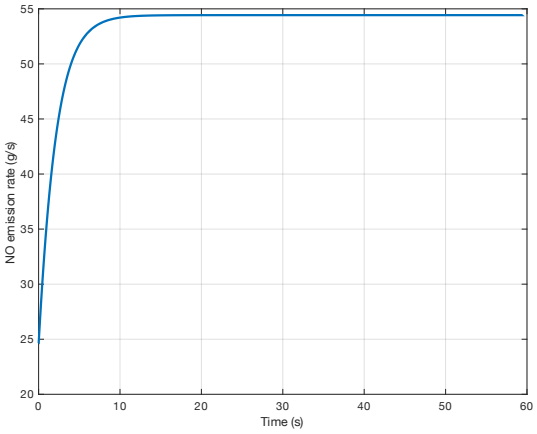
(b) Total fuel flow at initial speed = 15m/s and throttle input = 1



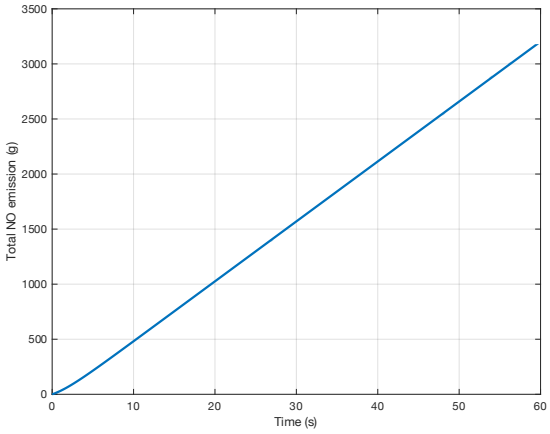
(c) CO emission rate at initial speed = 15m/s and throttle input = 1



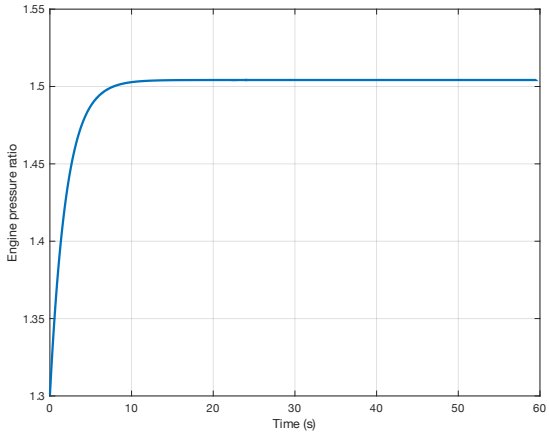
(d) Total CO emission at initial speed = 15m/s and throttle input = 1



(e) NO emission rate at initial speed = 15m/s and throttle input = 1



(f) Total NO emission at initial speed = 15m/s and throttle input = 1



(g) Engine pressure ratio at initial speed =

15m/s and throttle input = 1

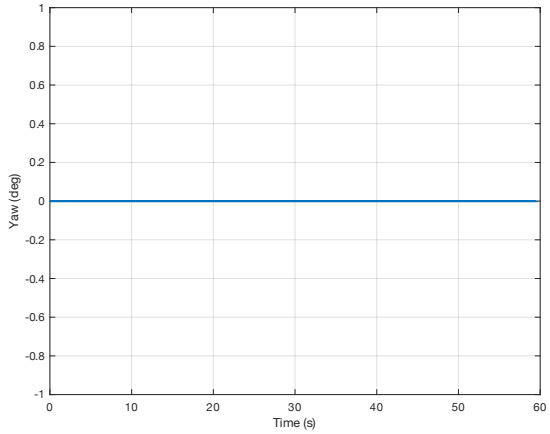
Figure 2.28:Engine responses at initial speed = 15m/s and throttle input = 1

d. Yaw at varying levels of rudder

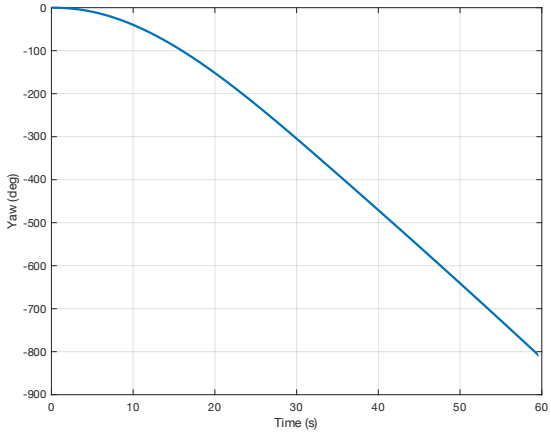
The yaw responses are obtained for different combinations of initial speeds and rudder inputs. Two initial speed levels (5 and 10 m/s) and three levels of rudder inputs (0, 30 and -30) are used in the experiment . The plots of yaw are shown in the Figure 2.29. The configuration of the initial speeds and rudder inputs for the six experiments is shown in Table 2.7. A throttle input of 0.0 is provided for all experiments. It can be seen that when the rudder value is positive, the yaw value is negative, and vice versa. Also, when the initial speed increases, the yaw value increases [43, 44].

Table 2.7:Set of initial speed and Rudder inputs.

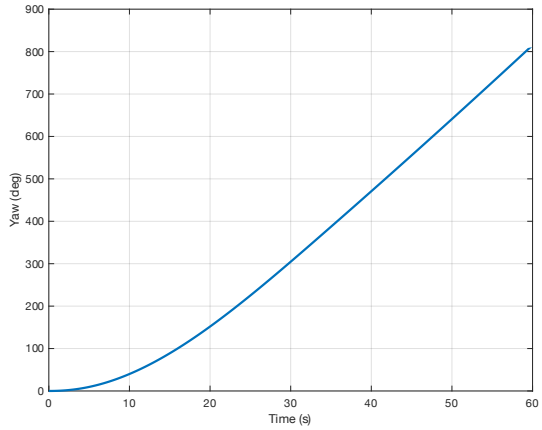
Case	Initial speed (m/s)	Rudder (deg)
1	5	0
2	5	30
3	5	-30
4	10	0
5	10	30
6	10	-30



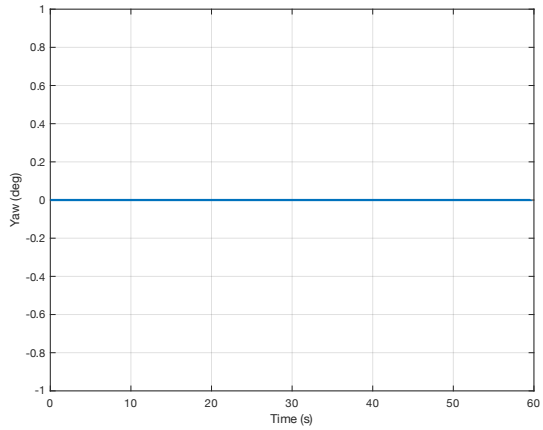
(a) Case 1



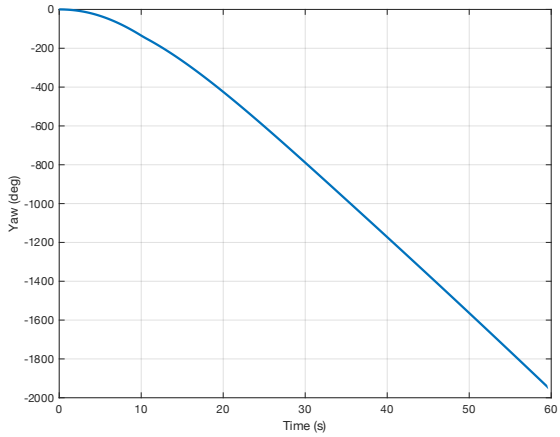
(b) Case 2



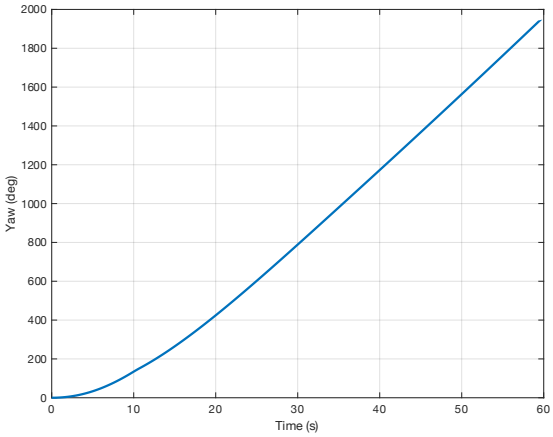
(c) Case 3



(d) Case 4



(e) Case 5



(f) Case 6

Figure 2.29:Yaw profiles obtained for different levels of speed and rudder variations

2.8 Summary

This chapter provided a comprehensive overview of the aircraft model used in this project. The non-linear Boeing 747-100 aircraft was used in this study and the mathematical model is a state-space model formulation derived from the general equation of motion of a rigid body. The three axis system of the aircraft was described which are: the body axis system, the earth reference system and the wind axis system. Thereafter, the aircraft state equations were introduced that includes rotational acceleration function, transitional function, attitude rates, earth relative velocity and pre-thrust rate.

Also, the output equations were farther explained. The five force and moment derivations were then defined and detailed which are: aerodynamic forces and moments, engine forces and moments, undercarriage forces and moments and gravity model. The last part of the chapter conducted some simulation experiments and the output responses are plotted, the following types of analysis are conducted in this regard: speed/acceleration at varying levels of throttle, deceleration at varying levels of braking and at varying levels of initial speed, engine responses to varying levels of initial speed at varying levels of throttle and finally, yaw at varying levels of rudder.

The following chapter will include using a PID controller for aircraft taxiing. Choosing the PID controller is based on its versatility and effectiveness in controlling systems. It is simple to design and implement and offer stability to the controlled system by adjusting the control effort based on the error signal. This helps maintain the system at the desired setpoint. The PID gains will be tuned using Genetic Algorithm (GA) for aircraft taxiing scenarios at a given airport, based on fuel consumed.

3 PID Controller for Aircraft Taxiing

3.1 Introduction

A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback [77]. It is extensively used in industrial control systems and various other applications that require a continuous control model.

An *error value* e(t) is calculated continuously by the PID controller as the difference between a desired setpoint (SP) and a measured process variable (PV) and a correction is applied based on proportional, integral, and derivative terms (denoted *P*, *I*, and *D* respectively) [78].

PID systems automatically apply accurate and responsive correction to a process/plant. Car cruise control is an everyday example, where speed would be reduced. When ascending a hill with applied constant engine power. The PID controller algorithm calculates the measured speed error to the desired speed with minimal delay and then overshoots by increasing the power output of the engine in a controlled manner.

The first theoretical analyses and practical applications of PIDs occurred in the field of automated steering systems on ships developed in the early 20th century [79]. Later, it was used in the manufacturing industry for automatic process control; then, it was widely used in pneumatic and electronic controllers. Today, the concept of PID is widely used in applications that require an optimised and accurate automatic control [80-101].

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. Figure 3.1 shows the principles for generating and applying these terms.

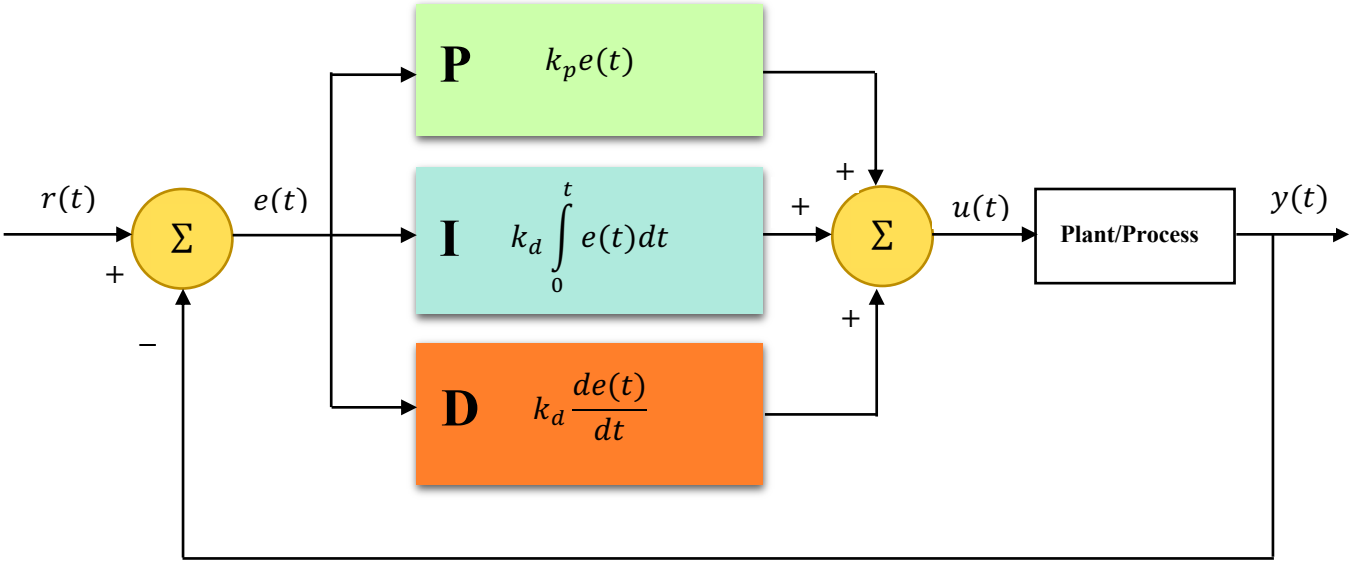


Figure 3.1:A block diagram of a PID controller in a feedback loop. r(t) is the desired process value or setpoint (SP), and y(t) is the measured process value (PV).

The PID controller is shown, an error value e(t) is continuously calculated as the difference between a desired setpoint SP=r(t) and a measured process variable PV=y(t): e(t)=r(t)−y(t), and a correction is applied based on proportional, integral, and derivative terms. The controller is attempting to minimize errors over time by adjusting the control variable u(t), for example, by opening a control valve, a new value is determined by the weighted sum of the control parameters.

Term **P** is proportional to the error e(t) current value. For example, if the error is large, the control output will be proportionally large using the "k_p" gain factor. An error between the process value and the set point will occur if we use proportional control alone because an error is required for generating proportional output response. Under the steady state of the process, the equilibrium is achieved with a steady error "offset".

Term **I** is responsible for the error past values and these values are integrated over time for the **I** term production. The remaining error is eliminated by the integral term if there is a remaining error after applying the proportional control by adding the control effect because of

the historic cumulative error value. The integral term will stop to grow when the error is eliminated. As a result, the proportional effect decreases as the error decreases.

Term **D** is called 'anticipatory control' as it estimates the future trend of the error, based on its current rate of change. It attempts to reduce the error's effect by applying the control result generated by the rate of change in the error. The faster changes are made, the greater the control or dampening effect.

Tuning – The balance of these effects is achieved by loop tuning to produce the optimal control function. The tuning constants are called "K" and must be calculated for each control application, as they depend on the complete loop response external to the controller. These are dependent on the process itself, any delays in control signal, the final control element (such as a control valve) and the measuring sensor behaviour. Generally, it is possible to initially enter approximate values of constants depending on the application type, but they are normally tuned or refined by "bumping" the process in practice by changing the setpoint and observing the response of the system.

Control action – A direct control action is used by the practical loop and mathematical model for all the terms, this means that an increase in positive errors leads to an increase in the correction of the positive output control. The system is called reverse acting if it is necessary to apply negative corrective action. For instance, if the valve in the flow loop was 100–0% valve opening for 0–100% control output – meaning that the controller action has to be reversed. Some process control schemes and final control elements require this reverse action. An example would be a valve for cooling water, where the fail-safe mode, in the case of signal loss, would be 100% opening of the valve; therefore 0% controller output needs to cause 100% valve opening.

The overall control function :

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \tag{3.1}$$

where the non-negative k_p, k_i and k_d are the coefficients for the proportional, integral, and derivative terms respectively (sometimes denoted *P*, *I*, and *D*).

An appropriate control action is sometimes provided by only one or two PID terms where the unused parameters are set to zero. This is called a PI, PD, P or I controller when the other control actions are absent. If the derivative action is sensitive to measurement noise, the PI controllers will be used only. Normally, to reach the target value of the system, the integral term is needed.

3.2 PID Controller Design Process

This section provides steps for designing a PID controller for aircraft taxiing.

3.2.1 Aircraft navigation optimization

The navigation system (Figure 3.2) consists of several blocks that allow aircraft to move on the ground in an optimal manner. At a particular airport, the scheduler will typically provide a list of waypoints as well as time deadlines to meet each waypoint. These waypoints are typically determined by an optimal scheduling algorithm (e.g., the k-quickest path problem with time windows (k- QPPTW) algorithm [7] or the airport multi-objective A* (AMOA*) algorithm [102], and the aircraft is required to follow this schedule in an optimal manner. The navigation system consists of an outer loop controller (denoted by the speed and heading algorithm), which determines the references. In contrast to existing open loop approaches [103-105], these reference points are calculated online based on the distance to the next waypoint and the time deadlines. The inner loop control system moves the control inputs so that the aircraft follow the reference points. When automation is not present, the inner loop system controller is usually the pilot. The inner loop controller can, however, serve as a guide to the pilot on how best to move the controller inputs so that taxiing between waypoints is optimal (i.e., uses the minimum amount of fuel). Therefore, the main goal of this part is to develop an optimal controller so as to move aircraft through the taxiways while meeting the schedule, whether through full automation or acting as a decision support. The approaches utilized here

is based on minimizing some set objectives such as the fuel consumed when following the schedule.

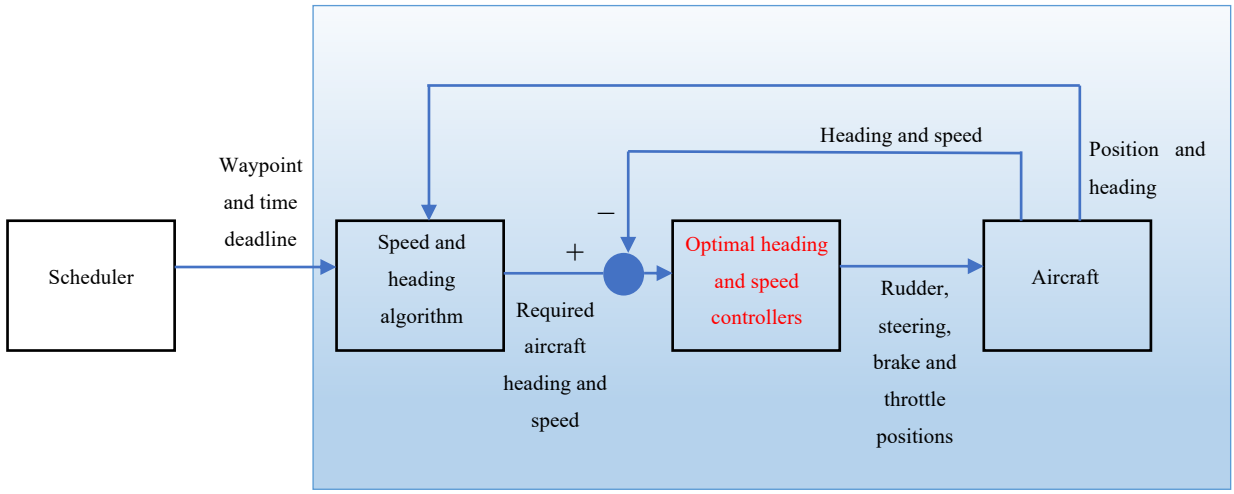


Figure 3.2:Block diagram of the navigation system [70].

3.2.1.1 Outer loop

An outer loop controller is part of the navigation system that determines the references (as indicated by the speed and heading algorithm). The time deadlines and the distance to the next waypoint are used to compute online the reference points. The control inputs are moved by the inner loop control system so that the reference points are followed by the aircraft. When automation is not available, usually the pilot is the inner loop system controller. However, the inner loop controller, can advise the pilot on how to transfer the inputs of the controller to taxi between waypoints with minimum amount of fuel.

The outer controller determines the heading and speed when following a specified schedule. This loop provides the references (speed and heading reference) to the inner controller. The references are determined based on the scheduler output and a simple example below is used to illustrate how the outer loop works.

Example:

Suppose the scheduler provides the following information to the outer loop controller "move the aircraft from along a straight line of distance 500 m in 50s".

It is the inner loop which calculates the speed and heading reference at any point in time. The process by which the reference is generated is explained in the following steps:

a) Determine type of segment: The first step is to determine if the movement to the specified way point involves passing through a turning or straight-line segment. The distance for the straight line segment is calculated using the following formula:

b) Calculate the distance (D) to the next way point.

$$D = \sqrt{(x_p - x_c)^2 + (y_p - y_c} \tag{3.2}$$

where (y_p, y_c) refers to the coordinate of the way point and (x_p, x_c) is the coordinate of the current position.

c) Calculate the predicted distance, assuming the aircraft continues to move at the current speed. The predicted distance D_{pred} is given as follows:

$$D_{pred} = S_c . t_r \tag{3.3}$$

where t_r is the time remaining to get to the waypoint and S_c is the current speed. The predicted distance is subtracted from the distance to the next waypoint to create the distance error variable (E).

d) Calculate reference base on the following equation:

$$S_{ref} = S_c + \gamma \mathrm{E} \tag{3.4}$$

Where S_{ref} is the reference to the inner controller, and γ is a constant that determines the rate of change of the reference speed. γ can also be optimized, but when it is not optimized, a value of $\gamma = 10$ was found, by trial and error, to work well.

e) In some cases, a speed constraint is specified at the end (e.g., when going from a straight segment into a turning segment a fixed speed is to be used at that turning segment). In such a case, the following algorithm is incorporated to calculate the reference speed:

(i) Based on the current speed S_c and noting the maximum deceleration ($1\ m \cdot s^{-2}$) and maximum acceleration (also $1\ m \cdot s^{-2_2}$), calculate the time (t_d) that would take based on the current speed for the aircraft to meet the required speed at the end. This time can be found by using the following equation: $t_d = \Delta S/1$, where ΔS is the absolute difference between S_c and the specified speed at the waypoint (S_f).

(ii) Given t_d and assuming constant rate of acceleration or deceleration, calculate the distance D_t required for such an acceleration/ deceleration to meet the speed constraint at the way point. This can be calculated as follows:

$$D_t = t_d \cdot \Delta S/2 \tag{3.5}$$

(iii) Calculate a buffer distance (D_b) which determines the total distance travelled in trying the meet the speed constraint at the way point. This is calculated as follows:

$$D_b = D_d + t_d \cdot S_f \tag{3.6}$$

The above equation determines the distance required to accelerate or decelerate to meet the required speed constraint at the waypoint.

(iv) If the remaining time deadline required to meet the waypoint is greater than t_d, then follow the normal control routine (as defined in steps (i)–(iii)) making sure to use $D - D_b$ in place of D and $t_r - t_d$ in place of t_r.

(v) If the remaining time deadline is, however, less than or equal to t_d ,then set $S_{ref} = S_f$.

3.2.1.2 Turning segments

For turning segments the above navigation algorithm is used but the distance is calculated in a different manner. One of the processes by which the aircraft navigates round a turning segment is described as follows:

The method utilised in taxiing mode is a form of trajectory following. The trajectory can be found using the following steps

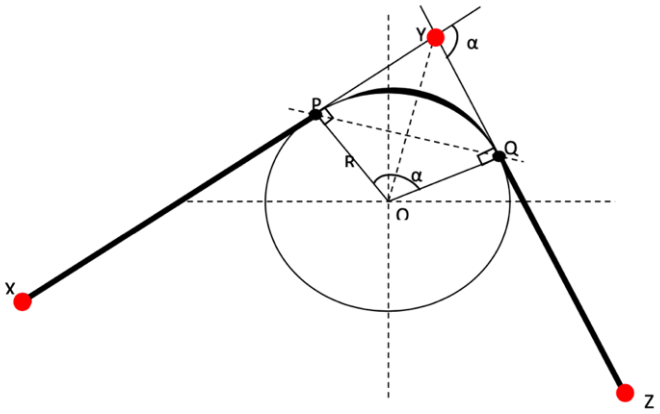


Figure 3.3:Illustration of turning segments [72].

The turn angle shown in Figure 3.3 is calculated as follows:

i. Calculate the turn angle

$$\text{Turn Angle } (\alpha) = \text{Heading(PX)} - \text{Heading(QZ)} \tag{3.7}$$

ii. Let $\theta = \alpha/2$ i.e. POY

iii. The turning segment can be transformed as shown in Figure 3.4:

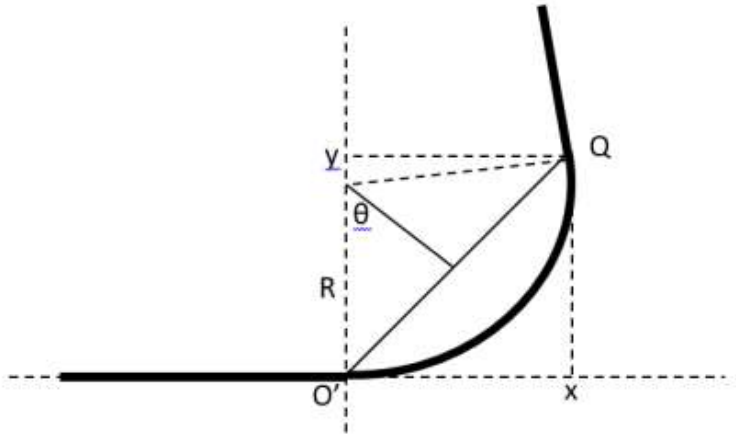


Figure 3.4:Calculating the turning segment variables [72].

iv. Calculating the turning segment variables:

$$D = |MQ| = 2 * (R * \sin\theta)$$

$$x = 2 * (R * \sin\theta \cos\theta) \tag{3.8}$$
$$y = 2 * (R * \sin^2 \theta)$$

v. Discretise the segment so that the controller is fed discrete internal waypoints. An example of an internal waypoints is shown in Figure 3.5:

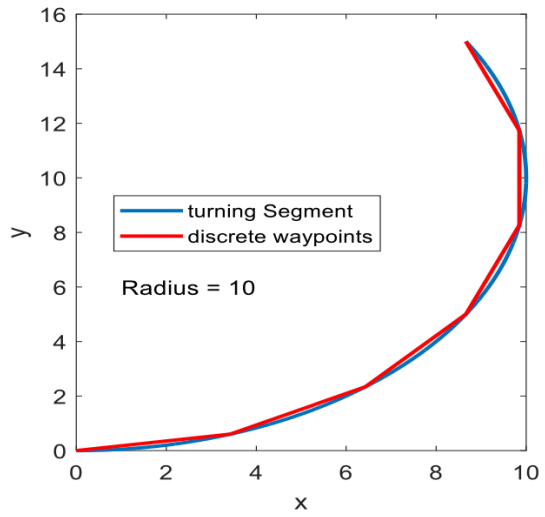


Figure 3.5:Discretisation of the turning segments [72].

vi. Design controller to follow the trajectory after rotation of the coordinate system.

The code snippet below can be utilised for the approach illustrated above:

```
% This shows how the turning angle is discretised so that trajectory following is achieved during turning when
% an aircraft is taxiing

maxTheta = 120;
theta = 0:sign(maxTheta)*0.1:maxTheta;

alpha = theta/2;

R = 10;

D = 2*R*sind(alpha);

x = D.*cosd(alpha);
```

```
y = D.*sind(alpha);

numberOfWaypoints = 7;
internalAngles = linspace(0, maxTheta, numberOfWaypoints);

internalAlpha = internalAngles/2;
DWay = 2*R*sind(internalAlpha);

xWay = DWay.*cosd(internalAlpha);

yWay = DWay.*sind(internalAlpha);

plot(x,y,'DisplayName','turning Segment', 'linewidth', 2)

hold on

plot(xWay, yWay, 'r', 'DisplayName','discrete waypoints', 'linewidth', 2)

legend('turning Segment' ,'discrete waypoints')
txt = sprintf('Radius = %0.2g', R);
text(1, 6, txt, 'fontsize', 12)

set(gca, 'fontsize', 12 )

 xlabel('x')
ylabel('y')
```

Another approach utilised in the development of the aircraft model for navigating through turning segments involves utilising the assumption that the turn rate is constant. The turn rate is given by the ratio of the turn angle and the time it is required to navigate the turning bend. The heading is thus increased at a constant rate whilst the speed is calculated in the same way as the earlier proposed navigation algorithm.

3.2.1.3 Inner loop control system

The inputs to the aircraft (throttle, braking and rudder) are manipulated by the adequately optimised PID controller developed in [106] so that the references are followed by the aircraft. Three sets of PID controllers exist in the inner loop control, hence manipulating the throttle, brakes, and rudder respectively. For braking, only the proportional controller is used. In order to prevent the brakes and throttle from being used concomitantly, the control block include a function that allows the brakes to be used solely when the aircraft speed overshoots the

reference speed, and the throttle to be employed when the current speed undershoots the reference speed. Figure 3.6 shows the PID control system configuration. The Genetic Algorithm (GA) in MATLAB is used to tune the PID parameters optimally for aircraft taxiing scenarios at a given airport, based on fuel consumed.

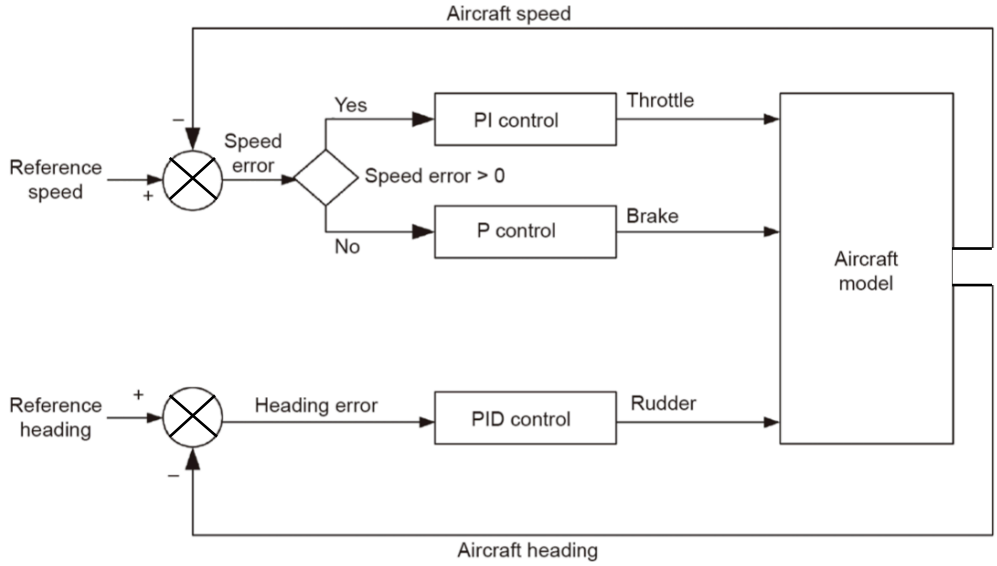
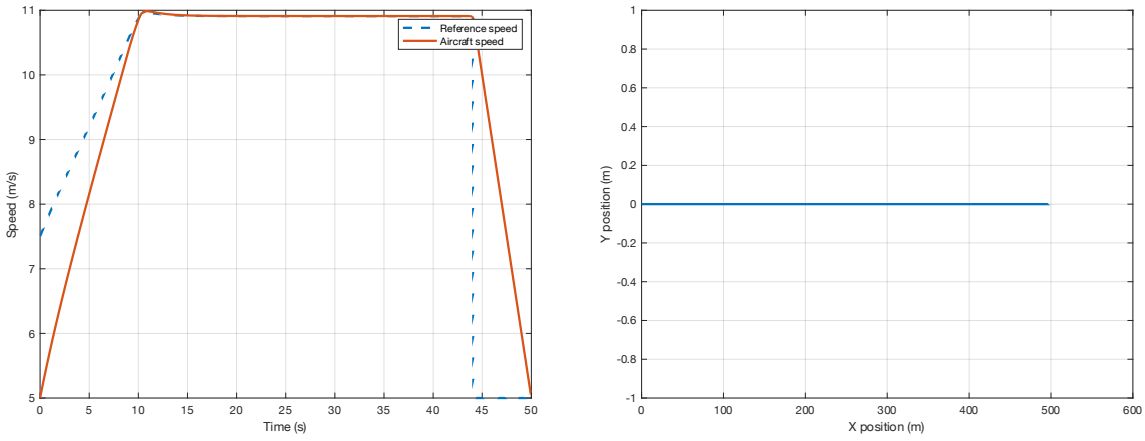


Figure 3.6:The PID control system configuration [70].

The PID controller described above has been tested using several scenarios that involve moving to/from the origin to the point (500 m, 0 m) in 50 s with an initial speed of 5 m/s and a final speed of 5 m/s. The result of testing the algorithm on this scenario is shown in Figure 3.7.



(a) (b)

Figure 3.7:(a) Aircraft reference and actual speeds of the straight line scenario where the final speed is specified (5 m/s); (b) X,Y position of the aircraft

3.3 The Limitation of PID Controller

Control problems arise when one is confronted with different scenarios, and as a result the optimal PID gains for one specific scenario would become sub-optimal for the other scenario. For each new scenario, the optimization algorithm had to be re-ran to find out the optimal gains for that scenario. As a case-study, the scheduler includes about 90000 taxiing scenarios, which would lead to significant computational costs to find the optimal PID gains for each one of these scenarios. Evidently, obtaining 90000 sets of optimal PID gains would not be practical [106].

3.4 The Clustering Solution

This problem is solved by applying the clustering strategy on the lists of scenarios. The lists of scenarios are divided into six clusters and each cluster has a cluster centre. The optimization is performed only on those scenarios that represent the cluster centres, and the PID gains of the particular centre are utilized across the whole cluster members [106]. The result of the cluster analysis for the case of the Manchester Airport is shown in Figure 3.8.

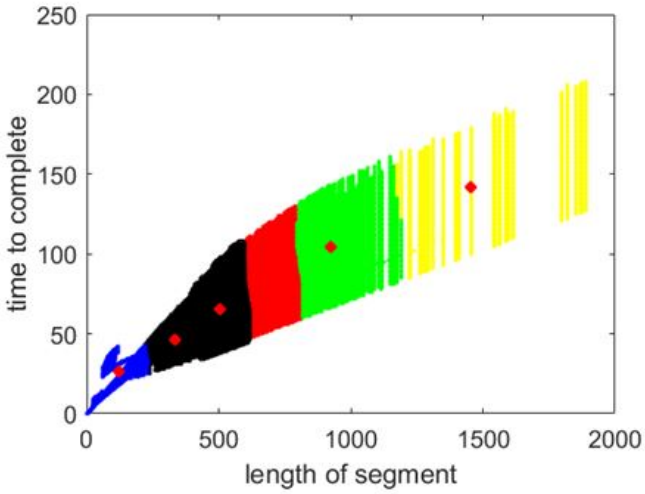


Figure 3.8:Clustering the scenarios with six clusters [70].

Table 3.1 includes details of the result of utilizing the clustered PID control approach as compared with two other methodologies on ten randomly chosen scenarios. The first relates to choosing a controller randomly from the six controllers, and the second involves choosing a fixed controller for all the scenarios chosen.

As can be seen from Table 3.1, using a clustering strategy results in a significantly better performance than using just one controller for all scenarios. It should be noted that a penalty term (2000 kg of fuel consumed) has been included for scenarios where the control strategy violates the scheduler or aircraft constraints. The total fuel consumed from the three different control strategies for the 1000 randomly chosen scenarios is shown in Table 3.2. It can be seen that, on average, the clustering-based approach consistently provides better performances than a strategy using a fixed controller.

Table 3.1:Comparison of the results of three control strategies [70].

Number	Cluster strategy		Random strategy		Fixed strategy	
	Fuel consumed (kg)	Controller	Fuel consumed (kg)	Controller	Fuel consumed (kg)	Controller
77 011	13.3512	2	2 014.57	3	13.35	2
85 775	37.2782	5	57.39	1	126.04	2
69 946	38.1160	3	46.48	4	2 069.71	2
55 515	32.6240	1	27.52	5	2 070.44	2
74 997	23.0480	3	23.68	2	23.67	2
79 525	23.2550	3	22.56	1	43.65	2
88 490	21.0820	3	2 021.06	1	31.25	2
52 906	8.8150	6	2 005.53	1	8.03	2
37 492	20.6830	4	20.68	4	29.16	2
89 095	14.5660	2	2 013.52	5	14.56	2
Total	232.8200	–	8 253.00	–	4 429.90	2

Table 3.2:Total fuel consumed for 1000 randomly chosen scenarios [70].

Cluster strategy	Random strategy	Fixed strategy (Controller 2)
56 414	696 363	321 495

3.5 Results of Experiments

3.5.1 Taxiing run along a circle path

A cirle layout is the first scenario used for aircraft taxiing test. The origin is taken to be the starting point of aircraft. The layout is divided into four different segments from this point. Figure 3.9 shows the plotts of the attained cartesian positions for the circle path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.3. The taxiing run results show that the PID controller is capable of following the specified route within the spciefied time deadline which is 180 s.

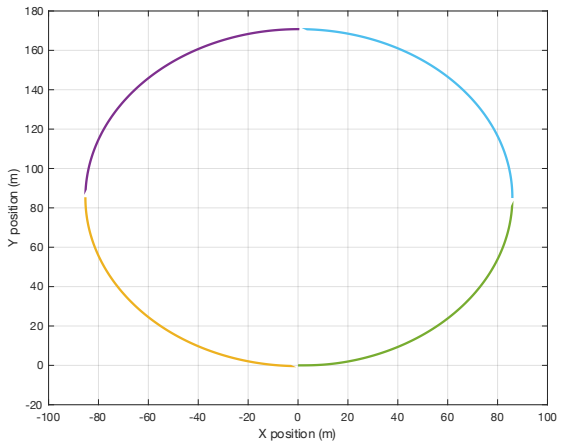


Figure 3.9:The aircraft test for the circle layout and its cartesian positions

Table 3.3:Geodetic coordinates of the circle waypoints.

Segment	X position (m)	Y position (m)	Time deadline (s)	Turn angle (°)
1	85.94	85.94	45	90
2	0	170.98	45	90
3	- 85.94	85.94	45	90
4	0	0	45	90

3.5.2 Taxiing run along a rectangular path

A rectangular layout is the second scenario used for aircraft taxiing test. The end left of straight part of the base of this rectangle is the starting point of aircraft. The layout is divided into eight (8) different segments from this point. Figure 3.10 shows the plotts of the attained cartesian positions for the rectangular path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.4. The taxiing run results show that the PID controller is capable of following the specified route within the spciefied time deadline which is 465 s.

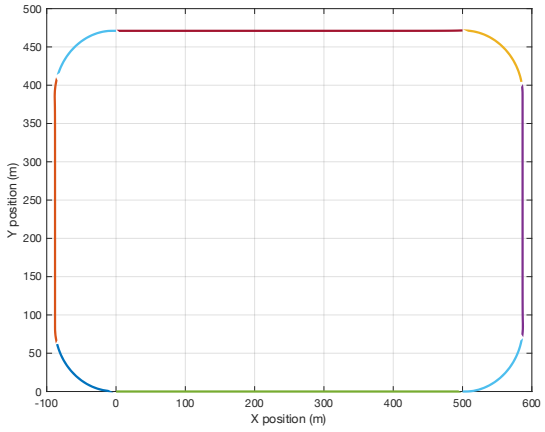


Figure 3.10:The aircraft test for the rectangular layout and its cartesian positions

Table 3.4:Geodetic coordinates of the rectangular waypoints.

Segment	X position (m)	Y position (m)	Time deadline (s)	Turn angle (°)
1	500	0	80	0
2	585.94	85.94	45	90
3	585.94	385.94	65	0
4	500	471.88	45	90
5	0	471.88	75	0
6	- 85.94	385.94	45	90
7	- 85.94	85.94	65	0
8	0	0	45	90

3.6 Summary

In this chapter, the PID controller has been introduced as it was the first controller applied for aircraft taxiing; the aircraft navigation system was described which consists of several blocks that allows aircraft to move from one point to the other on the ground in an optimal manner. An outer loop controller is part of the navigation system that determines the references (as indicated by the speed and heading algorithm) was also detailed with an example of moving the aircraft from along a straight line of distance 500 m in 50s". The turning segments were also outlined were the navigation algorithm is used but the distance is calculated in a different manner.

Three sets of PID controllers existed in the inner loop control, hence manipulating the throttle, brakes, and rudder respectively. For braking, only the proportional controller was used. The Genetic Algorithm (GA) in MATLAB is used to tune the PID parameters optimally for aircraft taxiing scenarios at a given airport, based on fuel consumed.

Control problems occurred when one is confronted with different taxiing scenarios, and as a result the optimal PID gains for one specific scenario would become sub-optimal for the other scenario. This problem was tackled using the clustering strategy on the lists of scenarios. It can be seen using a clustering strategy results in a significantly better performance than using just one controller for all scenarios.

The PID controller was validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout. The simulation results show that the PID controller is capable of following the specified route within the specified time deadline.

The following chapter will include enhancing the capabilities of the PID controller where the PID gains not need to be changed each time the scenario changed. The proposed artificial neural networks will be developed to solve this problem.

4 Artificial Neural Networks for Aircraft Taxiing

4.1 Introduction

Scientists had the inspiration for the artificial neural networks (ANN) from biological neural networks which exist in human brains [107] such as in Figure 4.1. The reason for this is to mimic the functionality of learning and solving problems that human brains are able to do easily and in a parallel way. In the modern world, the artificial neural networks are widely used in many areas such as computer vision, robotics, medicine, space problems and control etc. More and more experts have noticed the potential of neural networks development for the future.

Figure 4.1: Biological neural networks in the human brain [108].

Neural networks are considered one of the most effective learning methods that are used in learning to interpret sensor data of complex real-world problems. When artificial neural networks are implemented, weights are used to represent knowledge to relate the relationship between data. The network parameters are consequently adjusted to decrease the error between the desired outputs and neural network outputs using a gradient descent technique. Back-propagation network is one of the most commonly used neural networks. It is trained with

supervision using a gradient descent method to decrease the error between the network outputs and the desired outputs [109].

Artificial neural networks can be used in many applications, such as clustering [110-116], optimization[117-122], prediction [123-132], classification [133-141], data processing [142-144] and function approximation [145-152] . This is due to their ability to learn from the surrounding environment. The way they learn from the environment can be divided into three categories: unsupervised learning, which is known as learning without a teacher, supervised learning (with a teacher), and reinforcement learning [153]. In this project we will concentrate on the supervised learning where our neural network is provided by the inputs and then produces outputs to be compared with the desired outputs.

Backpropagation is a powerful learning technique for artificial neural network. There are many inputs and many outputs as well, the number of which depends on the problem aimed to be solved. Each two neurons are one way interconnected via a link. A numeric weight is assigned to each link. The neurons are organized in three different layers. In Figure 4.2, the input layer is the first layer follwed by the hidden layer and finally the output layer. More than one layer can be in the hidden layer. Values from the input vector are recived by the neurones in the input layer and then propagated to the rest of the network. The output vector values are generated by the output neurons in the output layer. The repeated gradient descent algorithm is applied to diminish the error, i.e., mean square error, between the neural network outputs and the desired outputs [109].

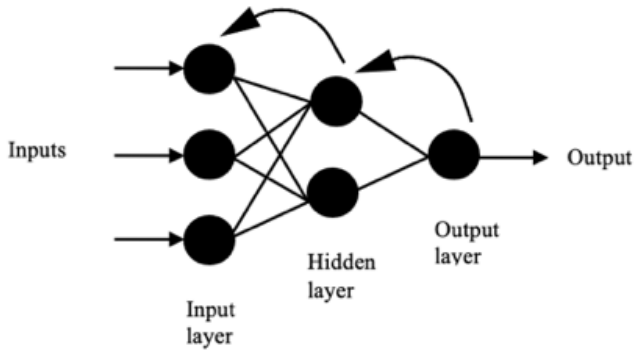


Figure 4.2:Backpropagation artificial neural networks architecture.

4.2 Neural Networks Equations

In artificial networks each step needs its own calculations using specific equations to achieve the required objective. First of all, in multilayer back propagation architecture, each hidden neuron H is calculated by multiplying all the inputs X by their associated weights which is connected to that hidden neuron, after that an activation function (sigmoid function) is applied to the summation of the multiplication v, forcing the output $\varphi(v)$ to range between 0 and 1;

$$\varphi(v) = \frac{1}{1 + \exp(-\lambda v)} \tag{4.1}$$

The same procedure is then applied to find the output Y_k, multiplying the results of the activation function φ^h by the weights that are connected to that output neuron W_{ki} and finally apply the activation function equation (4.1) to the summation of the multiplication.

$$Y_k = \varphi\left[\sum_{i=0}^{n} W_{ki}\, \varphi^h \left(\sum_{j=0}^{m} W_{ij}^h X_j\right)\right] \tag{4.2}$$

After calculating the output of the neural network, now the need is to calculate the error $e(t)$ which can be defined by finding the difference between the desired output $d(t)$ and the neural network output $y(t)$.

$$e(t) = d(t) - y(t) \tag{4.3}$$

On the other hand, to apply a backpropagation approach, firstly, the local gradient of the outputs must be calculated.

$$\delta_k(t) = \lambda . \varphi\big(v_k(t)\big) . \big[1 - \varphi\big(v_k(t)\big)\big] . e_k(t) \tag{4.4}$$

Secondly, calculating local gradients of the hidden neurons. So, the following equation shows the solution for that:

$$\delta_i^h(t) = \lambda . \varphi^h\left(v_i^h(t)\right) . \left[1 - \varphi^h\left(v_i^h(t)\right)\right] . \left[\sum_{k=1}^{l} \delta_k(t) . w_{ki}(t)\right] \quad (4.5)$$

After calculating the local gradient of the outputs and hidden neurons, the update of the output weights can now be done using:

$$w_{ki}(t+1) = w_{ki}(t) + \Delta w_{ki} \quad (4.6)$$

Now calculating the delta weight of the outputs and adding the momentum to speed up the convergence.

$$\Delta w_{ki}(t) = \eta . \delta_k(t) . h_i(t) + \alpha . \Delta w_{ki}(t-1) \quad (4.7)$$

To update the hidden weight, the following equation is used:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \quad (4.8)$$

Now calculating the delta weight of the hidden neurons and adding the momentum to speed up the convergence.

$$\Delta w_{ij}(t) = \eta . \delta_i^h(t) . x_j(t) + \alpha . \Delta \delta w_{ij}^h(t-1) \quad (4.9)$$

The final step can be done after achieving all the previous steps for the whole data samples which is called one epoch. After each epoch we calculate the root mean square error (RMSE) to find the lowest error that indicates that the neural network is sufficiently trained and has the optimal weights to be used later when we test the network. To calculate the RMSE the following equation can be applied:

$$\varepsilon(t) = \sqrt{\frac{1}{2} \cdot \sum_{k=1}^{l} e_k^2(t)} \tag{4.10}$$

4.3 Neural Network Design Process

This section provides steps for designing an offline neural network model for aircraft taxiing.

4.3.1 Data collection

It is known that the neural network needs data to learn and generalize. The data was obtained from the existed PID controller where one of the aircraft taxiing scenarios (waypoint is x=500 m, y=0 m, time is 50 s) was run offline, and the speed errors and change of speed errors were saved and used as inputs to train the neural network. Also the throttle and brake values of the same scenario were saved and used to be the desired outputs to the neural network during training process.

4.3.2 Build the network

The process by which the neural networks model was built relied on the Neural Networks toolbox in MATLAB[154]; the Network type being used is the feed-forward backpropagation [155, 156]. The built network includes: 2 inputs (speed error, change in speed error), 2 hidden layers with 10 and 5 neurons respectively and finally 2 outputs (throttle value, break value). *Trainlm* is used for training function, *learngdm* for adaptation learning function and for performance *MSE* is used. *Tansig* is the transfer function for the hidden layer neurons and *purelin* is the transfer function for the output layer neurons. The structure of the built neural networks model is shown in Figure 4.3.

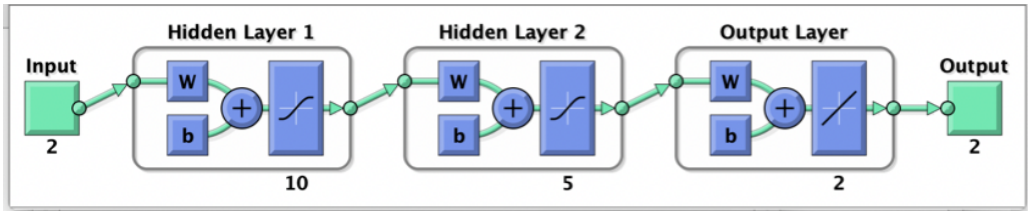


Figure 4.3:The structure of the built neural networks

4.3.3 Initialize the weights and biases

The weights and biases must be initialized before training the network. The neural network tool box will initialize the weights and biases to zeros.

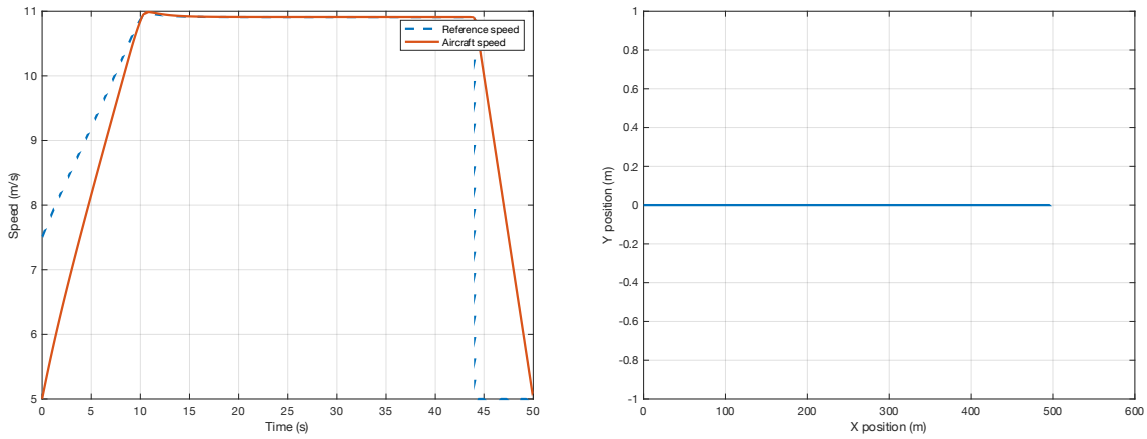
4.3.4 Train the network

After configuring the network, we need to tune the network parameters weights and biases [157, 158], hence the network performance is optimized. The network performance is measured using root mean square error where minimizing the overall error between the desired and actual outputs of the network is the goal of the training. The Root Mean Square Error (RMSE) for the throttle after training was 0.0276 and 0.0645 for braking. A learning rate parameter must be specified which represents the speed of the steps towards minimum error in order to achieve a realistic training time. Training will take too long if this quantity is too small, and if it is too large, the error will increase because of the gradient descent will degenerate.

4.3.5 Use the network

The network is successfully created and trained. Now we need to exported to MATLAB workspace and use it in the Simulink model to work as a controller for taxiing the aircraft. The aircraft inputs (throttle, rudder and braking) are manipulated by the inner control loop so that the references previously described in 3.2.1.1 are followed by the aircraft. Two sets of controllers exist in the inner loop control. The neural networks controller is responsible for manipulating throttle and braking inputs of the aircraft while the aircraft rudder is manipulated by the PID controller. In order to test the neural networks control system, we used a multitude of scenarios (some reported here others not for the sake of saving space) which include paths from the origin to the point (500 m, 0 m) in 50 s with an initial speed of 5 m/s and a final speed of 5 m/s. The Root Mean Square Error (RMSE) for the throttle after testing was 0.0090 and 0.0040 for braking. Figure 4.4 shows the result of testing the neural networks controller on this scenario. Indeed, the neural networks controller can be seen to be capable of following the

specified route within the specified time deadline. Figure 4.5 shows the configuration of the Neural Networks control system.



(a) (b)

Figure 4.4:(a) Aircraft reference and actual speeds of the straight line scenario where the final speed is specified (5 m/s); (b) X,Y position of the aircraft

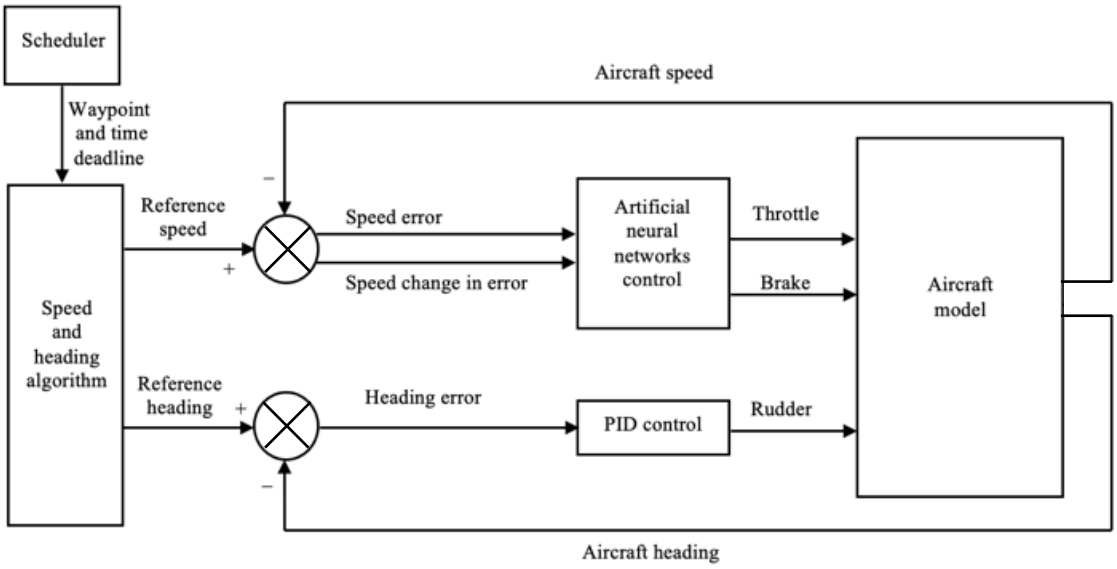


Figure 4.5:The artificial neural networks control system configuration

4.4 Results of Experiments

The optimal scheduling algorithm provides severals taxiing scenarios which have been used to test the proposed ANN controller described above that involve a rectangular path and a circle path. The RMSE was used to evaluate the proposed ANN controller's performance, whereby the overall error between the refrence speed and the aircraft's speed is calculated. The Matlab/Simulink software was used to perform all simulations. The result of testing the prposed controller on these two scenarios is discussed in the following subsections.

4.4.1 Taxiing run along a circle path

A cirle layout is the first scenario used for aircraft taxiing test. The origin is taken to be the starting point of aircraft. The layout is divided into four different segments from this point. Figure 4.6 shows the plotts of the attained cartesian positions for the circle path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.3. The RMSE for the speed was 0.0125 (m/s). The taxiing run results show that the neural networks controller is capable of following the specified route within the spciefied time deadline which is 180 s.

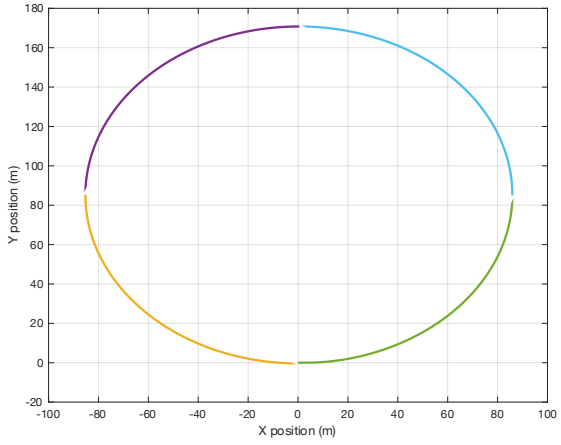


Figure 4.6:The aircraft test for the circle layout and its cartesian positions

4.4.2 Taxiing run along a rectangular path

A rectangular layout is the second scenario used for aircraft taxiing test. The end left of straight part of the base of this rectangle is the starting point of aircraft. The layout is divided into eight (8) different segments from this point. Figure 4.7 shows the plotts of the attained cartesian positions for the rectangular path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.4. The RMSE for the speed was 0.4714 (m/s). The taxiing run results show that the neural networks controller is capable of following the specified route within the spciefied time deadline which is 465 s.

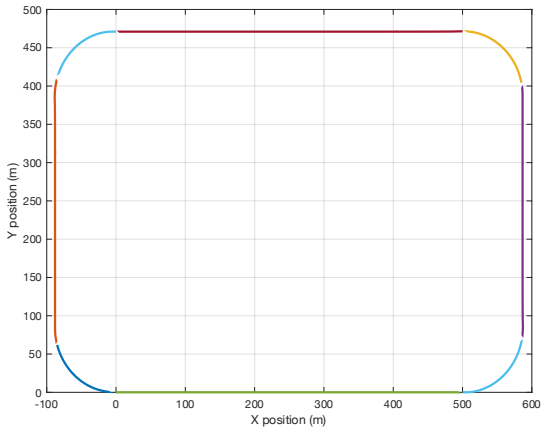


Figure 4.7: The aircraft test for the rectangular layout and its cartesian positions

4.5 Summary

In this chapter, the capabilities of the PID controller introduced in Chapter 3 were enhanced, aiming for a solution where there is no need to change the PID gains each time the scenario changes. The proposed artificial neural networks in this chapter is able to overcome this problem which has the ability to learn and generalize.

In this chapter, a new control has been proposed using artificial neural networks. It was the second controller applied for aircraft taxiing; backpropagation neural network was used to build the network. It consists of three layers: the input layer is the first layer follwed by the

hidden layer and finally the output layer. The repeated gradient descent algorithm is applied to diminish the error. The equations of each step in artificial networks has been presented and explained. The steps for designing the neural network model for this project has been provided, which includes: data collection, build the network, initialize weights and biases, train the network and finally use the network.

The artificial neural networks controller was validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout. The simulation results show that the neural networks controller is capable of following the specified route within the spciefied time deadline.

The following chapter will include proposing a new fuzzy inference system (FIS) controller. Fuzzy deals with vagueness and uncertainty, it is adaptive and replicates human actions (in this case the pilots who are expert). This controller will be built using the MATLAB ANFIS toolbox with two ways: grid partition and fuzzy c-mean clustering.

5 Adaptive Neuro-Fuzzy Inference System for Aircraft Taxiing

5.1 Introduction

Mamdani fuzzy inference system [159] is a model that has a basic structure which maps input characteristics to input membership functions, input membership functions to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership functions to a single-valued output or a decision associated with the output. Arbitrarily chosen and fixed membership functions used by such a system and predetermined rule structure by the interpretation of user of the variables' characteristics in the model [160].

Fuzzy inference techniques are applied to data modelling by ANFIS and the Neuro-Fuzzy Designer. As can be seen from the other fuzzy inference GUIs, the parameters determine the shape of the membership functions, and the shape of the membership function changes when these parameters changed. Using these Fuzzy Logic Toolbox applications, membership function parameters can be automatically chosen instead of selecting them manually by just looking at the data [161].

When we have an input/output dataset for a system that we want to apply fuzzy inference to it, it is not necessarily that we should have a predetermined model for it. Sometimes it is difficult to determine the shape and parameters of the membership functions by just looking at the data. Choosing these parameters, as the membership functions are tailored to input/output data, can be achieved using ANFIS neuro-adaptive learning techniques.

The neuro-adaptive learning methods work in a similar way to neural networks. Learning information about a data set is a method provided by Neuro-adaptive learning techniques for fuzzy modelling procedure. The membership function parameters are computed by Fuzzy

Logic Toolbox software that the best way to track the input/output data to the associated fuzzy inference system. This membership function parameter adjustment is accomplished by The Fuzzy Logic Toolbox function called ANFIS [162].

5.2 What Is ANFIS?

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. After constructing the fuzzy inference system (FIS) for a given input/output data set, ANFIS function tunes (adjusts) the membership function parameters using either a back propagation algorithm alone or in combination with a least squares type of method. This adjustment allows the fuzzy systems to learn from the data they are modelling [163].

5.3 Fuzzy Inference System Structure and Parameter Adjustment

The structure of the network is similar to that of the neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map [164].

The membership functions' parameters change through the learning process. A gradient vector facilitates the adjustment of these parameters. for a given set of parameters, this gradient vector provides a measure of how well the fuzzy inference system is modelling the input/output data. To adjust the parameters to reduce some error measure, any of several optimization routines can be applied when the gradient vector is obtained. This error measure is usually expressed as the sum of the squared difference between actual and desired outputs. For the estimation of the parameters of the membership function, ANFIS uses a combination of least squares estimation and back propagation or back propagation alone [165].

5.4 Know The Data

The ANFIS modelling approach is similar to many system identification techniques. First, a parameterized model structure is hypothesized (relating inputs to membership functions to rules to outputs to membership functions, and so on). Next, for ANFIS training, input/output data are collected in a usable form. Then train the FIS model by ANFIS to emulate the training data presented to it by the membership function parameters modification based on a chosen error criterion. In general, if the training data is fully representative of the features, then this type of modelling works well by ANFIS for training (estimating) membership function parameters of the FIS is intended to model.

5.5 Fuzzy Inference System Design Process

This section provides steps for designing a fuzzy inference system for aircraft taxiing.

5.5.1 Data Collection

The data was obtained from the existed PID controller where one of the aircraft taxiing scenarios (waypoint is x=500 m, y=0 m, time is 50 s) was run offline, and the speed errors and change of speed errors were saved and used as inputs to train the fuzzy inference system (FIS) model. Also the throttle and break values of the same scenario were saved and used to be the desired outputs to the FIS during training process.

5.5.2 Designing the fuzzy inference system

The following are the ways to design the fuzzy inference system (FIS) model for aircraft taxiing using the ANFIS toolbox using grid partition and fuzzy c-mean clustering. Figure 5.11 shows the configuration of the fuzzy inference system. The aircraft inputs (throttle, rudder and braking) are manipulated by the inner control loop so that the references previously described in 3.2.1.1 are followed by the aircraft. Three sets of controllers exist in the inner loop control. The two FIS controllers are responsible for manipulating throttle and braking inputs of the aircraft while the aircraft rudder is manipulated by the PID controller.

5.5.2.1 Grid partition

The data grid partitioning is used to generate Sugeno-type FIS of a single-output. Two controllers were built, one for the throttle and one for the break.

a. Throttle controller

ANFIS is used to generate a Sugeno fuzzy inference system to control the aircraft throttle with two inputs (error and change of error) and one output (throttle value). In this fuzzy system the Grid partition is used to determine number of membership functions for each input and number of fuzzy rules. Each input has 4 membership functions of gaussian type and the output membership function is constant type. The system has16 fuzzy rules. The training was done using hybrid optimization method with 200 epoch. Throttle Minimal training RMSE = 0.003932

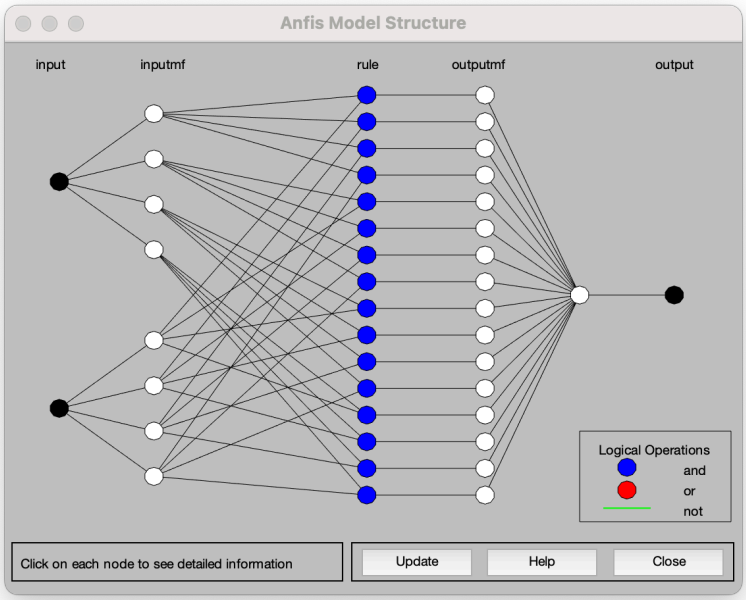


Figure 5.1:ANFIS model structure for throttle.

b. Break controller

ANFIS is used to generate a Sugeno fuzzy inference system to control the aircraft break with two inputs (error and change of error) and one output (break value). In this fuzzy system the

Grid partition is used to determine number of membership functions for each input and number of fuzzy rules. Each input has 4 membership functions of gaussian type and the output membership function is constant type. The system has16 fuzzy rules. The training was done using hybrid optimization method with 200 epoch. Break Minimal training RMSE = 0.001254

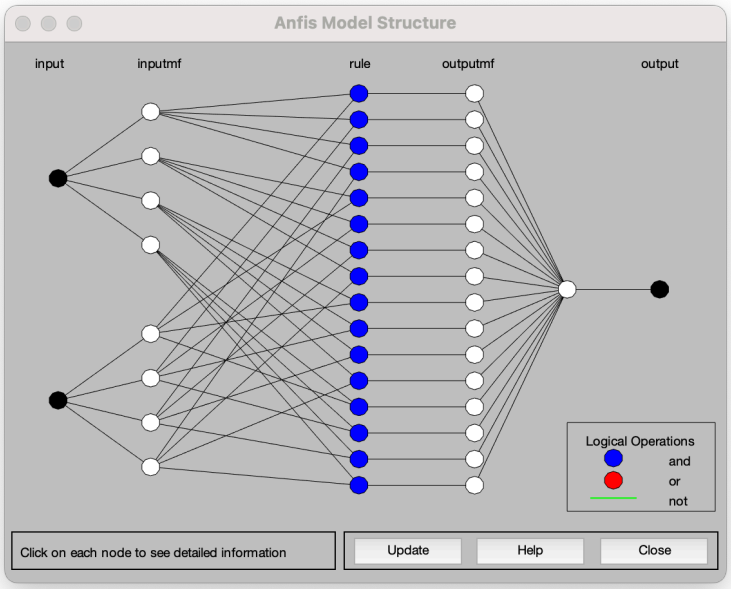
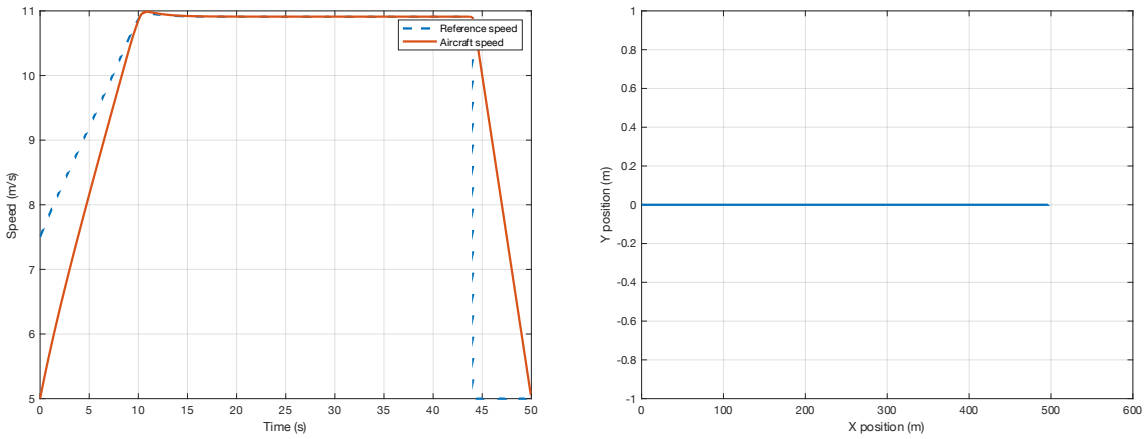


Figure 5.2:ANFIS model structure for break.

In order to test the fuzzy inference system (FIS), we used a multitude of scenarios (some reported here others not for the sake of saving space) which include paths from the origin to the point (500 m, 0 m) in 50 s with an initial speed of 5 m/s and a final speed of 5 m/s. Figure 5.3 shows the result of testing the proposed controller on this scenario. Indeed, the fuzzy inference system (FIS) can be seen to be capable of following the specified route within the specified time deadline.



(a) (b)

Figure 5.3:(a) Aircraft reference and actual speeds of the straight line scenario where the final speed is specified (5 m/s); (b) X,Y position of the aircraft

5.5.2.1.1 Results of experiments

a. Taxiing run along a circle path

A cirle layout is the first scenario used for aircraft taxiing test. The origin is taken to be the starting point of aircraft. The layout is divided into four different segments from this point. Figure 5.4 shows the plotts of the attained cartesian positions for the circle path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.3. The taxiing run results show that the fuzzy inference system (FIS) is capable of following the specified route within the spciefied time deadline which is 180 s.

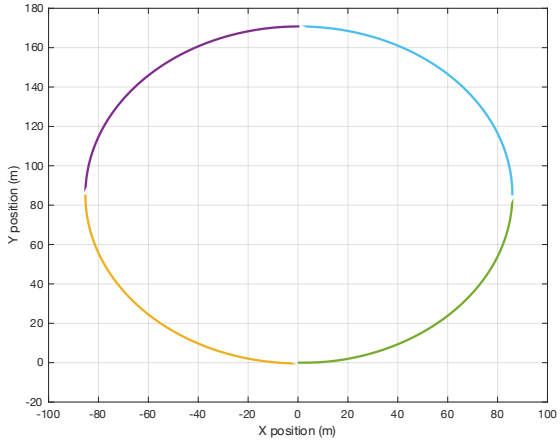


Figure 5.4:The aircraft test for the circle layout and its cartesian positions

b. Taxiing run along a rectangular path

A rectangular layout is the second scenario used for aircraft taxiing test. The end left of straight part of the base of this rectangle is the starting point of aircraft. The layout is divided into eight (8) different segments from this point. Figure 5.5 shows the plotts of the attained cartesian positions for the rectangular path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.4. The taxiing run results show that the fuzzy inference system (FIS) is capable of following the specified route within the spciefied time deadline which is 465 s.

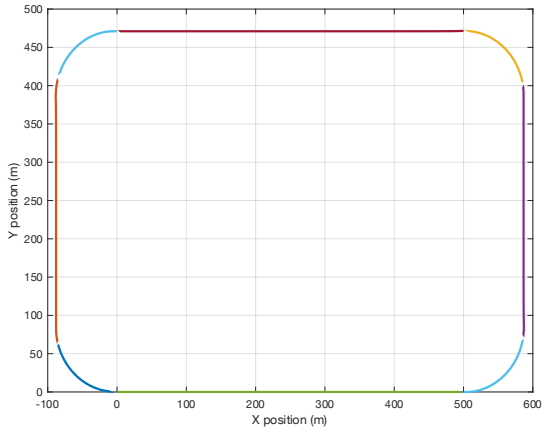


Figure 5.5:The aircraft test for the rectangular layout and its cartesian positions

5.5.2.2 Fuzzy c-mean clustering

The fuzzy c-mean is used to generate Sugeno-type FIS of a single-output. Two controllers were built, one for the throttle and one for the break.

a. Throttle controller

ANFIS is used to generate a Sugeno fuzzy inference system to control the aircraft throttle with two inputs (error and change of error) and one output (throttle value). In this fuzzy system the fuzzy c-mean clustering is used to cluster each input training data. Each input has 17 clusters/membership functions of gaussian type. The output membership function is constant type. The total number of rules are 17 rules. The training was done using hybrid optimization method with 50 epoch. Minimal training RMSE = 0.00062266

Below is the line of code that used to generate fuzzy inference system using fuzzy c-mean clustering method:

```
fismat = genfis3([ErrorInput ChangeInErrorInput],ThrottleOutput
,'sugeno',17);
```

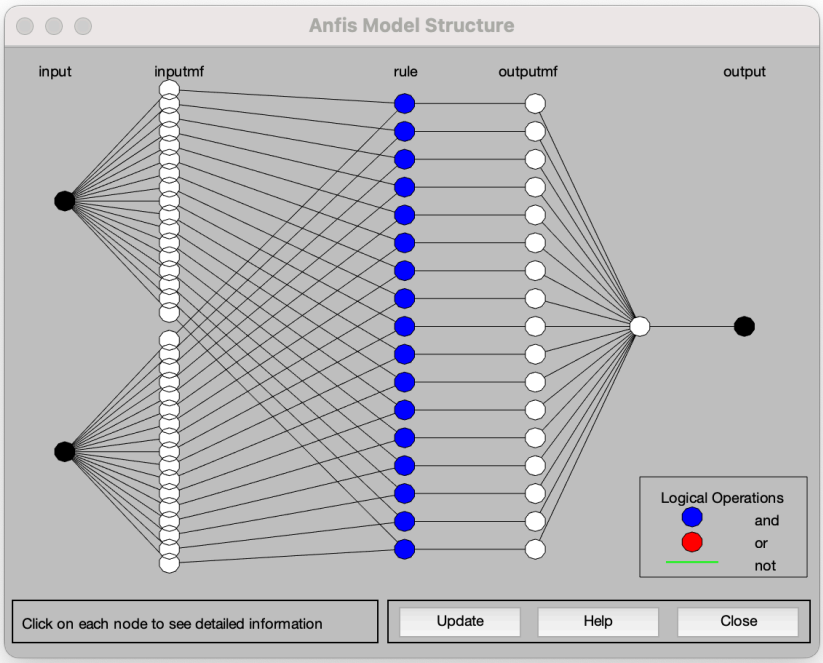


Figure 5.6:ANFIS model structure for throttle.

b. Break controller

ANFIS is used to generate a Sugeno fuzzy inference system to control the aircraft break with two inputs (error and change of error) and one output (break value). In this fuzzy system the fuzzy c-mean clustering is used to cluster each input training data. Each input has 60 clusters/membership functions of guession type. The output membership function is constant type. The total number of rules are 60 rules. The training was done using hybrid optimization method with 100 epoch. Minimal training RMSE = 0.000339357

Below is the line of code that used to generate fuzzy inference system using fuzzy c-mean clustering method:

```
fismat2 = genfis3([ErrorInput ChangeInErrorInput],BreakOutput
,'sugeno',60);
```

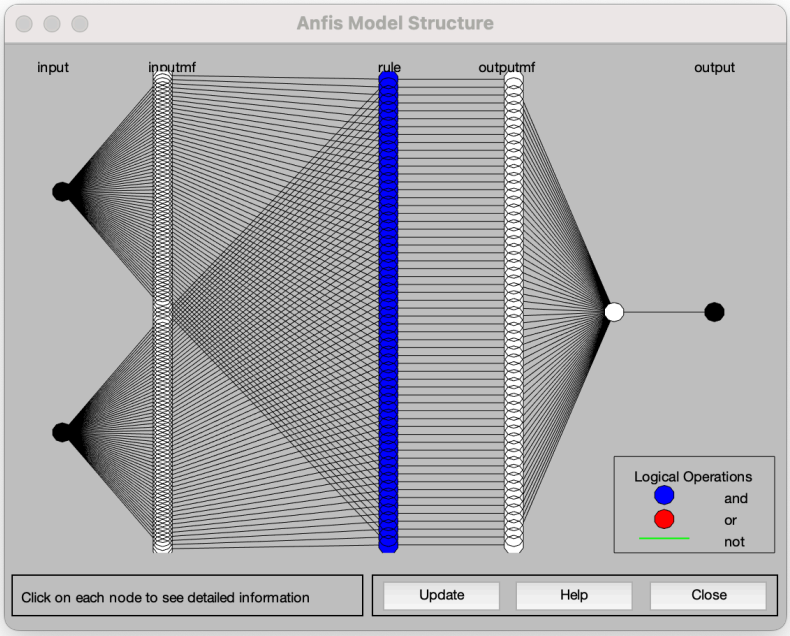
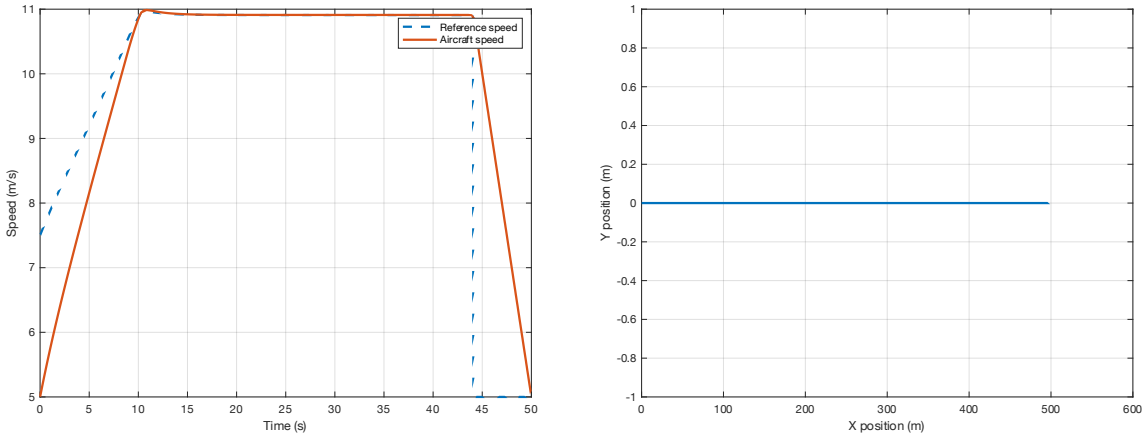


Figure 5.7:ANFIS model structure for throttle for brake.

In order to test the fuzzy inference system (FIS), we used a multitude of scenarios (some reported here others not for the sake of saving space) which include paths from the origin to the point (500 m, 0 m) in 50 s with an initial speed of 5 m/s and a final speed of 5 m/s. Figure 5.8 shows the result of testing the proposed controller on this scenario. Indeed, the fuzzy inference system (FIS) can be seen to be capable of following the specified route within the specified time deadline.



(a) (b)

Figure 5.8:(a) Aircraft reference and actual speeds of the straight line scenario where the final speed is specified (5 m/s); (b) X,Y position of the aircraft

5.5.2.2.1 Results of experiments

a. Taxiing run along a circle path

A cirle layout is the first scenario used for aircraft taxiing test. The origin is taken to be the starting point of aircraft. The layout is divided into four different segments from this point. Figure 5.9 shows the plotts of the attained cartesian positions for the circle path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.3. The taxiing run results show that the fuzzy inference system (FIS) is capable of following the specified route within the spciefied time deadline.

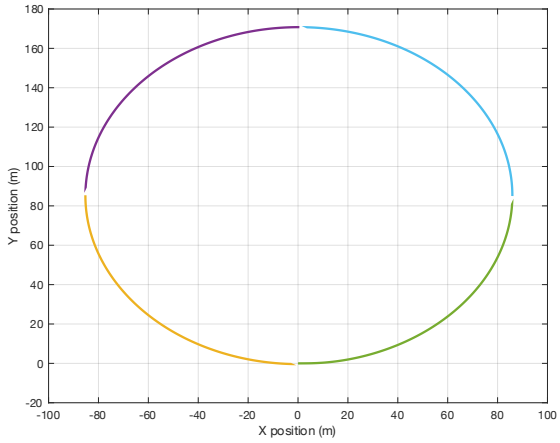


Figure 5.9:The aircraft test for the circle layout and its cartesian positions.

b. Taxiing run along a rectangular path

A rectangular layout is the second scenario used for aircraft taxiing test. The end left of straight part of the base of this rectangle is the starting point of aircraft. The layout is divided into eight (8) different segments from this point. Figure 5.10 shows the plotts of the attained cartesian positions for the rectangular path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.4. The

taxiing run results show that the fuzzy inference system (FIS) is capable of following the specified route within the spciefied time deadline.

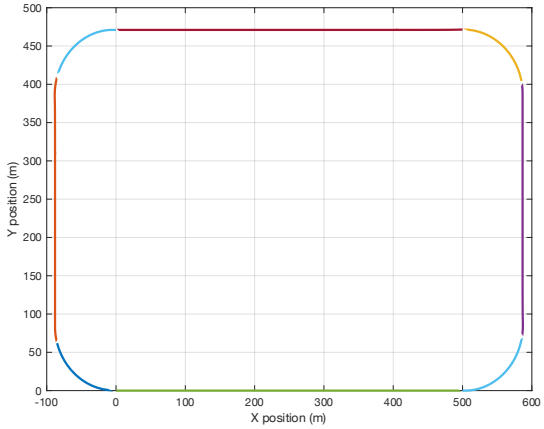


Figure 5.10:The aircraft test for the rectangular layout and its cartesian positions

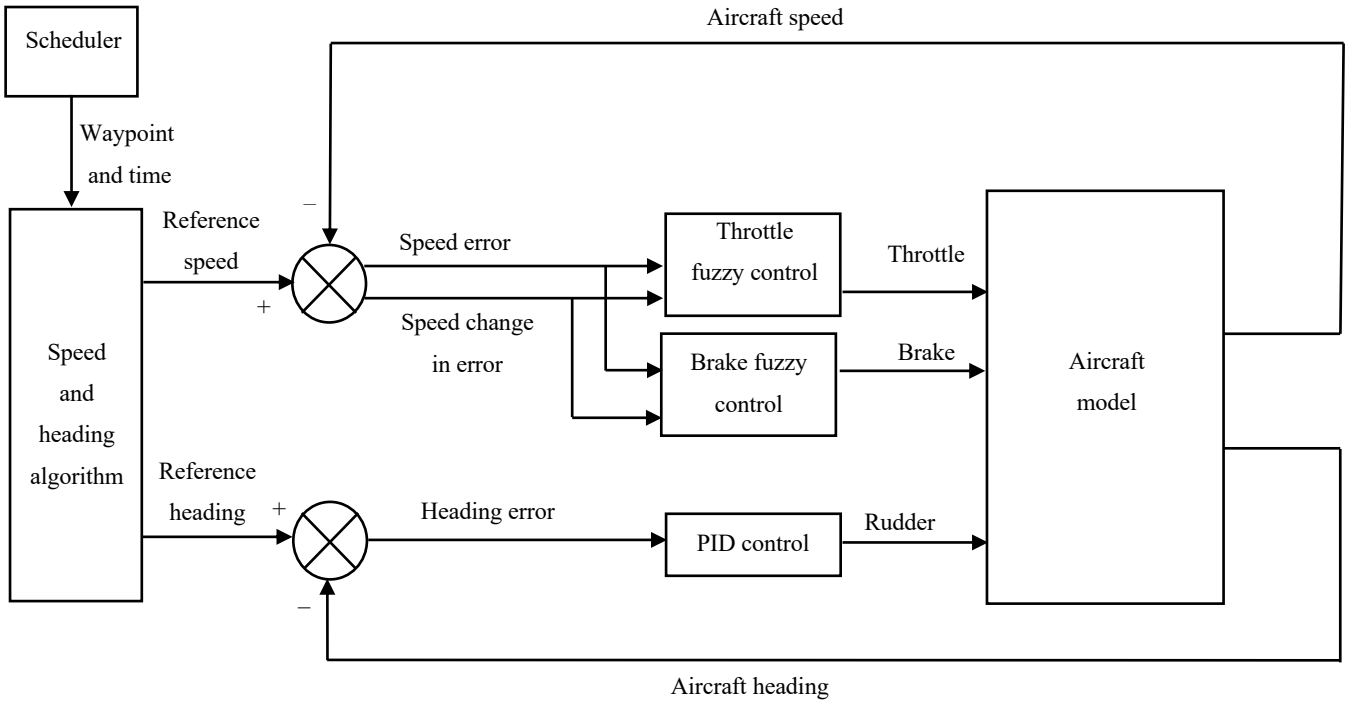


Figure 5.11:The fuzzy inference system (FIS) configuration.

5.6 Summary

The chapter started with a review of fuzzy inference system and how its structure and parameter adjustment. Also, an explanation of ANFIS (adaptive neuro-fuzzy inference system) has been presented which used in this part. It mainly used to tunes (adjusts) the membership function parameters that allows the fuzzy systems to learn from the data they are modelling.

The new proposed fuzzy inference system (FIS) controller was built using the MATLAB ANFIS toolbox with two ways: grid partition and fuzzy c-mean clustering. It was the third controller applied for aircraft taxiing. In both ways there were two controllers built, one for the throttle and one for the brake as ANFIS allows only for one output and in our system we have two outputs.

The fuzzy inference system controller with both grid partition and fuzzy c-mean clustering were validated using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout. The simulation results show that the fuzzy inference system (FIS) is capable of following the specified route within the spciefied time deadline.

The following chapter will include proposing a new way of training the controller, where the controller will be trained online instead of the offline training as the previous controllers. The proposed controller will be developed using reinforcement learning with Deep Deterministic Policy Gradient (DDPG) Agents.

6 Reinforcement Learning for Aircraft Taxiing

6.1 Introduction

Machine learning has three broad categories, and Reinforcement learning (RL) is one of them. These categories are: unsupervised, supervised learning and reinforcement learning. Reinforcement learning [166] studies the way of learning of artificial and natural systems to predict their behaviour consequences in environments and optimise it in which their situation changes by taking actions with rewards and punishments.

"Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them." [167]

Unlike the other two learning frameworks, RL works with dynamic environment data, whereas the other two operate using a static dataset. It is not aimed to label data or cluster data, but the goal is finding the best sequence of actions which produce the optimal behaviour. The reinforcement learning agent is responsible for solving this problem by interacting with, exploring, and learning from the environment.

RL [168] studies the way that natural and artificial systems can learn to predict the consequences of and optimize their behaviour in environments in which actions lead them from one state or situation to the next, and can also lead to rewards and punishments. Such environments exist in many fields, such as psychology, control theory, economics, and ethology. Animals, from the most primitive to the most immodest, encounter a series of such optimisation problems [169], which apparently they are able to solve impressively. RL was born from operational research and mathematical psychology, providing solutions for quantitative and qualitative computational level models [170]. Computer programmes have

been successfully trained to play games at a higher level using reinforcement learning (RL) than the best human players in the world. These programmes try to take the best action in games with imperfect world information, large action and state spaces, and uncertainty.

The literature in this area has been extensively published and is the subject of many recent reviews (including [168, 171-174]). In addition, there is a rapid accumulation of literature on the aspects of optimal decision-making, which are partially related to situations involving slow amounts of social factors or information such as games [175-177]. After a short overview of the general reinforcement learning for control (see [178-187] for more review), focusing only on some of the many latest results relevant to RL and its neural instantiation.

Training an agent to finish a task within an unknown environment is the goal of reinforcement learning. Observations and a reward are received from the environment to the agent, and then the agent sends actions to the environment as shown in Figure 6.1. The reward measures the success of the action in achieving the goal of the task.

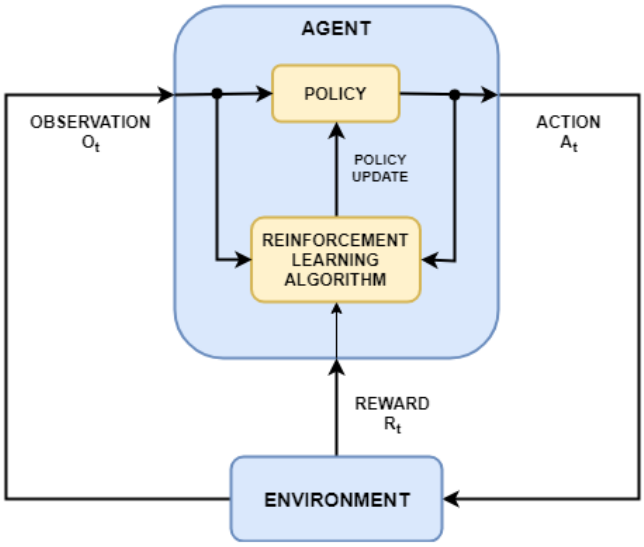


Figure 6.1:Reinforcement learning work cycle.

The main two components of the agent are: a policy and a learning algorithm.

- The **policy** works as a mapping that takes the observations from the environment and determines the best actions. Normally, The policy is a function approximator with tuneable parameters, such as a deep neural network.

- The **learning algorithm** always updates the parameters of the policy based on the reward, observations and actions. The goal of the learning algorithm is to find an optimal policy that maximises the cumulative reward received during the task.

We can say that reinforcement learning includes an agent that learns optimal behaviour through repeated trial and error interactions with the environment without human intervention.

6.2 Reinforcement Learning and Traditional Controls

The reinforcement learning and the control problem are similar; the same concept is represented by different methods and uses different terms. Both methods aim to determine the correct input of the system to generate the desired system behaviour.

Trying to design the best policy/controller that maps the environment/plant observed state to the best actions/the actuator commands. The observations from the environment are the state feedback signal, and the signal of the reference is connected to both the environment observations and the reward function as shown in Figure 6.2.

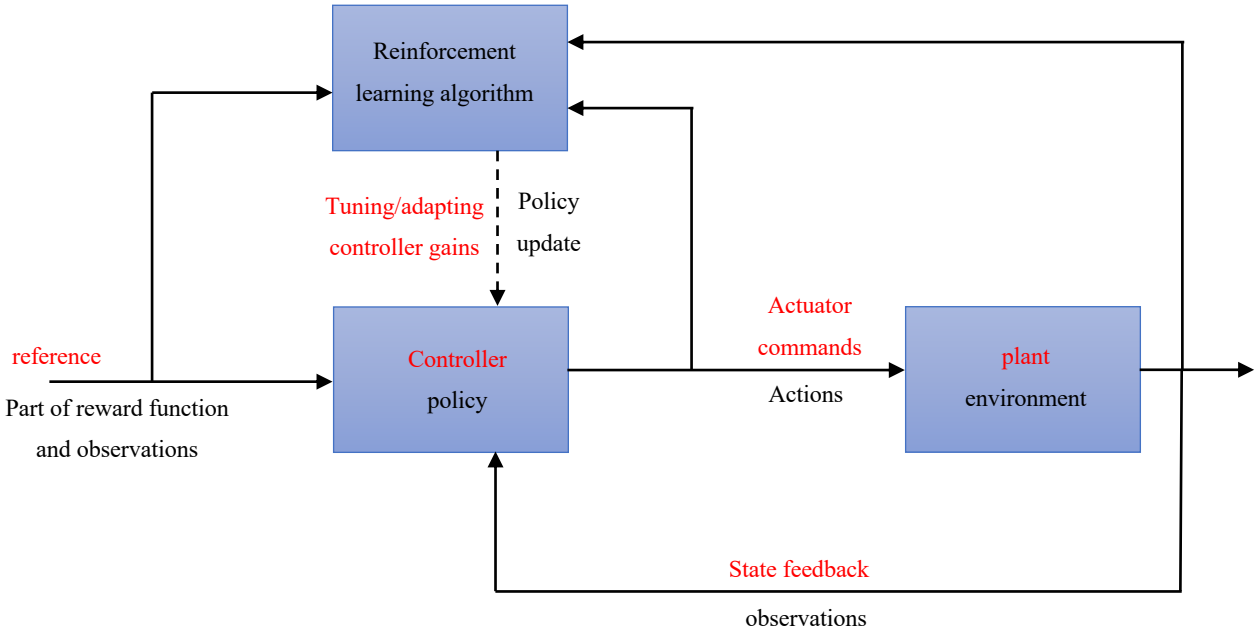


Figure 6.2:Reinforcement Learning and Traditional Controls.

6.3 The Workflow of Reinforcement Learning

The general reinforcement learning agent training steps are shown in Figure 6.3:

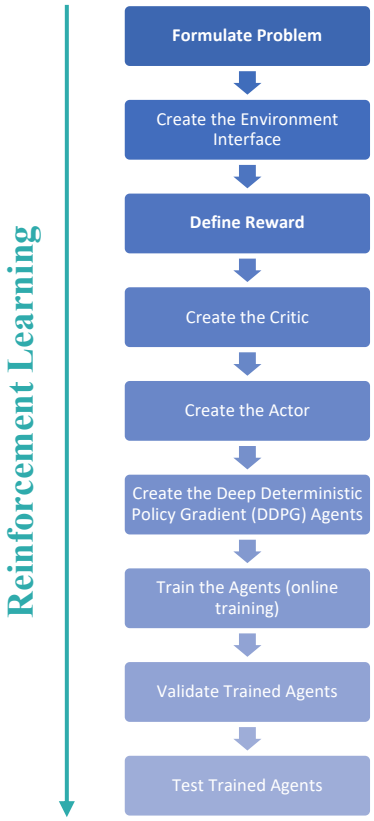


Figure 6.3:Reinforcement learning workflow

6.3.1 Formulate Problem

Defining what task that agent should learn that includes the must achieved primary and secondary goals and how it interacts with the environment. The goal in our problem is to make the agent find the best sequence of actions which make the aircraft to follow the reference speed to taxi an aircraft in an optimal manner.

6.3.2 Create the Environment Interface

Defining the environment in which the agent interacts, that include the environment dynamic model and the interface between the agent and environment. Figure 6.4 shows the configuration of the reinforcement learning control system. The aircraft inputs (throttle, rudder and braking) are manipulated by the inner control loop so that the references previously described in 3.2.1.1 are followed by the aircraft. Two sets of controllers exist in the inner loop control. The throttle and brake agents are responsible for manipulating throttle and braking inputs of the aircraft while the aircraft rudder is manipulated by the PID controller

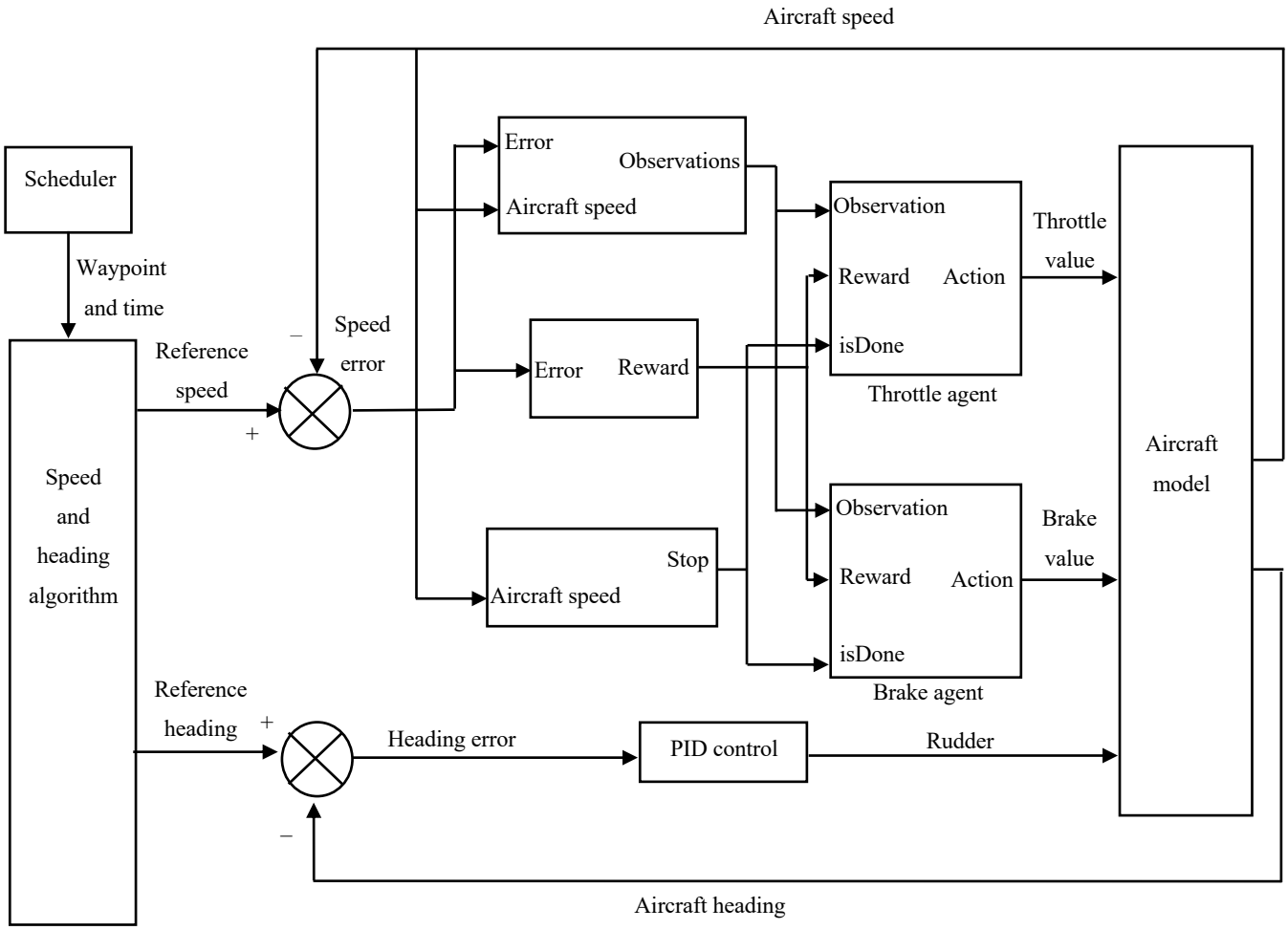


Figure 6.4:Reinforcement Learning control structure.

To create an environment model, the following must be defined:

- The signal of the action and observation where the it is used by the agent to be able to interact with the environment.
- The signal of the reward where it is used by the agent for its success measure. `obsInfo` defines the observation specification and `actInfo` defines the action specification.

```
obsInfo = rlNumericSpec([3 1],...
    LowerLimit=[-inf -inf 0  ]',...
    UpperLimit=[ inf  inf 20]');
obsInfo.Name = "observations";
obsInfo.Description = "integrated error, error, and measured aircraft's
speed";
```

```
actInfo = rlNumericSpec([1 1], ...
    LowerLimit= 0 ,...
    UpperLimit= 0.6);
actInfo.Name = "throttleValue";


actInfo2 = rlNumericSpec([1 1], ...
    LowerLimit= 0 ,...
    UpperLimit= 0.1);
actInfo2.Name = "brakeValue";
```

Build the environment interface object:

```
env =
rlSimulinkEnv("b747ReinforcementLearning2",["b747ReinforcementLearning2
/RL Agent1","b747ReinforcementLearning2/RL Agent2"],...
    {obsInfo,obsInfo},{actInfo,actInfo2});
```

`Ts` is sample time of the agent and `Tf` is the simulation time which are in seconds:

```
Ts = 1.0;
Tf = 50;
```

6.3.3 Define Reward

After setting the environment, the next step is to determine what the agent should do and how it will be rewarded for doing what is desired. Therefore, creating a reward function is required

so the policy is understood by learning algorithm when is getting better and eventually converges on the result we desire. Reward is a function that generates a scalar number which represents on the goodness of the agent when it is in a certain state and takes specific action as shown in Figure 6.5.

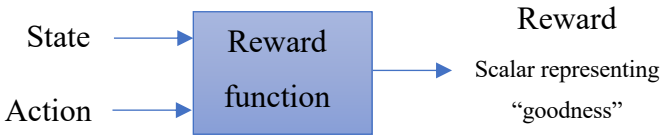


Figure 6.5: The reward function.

There is a similarity in the concept to a cost function that penalises poor performance of the system and increases the effort of the actuator. The difference is that cost functions try to minimise values, while reward functions try to maximise values. However, the same problem is solved because rewards is viewed as negative costs.

In reinforcement learning (RL), there is no restriction on the creation of rewards functions. Rewards can be calculated using thousands of parameters or from a nonlinear function. It entirely depends on what you need to effectively train your agents.

In the problem, a Gaussian function shown in Figure 6.6 was used as a reward function. The function and associated parameters are as follows:

$$f(x) = ae^{-\frac{(x-b)^2}{c^2}} \tag{6.1}$$

a = height of the curve's peak which is equal to 1
b = the position of the centre of the peak which is equal to 0
c = the standard deviation which is equal to 2
e= Euler's number
x = aircraft's speed error

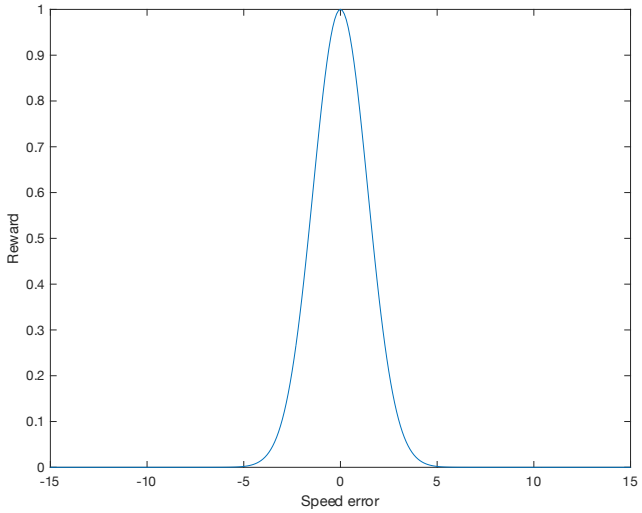


Figure 6.6:Gaussian function.

6.3.4 Create the Critic

A neural network is used to represent the value function (the critic) as Figure 6.7. The concept is similar to that of the table: the state observation and action are entered, and the value of that state/action pair will be returned by the neural network, choosing the action that will return the highest value is the policy. By time, the network converges slowly into a function that outputs the actual value of all actions in any continuous state space.

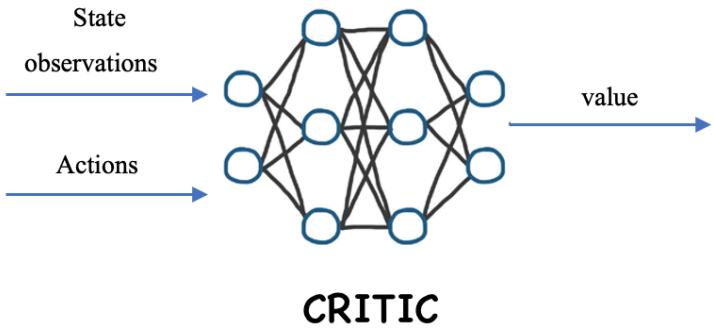


Figure 6.7:Value function-based learning.

To approximate the value function within the critic, a deep neural network is created. A network has two inputs (the observation and the action) and one output (the value). Three different paths are used; each path is specified as a row vector of layer objects as shown in Figure 6.8. The dimension of the action and observation spaces is obtained from the actInfo and obsInfo specifications.

```
statePath = [
    featureInputLayer(obsInfo.Dimension(1),Name="netObsIn")
    fullyConnectedLayer(50)
    reluLayer
    fullyConnectedLayer(25,Name="CriticStateFC2")];


actionPath = [
    featureInputLayer(actInfo.Dimension(1),Name="netActIn")
    fullyConnectedLayer(25,Name="CriticActionFC1")];


commonPath = [
    additionLayer(2,Name="add")
    reluLayer
    fullyConnectedLayer(1,Name="CriticOutput")];


criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);


criticNetwork = connectLayers(criticNetwork, ...
    "CriticStateFC2", ...
    "add/in1");
criticNetwork = connectLayers(criticNetwork, ...
    "CriticActionFC1", ...
    "add/in2");
```

The critic network configuration is viewed:

```
figure
plot(criticNetwork)
```

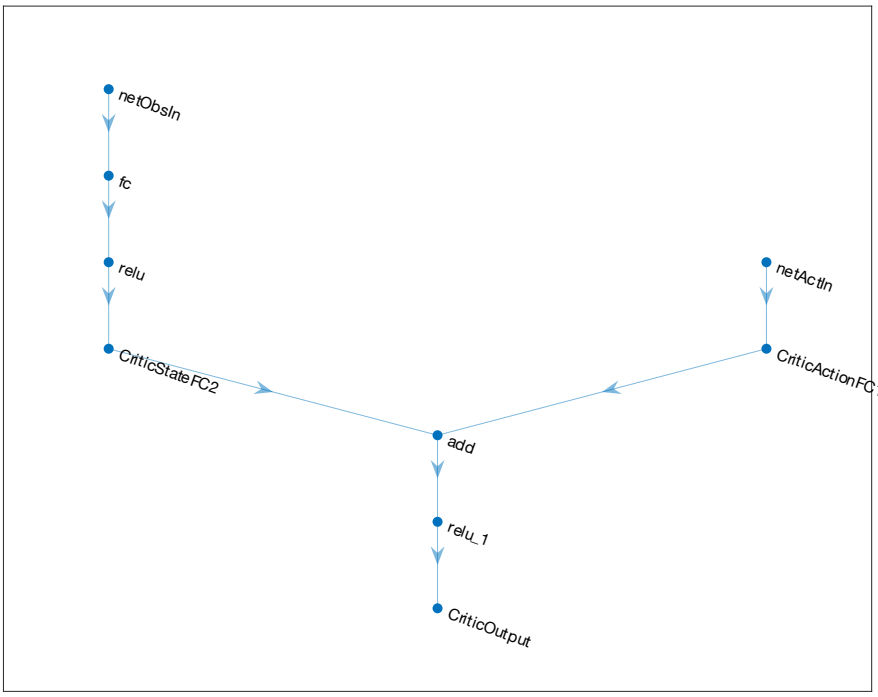


Figure 6.8:The critic network.

The network is convertd to a `dlnetwork` object and its properties is summarized:

```
criticNetwork = dlnetwork(criticNetwork);
summary(criticNetwork)
```

To create the critic approximator object we need: the specified deep neural network, observation and action, and network inputs names.

```
critic1 = rlQValueFunction(criticNetwork,obsInfo,actInfo, ...
    ObservationInputNames="netObsIn", ...
    ActionInputNames="netActIn");


critic2 = rlQValueFunction(criticNetwork,obsInfo,actInfo2, ...
    ObservationInputNames="netObsIn", ...
    ActionInputNames="netActIn");
```

6.3.5 Create the Actor

The actor is a network as in Figure 6.9, that takes the current state as input and gives the best action as output, as seen with the policy function method. Then the critic estimates the value of the pair of this action that the actor took and the state.

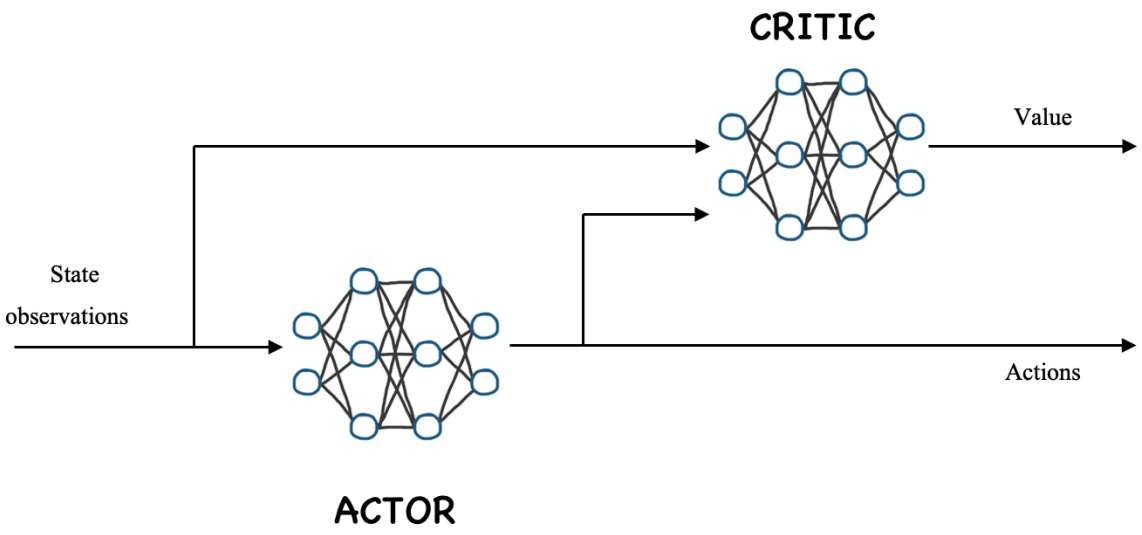


Figure 6.9:Actor-critic learning algorithms.

To approximate the policy within the actor, a deep neural network is created. The network has one input (the observation) and one output (the action) of a row vector of layer objects as shown in Figure 6.10. The dimension of the action and observation spaces is obtained from the `obsInfo` and `actInfo` specifications.

```
actorNetwork = [
    featureInputLayer(obsInfo.Dimension(1))
    fullyConnectedLayer(3)
    tanhLayer
    fullyConnectedLayer(actInfo.Dimension(1))
    ];
```

The actor network configuration is viewed:

```
figure
plot(actorNetwork)
```

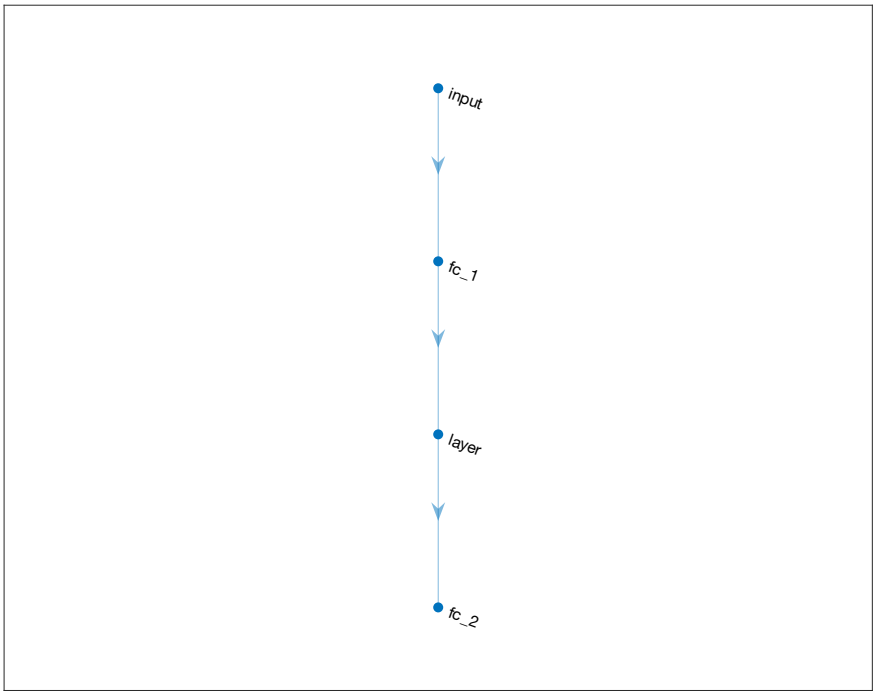


Figure 6.10:The actor network .

The network is converted to a `dlnetwork` object and its properties is summarized:

```
actorNetwork = dlnetwork(actorNetwork);
summary(actorNetwork)
```

To create the actor approximator object we need: the specified deep neural network, observation and action.

```
actor1 = rlContinuousDeterministicActor(actorNetwork,obsInfo,actInfo);
actor2 = rlContinuousDeterministicActor(actorNetwork,obsInfo,actInfo2);
```

rlContinuousDeterministicActor

In this object a function approximator is implemented and used as a deterministic actor with a continuous action space within a reinforcement learning agent. The input for this continuous

deterministic actor is an environment state, and the action is retuned as output that maximises the expected discounted cumulative long-term reward, thus a deterministic policy is implemented.

The Learning Cycle of The Actor-Critic

The actor in Figure 6.11 is responsible for choosing the action and sending it to the environment, as does the policy function algorithm.

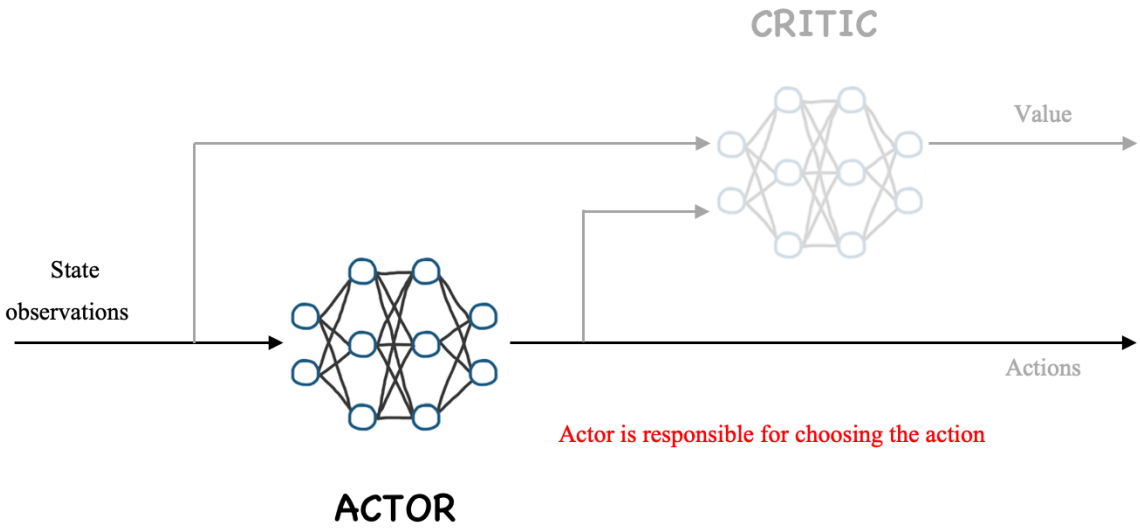


Figure 6.11:Actor-critic learning algorithm.

The critic is responsible for predicting the value of that action as shown in Figure 6.12.

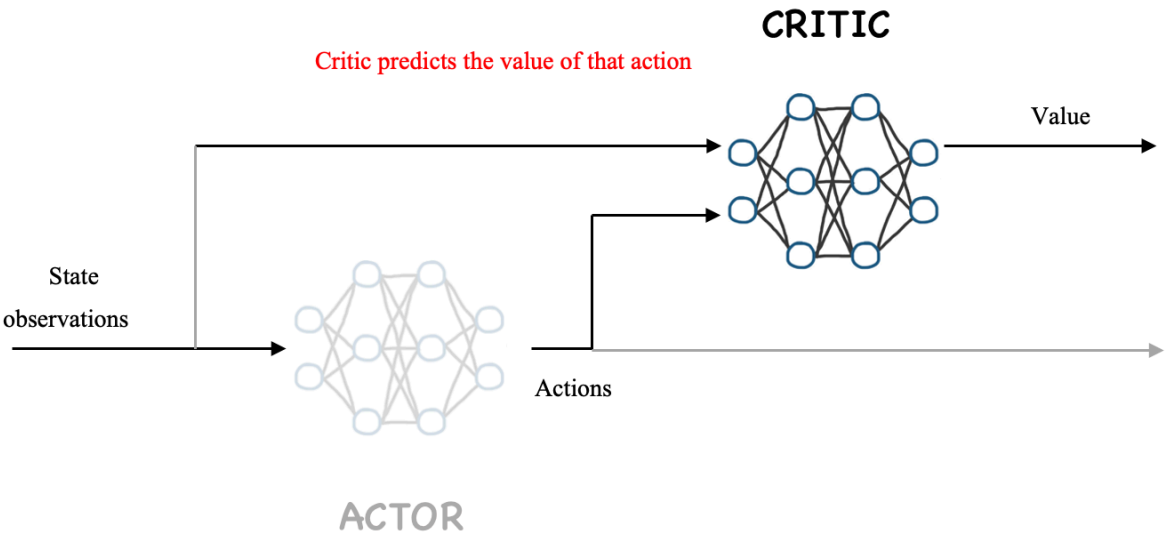


Figure 6.12:Actor-critic learning algorithm.

The reward from the environment is used by the critic to determine the value prediction accuracy as shown in Figure 6.13. The error is the difference between the new estimated value of the previous state and the old value of the previous state from the critic network. The error indicates whether things have improved or deteriorated more than expected, so the critic can understand.

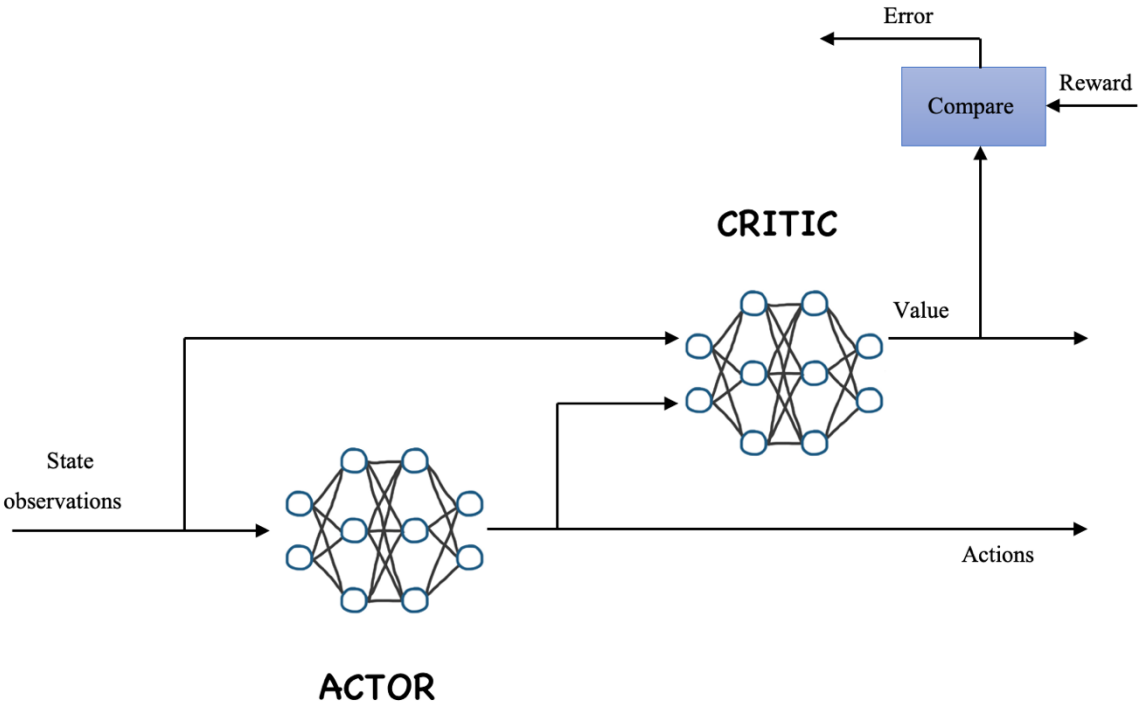


Figure 6.13:Actor-critic learning algorithm.

The critic is updated by This error as shown in Figure 6.14, therefore that next time it can predict better when it is in this state.

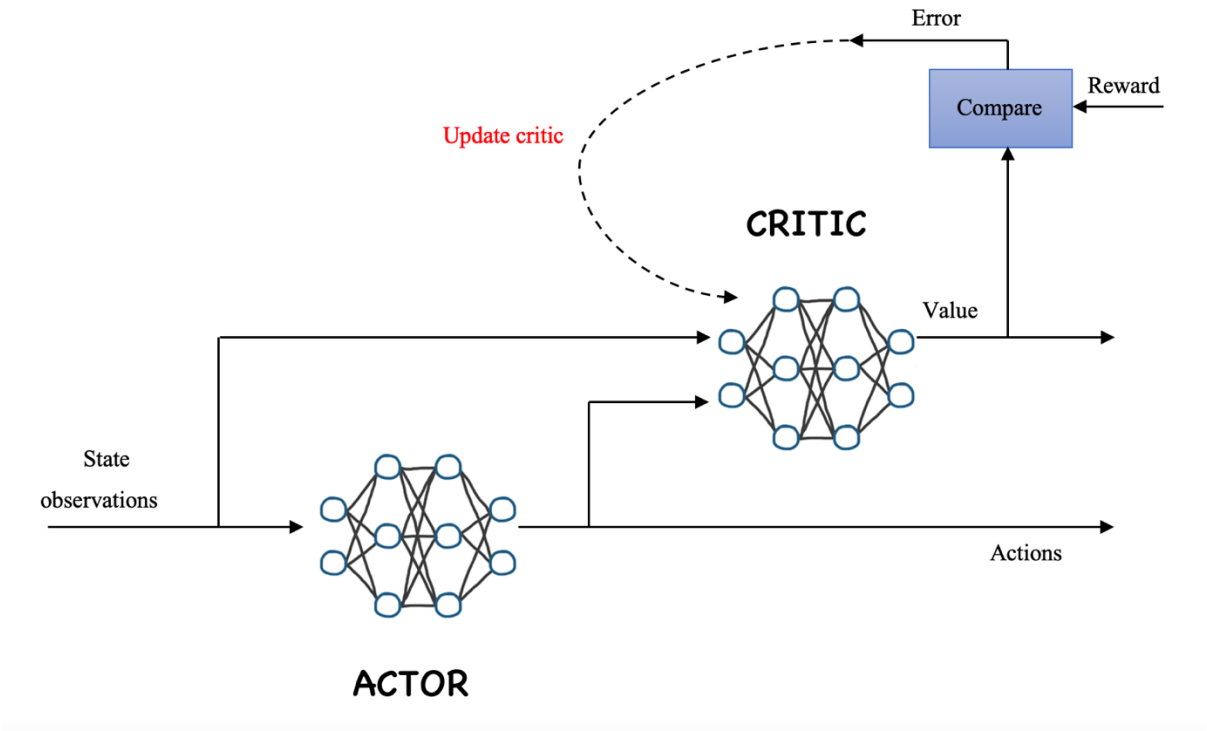


Figure 6.14:Actor-critic learning algorithm.

With the critic response, the actor is also updated as shown in Figure 6.15. Thus, in the future, it will adjust its probability of taking that action again. Therefore, the reward slope is ascended by the policy.

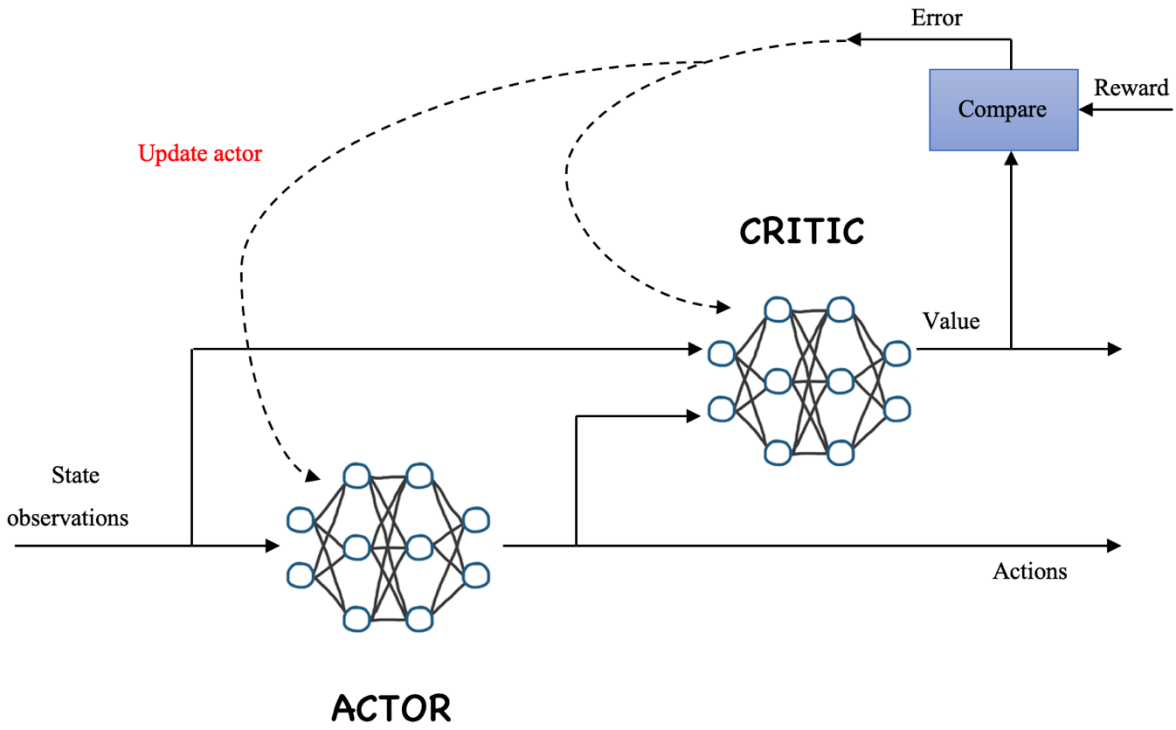


Figure 6.15:Actor-critic learning algorithm.

The policy of an actor-critic is used by many of the learning algorithms. Here, we used the deep deterministic policy gradient algorithm (DDPG). Neural networks are used to represent the actor and critic where the optimal behaviour is learnt as shown in Figure 6.16. The feedback from the critic is used by the actor to learn the correct actions to take and to determine whether it is a good or bad action. The rewards received are used by the critic to learn the value function, so the action taken from the actor is properly criticised.

With the method of actor-critic, the value function and policy benefit are taken by the agent. Continuous state and action spaces are both handled by the Actor-critic, and learning is speeded up when the returned reward has high variance. Continuous spaces of state and action can be handled by actor-critics.

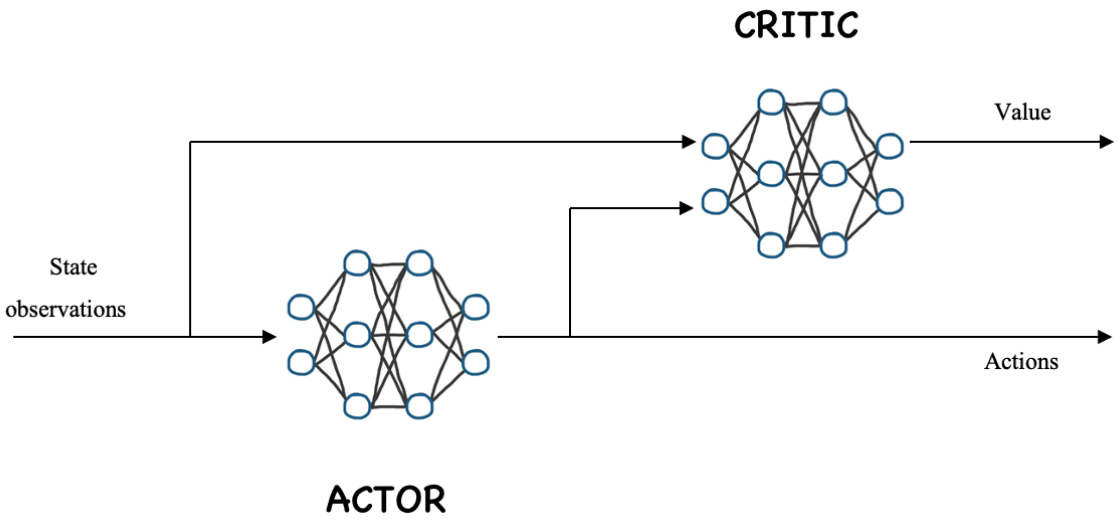


Figure 6.16:Actor-critic learning algorithm.

6.3.6 Create the DDPG Agents

- The deep deterministic policy gradient (DDPG) algorithm is a method of reinforcement leaning with an actor-critic, online, off-policy, model-free where an optimal policy is computed that maximizes the long-term reward. Only the continuous action space can be used. In our problem two DDPG agents were created, one for the throttle and one for the break.

The specified actor and critic approximator objects are used to create the DDPG agents:

```
agent1 = rlDDPGAgent(actor1,critic1);
agent2 = rlDDPGAgent(actor2,critic2);
```

Options for the agent, the actor, and the critic are specified using dot notation:

```
agent1.SampleTime = Ts;


agent1.AgentOptions.TargetSmoothFactor = 1e-3;
agent1.AgentOptions.DiscountFactor = 1.0;
agent1.AgentOptions.MiniBatchSize = 64;
agent1.AgentOptions.ExperienceBufferLength = 1e6;


agent1.AgentOptions.NoiseOptions.Variance = 0.3;
agent1.AgentOptions.NoiseOptions.VarianceDecayRate = 1e-5;


agent1.AgentOptions.CriticOptimizerOptions.LearnRate = 1e-03;
agent1.AgentOptions.CriticOptimizerOptions.GradientThreshold = 1;
```

```
agent1.AgentOptions.ActorOptimizerOptions.LearnRate = 1e-04;
agent1.AgentOptions.ActorOptimizerOptions.GradientThreshold = 1;


%%%%%%%%%


agent2.SampleTime = Ts;


agent2.AgentOptions.TargetSmoothFactor = 1e-3;
agent2.AgentOptions.DiscountFactor = 1.0;
agent2.AgentOptions.MiniBatchSize = 64;
agent2.AgentOptions.ExperienceBufferLength = 1e6;


agent2.AgentOptions.NoiseOptions.Variance = 0.3;
agent2.AgentOptions.NoiseOptions.VarianceDecayRate = 1e-5;


agent2.AgentOptions.CriticOptimizerOptions.LearnRate = 1e-03;
agent2.AgentOptions.CriticOptimizerOptions.GradientThreshold = 1;
agent2.AgentOptions.ActorOptimizerOptions.LearnRate = 1e-04;
agent2.AgentOptions.ActorOptimizerOptions.GradientThreshold = 1;
```

6.3.7 Train the Agents (online training)

The agents approximators are trained using the defined environment, agents learning algorithm and reward.

To train the agents, the training options are first specified:

- Each training is run for at most `5000` episodes. Each episode lasts for at most `ceil(Tf/Ts)` (that is `50`) time steps.
- Training stopoed when the agents receives an average cumulative reward close to 50. At this point, the agent can control the aircraft's speed to track the reference speed.

```
trainOpts = rlMultiAgentTrainingOptions(...
    MaxEpisodes=5000, ...
    MaxStepsPerEpisode=ceil(Tf/Ts), ...
    ScoreAveragingWindowLength=20, ...
    Verbose=false, ...
    Plots="training-progress",...
    StopTrainingCriteria="AverageReward",...
    StopTrainingValue=50);
```

The agents are trained using the `train` function:

```
trainingStats = train([agent1,agent2],env,trainOpts);
```

The result of training the agents is shown in Figure 6.17.

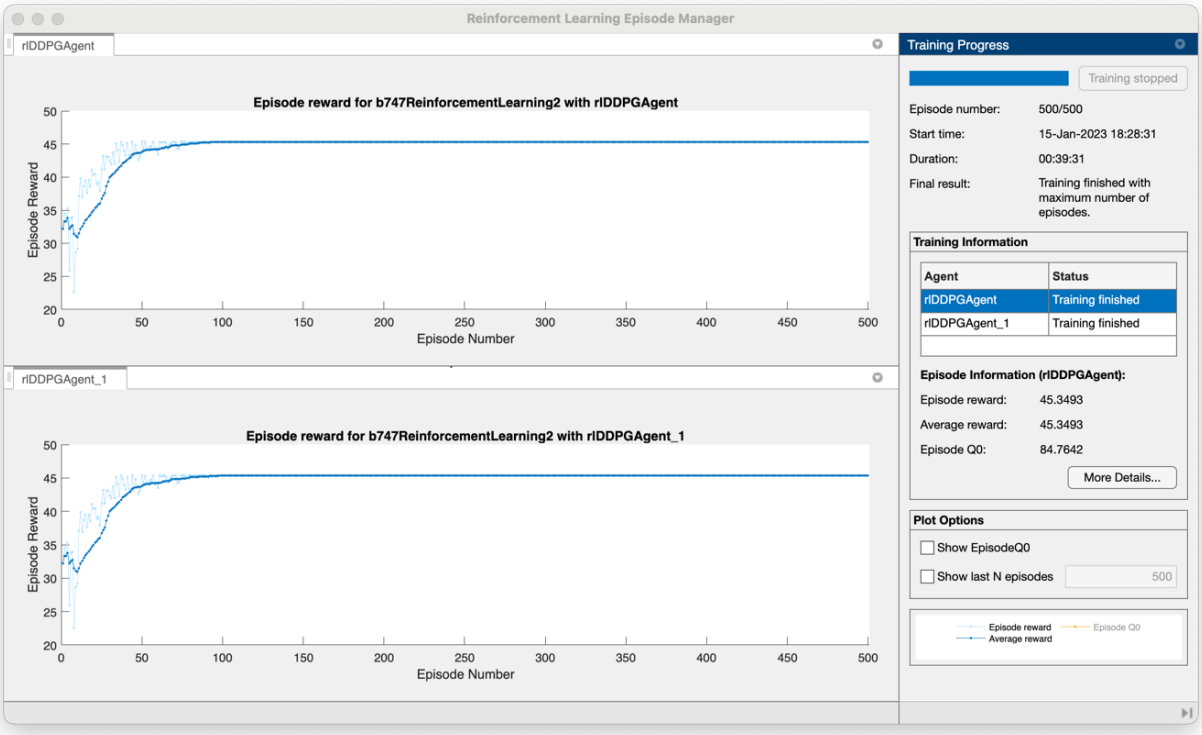


Figure 6.17:Episode reward.

6.3.8 Validate Trained Agents

To evaluate the trained agents performance, multitude of reference speeds to be followed with different initial speeds are used, which are:

Initial speed=5

- The reference speed is set to be: reference speed= 3, the result is shown in Figure 6.18.

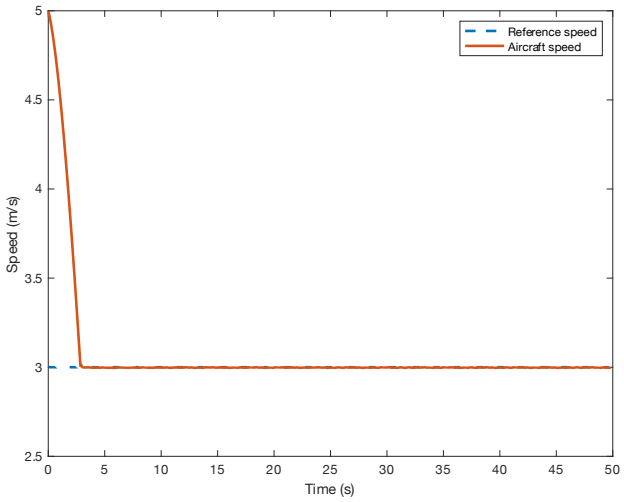


Figure 6.18:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 8, the result is shown in Figure 6.19.

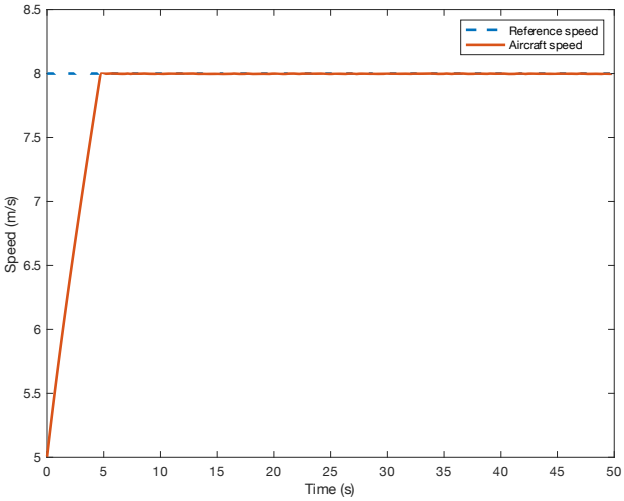


Figure 6.19:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 10, the result is shown in Figure 6.20.

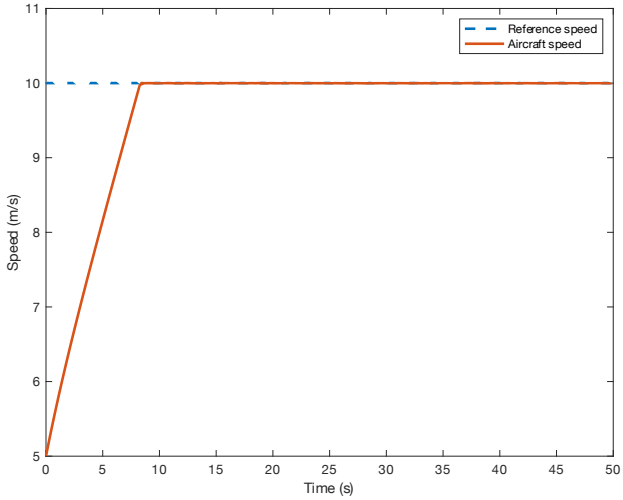


Figure 6.20:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 14, the result is shown in Figure 6.21.

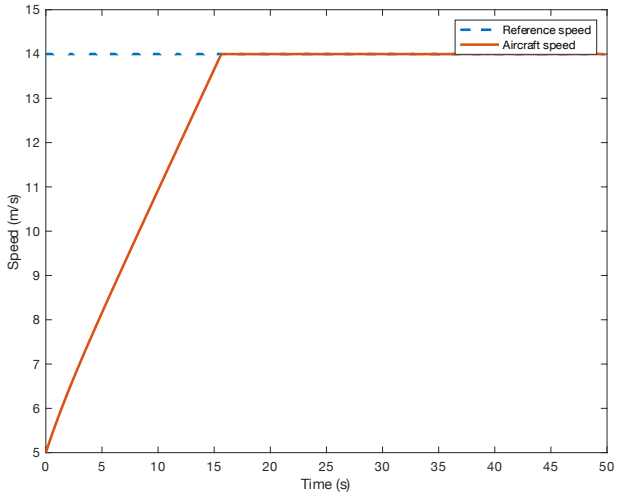


Figure 6.21:Aircraft reference and actual speeds.

Initial speed=3

- The reference speed is set to be: reference speed= 2, the result is shown in Figure 6.22.

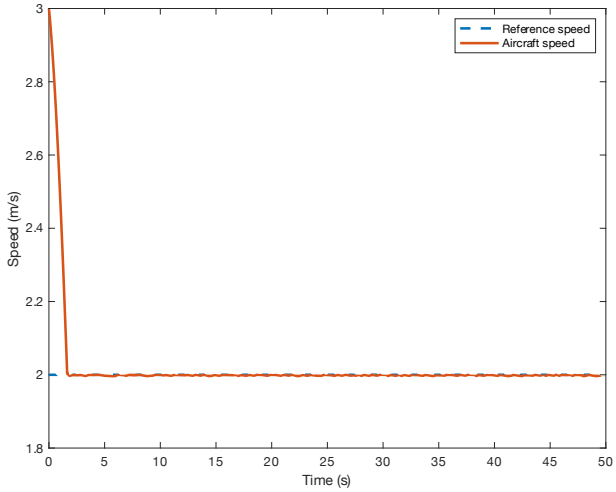


Figure 6.22:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 8, the result is shown in Figure 6.23.

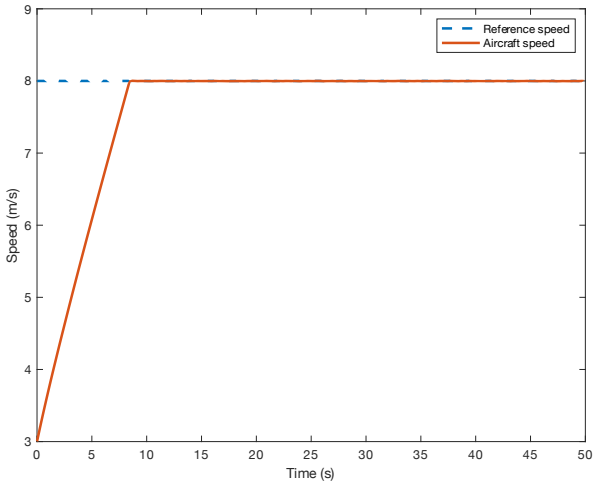


Figure 6.23:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 10, the result is shown in Figure 6.24.

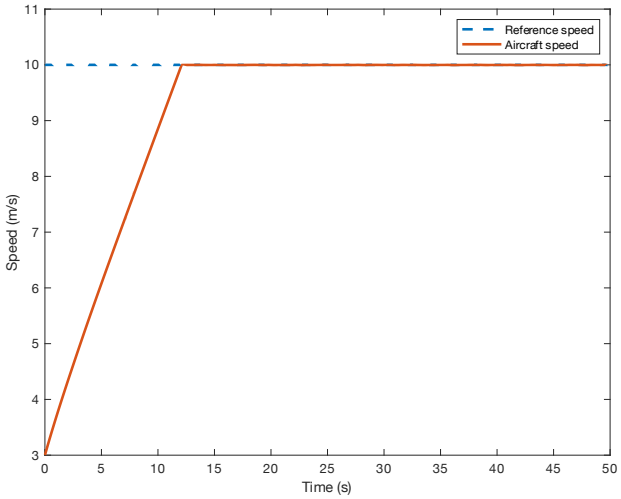


Figure 6.24:Aircraft reference and actual speeds.

- The reference speed is set to be: reference speed= 14, the result is shown in Figure 6.25.

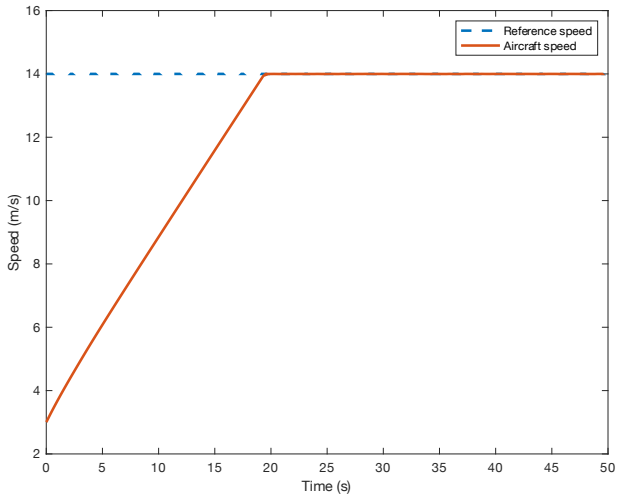


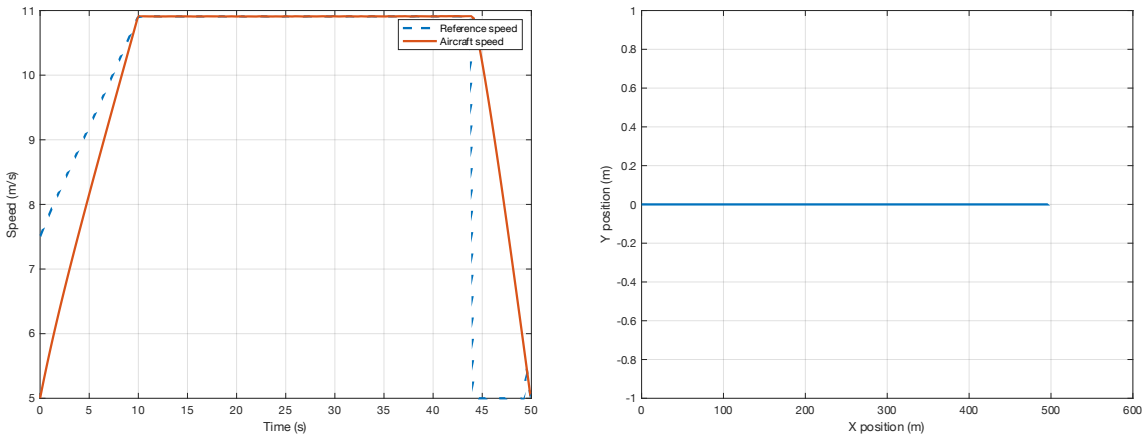
Figure 6.25:Aircraft reference and actual speeds.

6.3.9 Test Trained Agents

a. Taxiing Run Along A straight Line Path

In order to test the RL agents, a straight line path scenario is used which include path from the origin to the point (500 m, 0 m) in 50 s with an initial speed of 5 m/s and a final speed of 5 m/s. Figure 6.26 shows the result of testing the proposed controller on this scenario. Indeed, the RL

agents can be seen to be capable of following the specified route within the specified time deadline.



(a) (b)

Figure 6.26:(a) Aircraft reference and actual speeds of the straight line scenario where the final speed is specified (5 m/s); (b) X,Y position of the aircraft

b. Taxiing run along a circle path

A cirle layout is a scenario used for aircraft taxiing test. The origin is taken to be the starting point of aircraft. The layout is divided into four different segments from this point. Figure 6.27 shows the plotts of the attained cartesian positions for the circle path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.3. The taxiing run results show that the RL agents are capable of following the specified route within the spciefied time deadline which is 180 s.

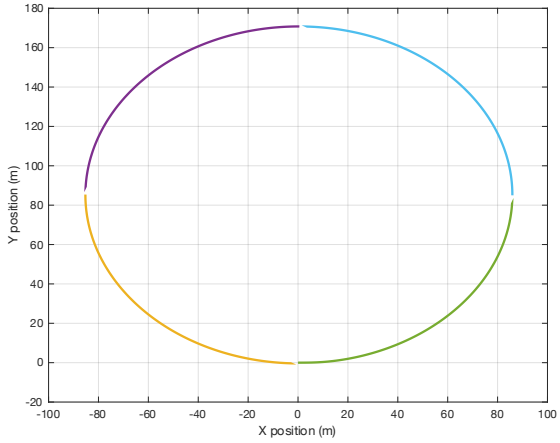


Figure 6.27:The aircraft test for the circle layout and its cartesian positions.

c. Taxiing run along a rectangular path

A rectangular layout is a scenario used for aircraft taxiing test. The end left of straight part of the base of this rectangle is the starting point of aircraft. The layout is divided into eight (8) different segments from this point. Figure 6.28 shows the plotts of the attained cartesian positions for the rectangular path different segments. The coordinates of the waypoints for the simulated segments, the time deadlines and the turning angle are shown in Table 3.4. The taxiing run results show that the RL agents are capable of following the specified route within the spciefied time deadline which is 465 s.

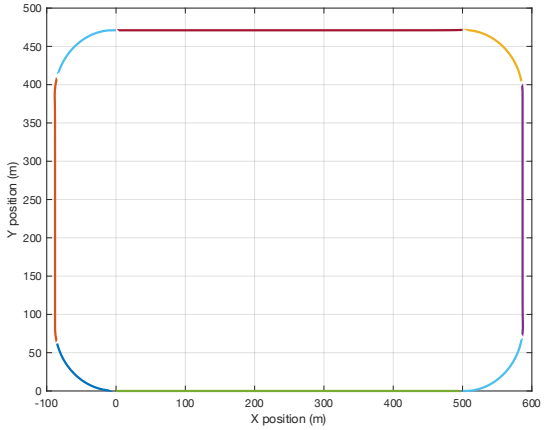


Figure 6.28:The aircraft test for the rectangular layout and its cartesian positions.

6.4 Summary

In this chapter, a new way of training the controller has been proposed. Looking to train the controller online instead of offline like the previous controllers. The way of designing this type of controllers was using reinforcement learning where it works with dynamic environment data, whereas the other two operate using a static dataset. It was the fourth controller applied for aircraft taxiing. The Deep Deterministic Policy Gradient (DDPG) Agents were used for reinforcement learning algorithm. Two DDPG agents were built, one for the throttle and one for the brake.

The chapter started by an introduction to the reinforcement learning and its applications. Then the difference between the reinforcement learning and the traditional control has been clarified. Also, the workflow of reinforcement learning has been listed and detailed, where it consists of 9 steps as follow: formulate problem, create the environment interface, define reward, create critic, create actor, create the deep deterministic policy gradient agents, train the agents, validate the trained agents and finally test the trained agents.

The reinforcement learning agents were tested using several scenarios that involve a straight line scenario, a circle layout and a rectangular layout. The simulation results show that reinforcement learning agents are capable of following the specified route within the specified time deadline.

7 Conclusions and Future Work

7.1 Conclusions

New optimal navigation of taxiing aircraft have been presented in this project. The newly proposed approaches are generic despite the fact that the associated algorithms have been developed and tested on the specific Boeing 747-100 aircraft MATLAB-SIMULINK model. The proposed approaches used Artificial Intelligence (AI) so that taxiing is performed in an optimal manner.

The contributions of this thesis can be summarized as follows:

a) The simulating a non-linear Boeing 747-100 aircraft model in MATLAB Simulink.

This study used the NASA reports to drive the non-linear Boeing 747-100 aircraft model simulated in MATLAB Simulink. A state-space model formulation is the mathematical model that is used which is derived from a rigid body general equation of motion. The Boeing 747 was first produced in 1967, it is a jumbo transport aircraft with four fanjet. For the taxiing process, the Boeing 747 is designed with a rudder and a tiller that control the aircraft direction. It also has a movable stabilizer with four ailerons, inward and outward Krueger flaps but these are assumed not to be used during taxiing so they are not included in the modelling. The two inner engines are responsible for generating the thrust force. The state space form has been used to represent the aircraft rigid body equations (with 13 states). In particular, the states include the standard 12-dimensional variable (which governs the body rates and accelerations) as well as an additional variable that has been called ‘‘pre-thrust” and represents the dynamics of the engine.

b) The Design of aircraft navigation system and the PID controller for aircraft taxiing.

The first controller was developed for aircraft taxiing was the PID controller. The aircraft navigation system consists of several blocks that allows taxiing the aircraft between the way points in an optimal manner. The references (speed and heading) are calculated online by speed and heading algorithm that exists in the outer loop controller. Then, these references are followed by the aircraft. The work of this algorithm was explained step by step with an example of moving the aircraft from along a straight line of distance 500 m in 50s" with all the required equations. The turning segments were also outlined were the navigation algorithm is used but the distance is calculated in a different manner.

The inner loop control consists of three sets of PID controllers to manipulate the throttle, brakes, and rudder respectively. The proportional controller was only used for braking. The PID gains were optimally tuned based on fuel consumed using Genetic Algorithm (GA) in MATLAB for aircraft taxiing scenarios.

With different taxiing scenarios, control problems arisen as the PID control's gains for one scenario are sub-optimal for the other scenario. The clustering strategy on the lists of scenarios into six clusters where each cluster had an optimized cluster centre was used to tackle the problem. The PID gains of that centres were utilized by the cluster members which therefore led to a significantly better performance than using just one controller for all scenarios in term of fuel consumption.

To validate the PID controller, several taxiing scenarios were used that include straight line path, a circle layout and a rectangular layout. The results proved the capability of the PID controller to follow the specified route within the specified time deadline.

The capabilities of the PID controller was enhanced using the artificial neural networks, where the PID controller needs to change the PID gains each time when the scenario changed. The artificial neural networks overcame this deficiency because it is considered one of the most effective learning methods that has a strong ability to learn and generalize.

c) The design of artificial neural networks controller for aircraft taxiing.

The artificial neural networks controller was developed as it was the second controller applied for aircraft taxiing. To build the network, backpropagation neural network was used as it is a powerful learning technique. The built network consists of: 2 inputs (speed error, change in speed error), 2 hidden layers with 10 and 5 neurons respectively and finally 2 outputs (throttle value, break value). *Trainlm* was used for training function, *learngdm* for adaptation learning function and for performance *MSE* was used. *Tansig* was the transfer function for the hidden layer neurons and *purelin* was the transfer function for the output layer neurons. The steps for designing the neural network model for this project has been provided, which includes: data collection, build the network, initialize weights and biases, train the network and finally use the network.

To validate the artificial neural networks controller, several taxiing scenarios were used that include straight line path, a circle layout and a rectangular layout. The results proved the capability of the artificial neural networks controller to follow the specified route within the specified time deadline.

d) The designing a Fuzzy Inference System (FIS) model for aircraft taxiing.

A new proposed fuzzy inference system (FIS) model controller was built using the MATLAB ANFIS toolbox. It was the third controller applied for aircraft taxiing. ANFIS was mainly used to tunes (adjusts) the membership function parameters that allows the fuzzy systems to learn from the data they are modelling. The FIS was built using two ways: grid partition and fuzzy c-mean clustering.

In grid partition, the data grid partitioning was used to generate Sugeno-type FIS of a single-output. There were two controllers were built: one for the throttle and one for the break. The throttle controller has two inputs (error and change of error) and one output (throttle value). Each input has 4 membership functions of gaussian type and the output membership function is constant type. The system has16 fuzzy rules. The break controller has two inputs (error and change of error) and one output (break value). Each input has 4 membership functions of gaussian type and the output membership function is constant type. The system has16 fuzzy rules.

In fuzzy c-mean clustering, The data fuzzy c-mean clustering is used to generate Sugeno-type FIS of a single-output. There were two controllers were built: one for the throttle and one for the break. The throttle controller has two inputs (error and change of error) and one output (throttle value). Each input has 17 clusters/membership functions of gaussian type. The output membership function is constant type. The total number of rules are 17 rules. The break controller has two inputs (error and change of error) and one output (break value) . Each input has 60 clusters/membership functions of guession type. The output membership function is constant type. The total number of rules are 60 rules.

To validate the FIS controller, several taxiing scenarios were used that include straight line path, a circle layout and a rectangular layout. The results proved the capability of the FIS controller to follow the specified route within the specified time deadline.

e) The design of an online controller using reinforcement learning.

An online controller was proposed for aircraft taxiing that works with dynamic environment data. It was the fourth controller applied using reinforcement learning. The goal was finding the best sequence of actions which produce the optimal behaviour. The reinforcement learning agent was responsible for solving this problem by interacting with, exploring, and learning from the environment. The Deep Deterministic Policy Gradient (DDPG) Agents were used for reinforcement learning algorithm. Two DDPG agents were built, one for the throttle and one for the brake. The reward measured the success of the action in achieving the goal of the task.

The main two components of the agent are: a policy (takes the observations from the environment and determines the best actions) and a learning algorithm (finds an optimal policy that maximises the cumulative reward received). There were 9 steps to train reinforcement learning agent which are :

- Formulate problem: the goal in our problem is to make the agent find the best sequence of actions which make the aircraft to follow the reference speed.
- Create the environment interface: defining the interface between the agent and environment which includes the observation signal and the action signal.

- Define reward: reward is a function that generates a scalar number which represents on the goodness of the agent. In our problem a Gaussian function was used as a reward function where the input is aircraft's speed error, curve's peak equal to 1, the position of the centre of the peak equal to 0 and standard deviation which equal to 2.
- Create critic: a neural network was used to represent the value function where the inputs to the network are the observation and the action and the output is the value.
- Create actor: The actor is a network that takes the current state as input and gives the best action as output.
- Create the deep deterministic policy gradient (DDPG) agent: it is a method of reinforcement leaning where an optimal policy was computed that maximizes the long-term reward. In our problem two DDPG agents were created, one for the throttle and one for the break.
- Train the agents: the agents were trained online using the defined environment, agents learning algorithm and reward.
- Validate the trained agents: the performance of the trained agents was validated with multitude of reference speeds to be followed with different initial speeds are used.
- Test the trained agents: To test the reinforcement learning agents, several taxiing scenarios were used that include straight line path, a circle layout and a rectangular layout. The results proved the capability of the reinforcement learning agents to follow the specified route within the specified time deadline.

The proposed approaches can be utilized for two scenarios. The first one to serve as decision support systems. In this case, they can advise the pilot on the best control inputs. The second one to be used for a fully automated taxiing aircraft. When tested on different taxiing problem, the proposed approaches can run successfully with all taxiing scenarios. The results in this work represent a significant step forward towards guiding aircrafts from gate to gate in airports all around the world. In addition, the proposal in this research aids in the reduction of aircraft fuel consumption and congestion as a whole. This research work is particularly salient towards contributing significantly and positively to greenhouse gas emission. Future studies will attempt to quantify the effects of the many sources of uncertainties in aircraft ground movement, such as pilot behaviour and different aircraft configurations.

7.2 Future Work

These are some topics that could be considered for the future:

1- Self-organising fuzzy logic control (SOFLC)

The controller in [188] is able to automatically improve and develop, and is used for non-linear and dynamic processes. It is called learning controller or adaptive because it has the ability to change based on the process it is controlling until it converges to a desired result. The SOFLC performs two main tasks simultaneously: (a) monitor the controlled process while issuing the appropriate control action and (b) use these control actions' results to improve them further. In the future, the SOFLC could be used to control the aircraft's speed to follow the reference speed, as illustrated in Figure 7.1.

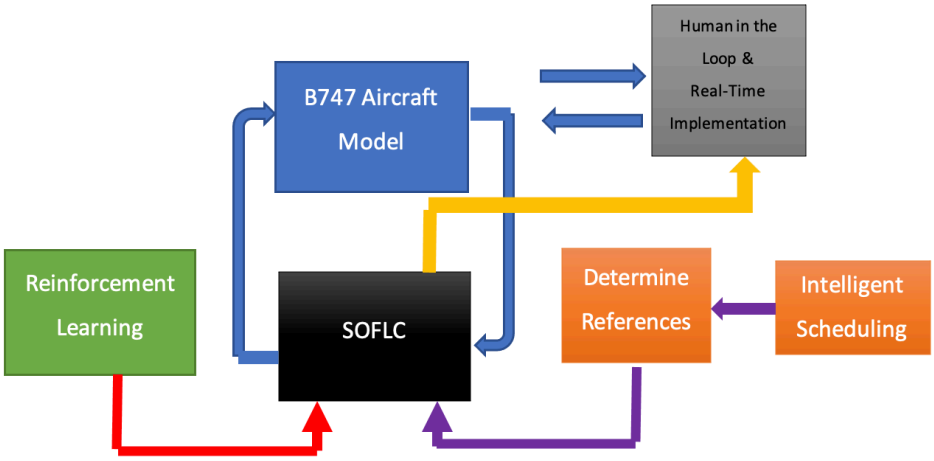


Figure 7.1:The integration of SOFLC in our system.

2- Intelligent scheduling

The intelligent scheduler concerns with allocating optimal rout to taxiing aircraft from parking stands to runways. It consist of scheduling the aircraft between airport facilities in an effective way and without any conflicts. When the aircraft arrives, it is routed from the runway to the stand whereas when it depart, it is routed from its park to the runway. The taxiways described as a network of roads which link all the facilities of the airport. This schedule is then realistically followed by the aircraft where the pilot is involved in the loop as the controller. The ant colony optimization is used in scheduling problems that requires finding the optimal rout. In the future, the intelligent scheduler could be developed using the ant colony optimization to find the optimal rout to taxiing aircraft. This route with the reference speed will be then sent to controller to be followed as shown in Figure 7.2.

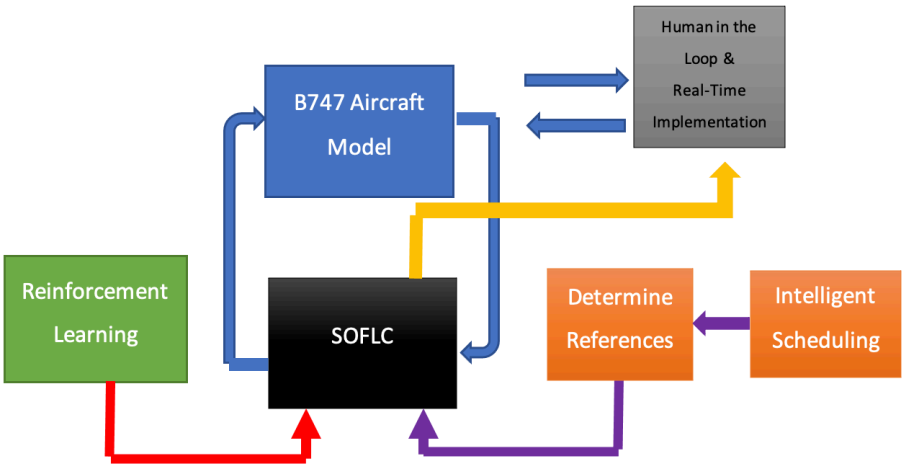


Figure 7.2:The integration of intelligent scheduling in our system.

3- Follow the green concept for active guidance of piloted aircraft

The concept of Follow-the-Greens (FtG) guidance in [189] is used in current airport surface operations which provides taxiing based on 4DT. According to the assigned 4DTs, the piloted aircraft is guided by adjusting the position of ground green lamps navigation dynamically based on the assigned 4DTs. In the future, the FtG guidance could be integrated in the project and used to guide the aircraft, so the controller will follow the 4DT based on the green lamps as shown in Figure 7.3.

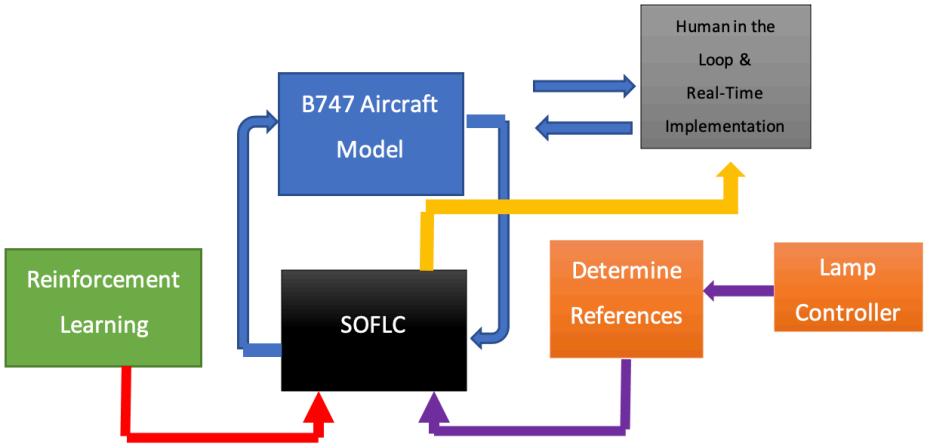


Figure 7.3:The integration of Lamp Controller in our system.

4- Flight trajectory prediction using deep learning and reinforcement learning

The aim of Personalized Route Recommendation (PRR) is generating user-specific route suggestions from a source to a destination from the user's queries. The neural networks is proposed in [190] which has the ability to automatically learn classic heuristic algorithm's cost functions for the PRR task. Complex trajectory patterns are difficult to be captured by Machine learning methods while learnable neural networks work effectively in capturing the complex data characteristics. In future, the PRR could be integrated in the project and used to generate the route based on the pilot query using the neural networks as shown in Figure 7.4.

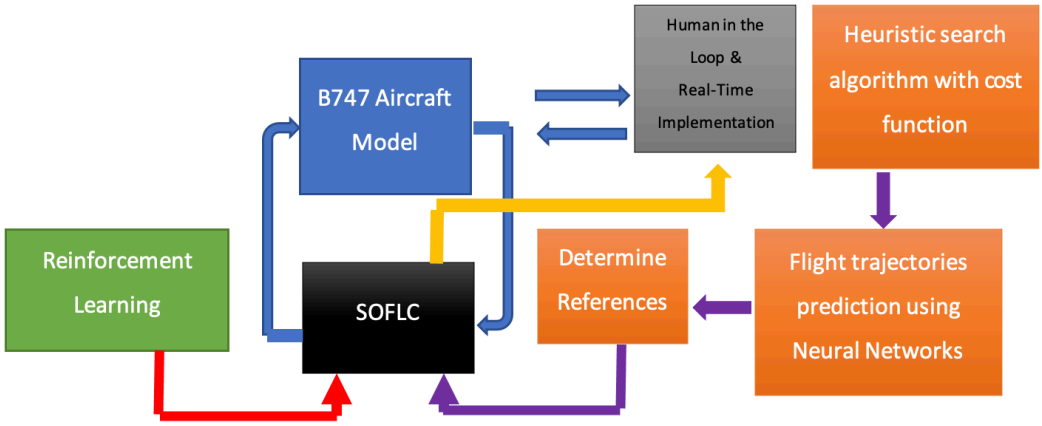


Figure 7.4:The integration of Neural Networks for 4D trajectories prediction in our system.

5- Deep reinforcement learning combined with exact shortest path algorithms

The "sequence to sequence learning" method is used in [191]. The flight plan is the input sequence and the flight trajectory is the output sequence. The sequential learning problem can be solved by employing an encoder-decoder recurrent neural network. The flight plan is learnt by the encoder and then this flight plan is translated by the decoder into a full 4D trajectory. In the future, the sequence to sequence learning could be integrated in the project and used to generate the 4D trajectory based on the pilot plan using the encoder-decoder recurrent neural network as shown in Figure 7.5.

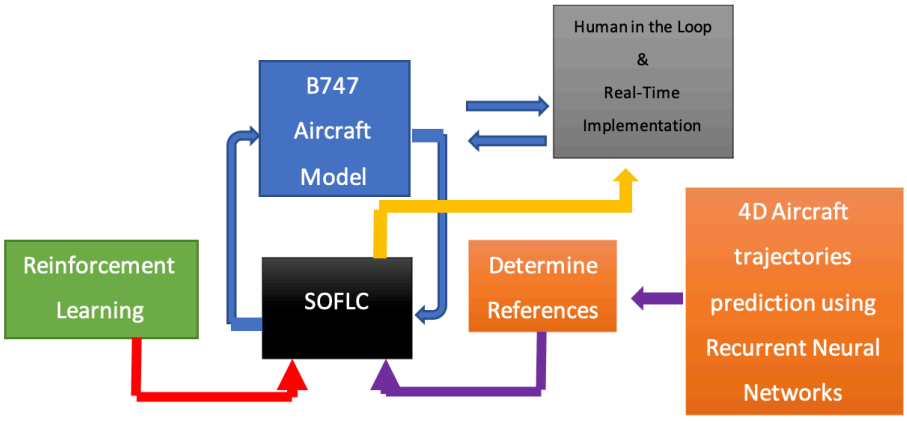


Figure 7.5:The integration of Recurrent Neural Networks for 4D trajectories prediction in our system.

6- Real-time implementation

In real time implementation the system will be implemented on a B747 Aircraft Simulator based in Cranfield University with real pilots as shown in Figure 7.6 So, the pilots will be monitored in real time by measuring their vital signs: Hear Rate Variability (HRV), brain waves, pupil diameter to monitor their psychosocial stress level.

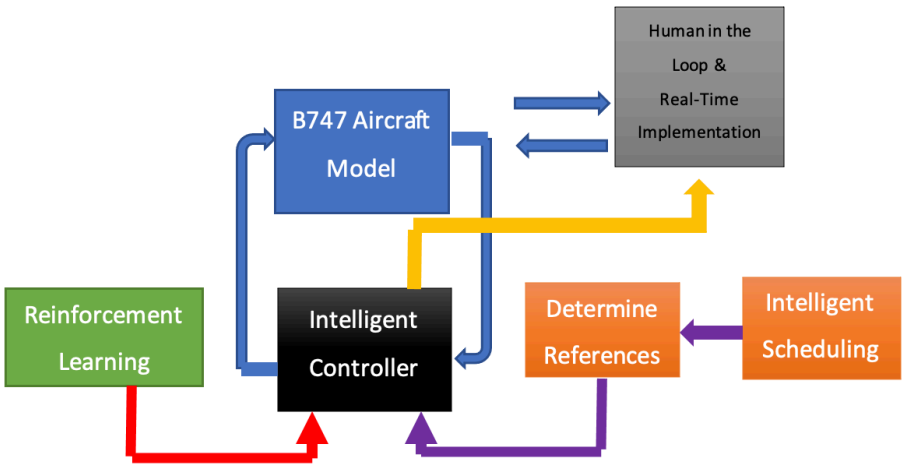


Figure 7.6:Real time implementation of the proposed system.

References

1. Maceková, B. and Ľ. Fábry, *strategic guidance in support of the execution of the european atm master plan.* Acta Avionica, 2012. **14**(24).
2. Planning, J., *Concept of operations for the next generation air transportation system*, in *Technical Report*. 2007, Citeseer.
3. Obajemu, O., et al. *Fuzzy Modelling of Fuel Consumptions and Emissions for Optimal Navigation of a BOEING-747 Aircraft*. in *2020 IEEE Aerospace Conference*. 2020.
4. Brooker, P., *SESAR and NextGen: Investing in new paradigms.* The Journal of Navigation, 2008. **61**(2): p. 195.
5. Clare, G. and A. Richards. *Airport ground operations optimizer*. in *8th Innovative Research Workshop & Exhibition Proceedings, Eurocontrol*. 2009.
6. Cook, A., et al., *New perspectives for air transport performance.* Third SESAR Innovation Days, 2013.
7. Chen, J., et al., *Toward a More Realistic, Cost-Effective, and Greener Ground Movement Through Active Routing—Part I: Optimal Speed Profile Generation.* IEEE Transactions on Intelligent Transportation Systems, 2015. **17**(5): p. 1196-1209.
8. Graham, A., *Managing airports: an international perspective. (Japanese version)*. 2010, Chuokeizai-Sha Inc.
9. Deonandan, I. and H. Balakrishnan. *Evaluation of strategies for reducing taxi-out emissions at airports*. in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*. 2010.
10. Haus, S., et al. *Control theoretic concept for intuitive guidance of pilots during taxiing*. in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*. 2011. IEEE.
11. Li, J., et al. *Departure Scheduling and Taxiway Path Planning under Uncertainty*. in *AIAA Aviation 2019 Forum*. 2019.
12. Weiszer, M., J. Chen, and P. Stewart, *A real-time Active Routing approach via a database for airport surface movement.* Transportation Research Part C: Emerging Technologies, 2015. **58**: p. 127-145.
13. Ravizza, S., et al., *The trade-off between taxi time and fuel consumption in airport ground movement.* Public Transport, 2013. **5**(1-2): p. 25-40.
14. Guépet, J., et al., *Integration of aircraft ground movements and runway operations.* Transportation research part E: logistics and transportation review, 2017. **104**: p. 131-149.
15. Brownlee, A.E., et al. *A rolling window with genetic algorithm approach to sorting aircraft for automated taxi routing*. in *Proceedings of the Genetic and Evolutionary Computation Conference*. 2018.
16. Brownlee, A.E.I., et al., *A fuzzy approach to addressing uncertainty in Airport Ground Movement optimisation.* Transportation Research Part C: Emerging Technologies, 2018. **92**: p. 150-175.
17. Psaraftis, H.N., *A dynamic programming approach to the aircraft sequencing problem*. 1978, Cambridge, Mass.: Massachusetts Institute of Technology, Flight ….

18. Bayen, A.M., et al. *An approximation algorithm for scheduling aircraft with holding time*. in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*. 2004. IEEE.
19. Lee, H. and H. Balakrishnan. *Fuel cost, delay and throughput tradeoffs in runway scheduling*. in *2008 American Control Conference*. 2008. IEEE.
20. Lieder, A., D. Briskorn, and R. Stolletz, *A dynamic programming approach for the aircraft landing problem with aircraft classes.* European Journal of Operational Research, 2015. **243**(1): p. 61-69.
21. Leese, R., *The sequencing of aircraft departures.* 2001.
22. Brinton, C.R. *An implicit enumeration algorithm for arrival aircraft*. in *[1992] Proceedings IEEE/AIAA 11th Digital Avionics Systems Conference*. 1992. IEEE.
23. Beasley, J.E., et al., *Scheduling aircraft landings—the static case.* Transportation science, 2000. **34**(2): p. 180-197.
24. Sölveling, G. and J.-P. Clarke, *Scheduling of airport runway operations using stochastic branch and bound methods.* Transportation Research Part C: Emerging Technologies, 2014. **45**: p. 119-137.
25. D'Ariano, A., et al. *Optimal sequencing of aircrafts take-off and landing at a busy airport*. in *13th International IEEE Conference on Intelligent Transportation Systems*. 2010. IEEE.
26. Bianco, L., P. Dell'Olmo, and S. Giordani, *Minimizing total completion time subject to release dates and sequence-dependentprocessing times.* Annals of Operations Research, 1999. **86**(0): p. 393-415.
27. Shi, W., et al., *A heuristic algorithm for solving the aircraft landing scheduling problem with a landing sequence division.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2019. **102**(8): p. 966-973.
28. Cheng, V.H., L.S. Crawford, and P. Menon. *Air traffic control using genetic search techniques*. in *Proceedings of the 1999 IEEE international conference on control applications (Cat. No. 99CH36328)*. 1999. IEEE.
29. Hu, X.-B. and E. Di Paolo, *Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling.* IEEE Transactions on Intelligent Transportation Systems, 2008. **9**(2): p. 301-310.
30. Randall, M. *Scheduling aircraft landings with ant colony optimisation*. in *IASTED International Conference on Artificial Intelligence and Soft Computing*. 2002.
31. Wu, L.-J., et al. *Multi-runway aircraft arrival scheduling: A receding horizon control based ant colony system approach*. in *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019. IEEE.
32. Bäuerle, N., O. Engelhardt-Funke, and M. Kolonko, *On the waiting time of arriving aircrafts and the capacity of airports with one or two runways.* European Journal of Operational Research, 2007. **177**(2): p. 1180-1196.
33. Atkin, J.A., et al., *Hybrid metaheuristics to aid runway scheduling at London Heathrow airport.* Transportation Science, 2007. **41**(1): p. 90-106.
34. Van Leeuwen, P., H. Hesselink, and J. Rohling, *Scheduling aircraft using constraint satisfaction.* Electronic notes in theoretical computer science, 2002. **76**: p. 252-268.
35. Atkin, J.A.D., *On-line decision support for take-off runaway scheduling at London Heathrow Airport*. 2008, University of Nottingham.
36. Cheng, V.H.L. and G.D. Sweriduk. *Trajectory design for aircraft taxi automation to benefit trajectory-based operations*. in *2009 7th Asian Control Conference*. 2009.

37. Wu, C., et al., *Train speed trajectory optimization with on-board energy storage device.* IEEE Transactions on Intelligent Transportation Systems, 2018. **20**(11): p. 4092-4102.
38. Bakowski, D.L., et al., *NextGen Surface Trajectory-Based Operations (STBO): Evaluating Conformance to a Four-dimensional Trajectory (4DT).* Procedia Manufacturing, 2015. **3**: p. 2458-2465.
39. Urvoy, C., et al. *Concept for a human centered 4d surface guidance system*. in *Proceedings of the 29th Conference of the European Association for Aviation Psychology*. 2010.
40. Re, F. *Viability and state of the art of environmentally friendly aircraft taxiing systems*. in *2012 Electrical Systems for Aircraft, Railway and Ship Propulsion*. 2012. IEEE.
41. Khadilkar, H. and H. Balakrishnan, *Estimation of aircraft taxi fuel burn using flight data recorder archives.* Transportation Research Part D: Transport and Environment, 2012. **17**(7): p. 532-537.
42. Allerton, D., *Principles of flight simulation*. 2009: John Wiley & Sons.
43. Hanke, C.R., *The Simulation of a Large Jet Transport Aircraft. Volume 1-Mathematical Model.* 1971.
44. Hanke, C.R. and D.R. Nordwall, *The simulation of a jumbo jet transport aircraft. Volume 2: Modeling data.* 1970.
45. Krüger, W., et al., *Aircraft landing gear dynamics: simulation and control.* Vehicle System Dynamics, 1997. **28**(2-3): p. 119-158.
46. Young, D., *Aircraft landing gears—the past, present and future.* Proceedings of the Institution of Mechanical Engineers, Part D: Transport Engineering, 1986. **200**(2): p. 75-92.
47. Allerton, D., *The impact of flight simulation in aerospace.* The Aeronautical Journal, 2010. **114**(1162): p. 747-756.
48. Pritchard, J., *Overview of landing gear dynamics.* Journal of aircraft, 2001. **38**(1): p. 130-137.
49. Boril, J., et al. *Aviation simulation training in the Czech air force*. in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. 2015. IEEE.
50. Wood, G., M. Blundell, and S. Sharma, *A low parameter tyre model for aircraft ground dynamic simulation.* Materials & Design, 2012. **35**: p. 820-832.
51. Kapadoukas, G.G. and A. Self, *A taxonomy of aircraft ground handling modes.* Aircraft Engineering and Aerospace Technology, 2000.
52. Barnes, A.G. and T.J. Yager. *Enhancement of aircraft ground handling simulation capability*. 1998. AGARD.
53. Daniels, J.N., *A method for landing gear modeling and simulation with experimental validation*. 1996.
54. Hajiloo, A. and W. Xie, *The stochastic robust model predictive control of shimmy vibration in aircraft landing gears.* Asian Journal of Control, 2015. **17**(2): p. 476-485.
55. Esmailzadeh, E. and K. Farzaneh, *Shimmy vibration analysis of aircraft landing gears.* Journal of Vibration and Control, 1999. **5**(1): p. 45-56.
56. Krüger, W.R. and M. Morandini, *Recent developments at the numerical simulation of landing gear dynamics.* CEAS Aeronautical Journal, 2011. **1**: p. 55-68.
57. Somieski, G., *Shimmy analysis of a simple aircraft nose landing gear model using different mathematical methods.* Aerospace Science and Technology, 1997. **1**(8): p. 545-555.
58. Currey, N.S., *Aircraft landing gear design: principles and practices*. 1988: Aiaa.

59. Raymer, D., *Aircraft design: a conceptual approach*. 2012: American Institute of Aeronautics and Astronautics, Inc.
60. Ladda, V. and H. Struck. *Operational loads on landing gear*. in *AGARD Conference Proceedings*. 1991.
61. Denti, E. and D. Fanteria, *Models of wheel contact dynamics: An analytical study on the in-plane transient responses of a brush model.* Vehicle System Dynamics, 2000. **34**(3): p. 199-225.
62. Vaishnav, P. *Low-hanging fruit? The costs and benefits of reducing fuel burn and emissions from taxiing aircraft*. in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2012.
63. Zhang, Y., X. Li, and C. Li. *The establishment of the UAV'ground taxiing model*. in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. 2016. IEEE.
64. Bo, L., J. Zongxia, and W. Shaoping. *Research on modeling and simulation of aircraft taxiing rectification*. in *2006 IEEE Conference on Robotics, Automation and Mechatronics*. 2006. IEEE.
65. Ferguson, S.W., *A mathematical model for real time flight simulation of a generic tilt-rotor aircraft.* NASA CR-166536, 1988. **1**.
66. Gillespie, T., *Fundamentals of vehicle dynamics*. 2021: SAE international.
67. Wong, J.Y., *Theory of ground vehicles*. 2022: John Wiley & Sons.
68. Khapane, P.D., *Simulation of asymmetric landing and typical ground maneuvers for large transport aircraft.* Aerospace Science and Technology, 2003. **7**(8): p. 611-619.
69. Wang, Q. and R.F. Stengel, *Robust nonlinear flight control of a high-performance aircraft.* IEEE Transactions on Control Systems Technology, 2004. **13**(1): p. 15-26.
70. Obajemu, O., et al., *Real-time four-dimensional trajectory generation based on gain-scheduling control and a high-fidelity aircraft model.* Engineering, 2021.
71. Duke, E.L., R.F. Antoniewicz, and K.D. Krambeer, *Derivation and definition of a linear aircraft model*. 1988.
72. *Transit research group report*.
73. Allerton, D., *Principles of flight simulation*. Vol. 27. 2009: John Wiley & Sons.
74. Organization, M.I.C.A., *ICAO Committee on Aviation Environmental Protection Working Group. Aircraft Engine exhaust emissions databank.*
75. Obajemu, O., M. Mahfouf, and J.W.F. Catto, *A New Fuzzy Modeling Framework for Integrated Risk Prognosis and Therapy of Bladder Cancer Patients.* IEEE Transactions on Fuzzy Systems, 2018. **26**(3): p. 1565-1577.
76. Obajemu, O. and M. Mahfouf, *A Dirichlet Process Based Type-1 and Type-2 Fuzzy Modeling for Systematic Confidence Bands Prediction.* IEEE Transactions on Fuzzy Systems, 2019. **27**(9): p. 1853-1865.
77. Araki, M., *Control systems, robotics and automation—vol ii—pid control.* Kyoto University, Japan, 2010.
78. Bennett, S., *Development of the PID controller.* IEEE Control Systems Magazine, 1993. **13**(6): p. 58-62.
79. Bennett, S., *A brief history of automatic control.* IEEE Control Systems Magazine, 1996. **16**(3): p. 17-25.

80. Sahu, P.C., R.C. Prusty, and B. Sahoo, *Modified sine cosine algorithm-based fuzzy-aided PID controller for automatic generation control of multiarea power systems.* Soft Computing, 2020. **24**(17): p. 12919-12936.
81. Jin, X., et al., *Simulation of hydraulic transplanting robot control system based on fuzzy PID controller.* Measurement, 2020. **164**: p. 108023.
82. Ali, M.N., et al., *Resilient design of robust multi-objectives PID controllers for automatic voltage regulators: D-decomposition approach.* IEEE Access, 2021. **9**: p. 106589-106605.
83. Azar, A.T., et al. *Implementation of PID controller with PSO tuning for autonomous vehicle*. in *International Conference on Advanced Intelligent Systems and Informatics*. 2019. Springer.
84. Altbawi, S.M.A., et al., *Optimal design of Fractional order PID controller based Automatic voltage regulator system using gradient-based optimization algorithm.* Journal of King Saud University-Engineering Sciences, 2021.
85. Daraz, A., et al., *Modified PID controller for automatic generation control of multi-source interconnected power system using fitness dependent optimizer algorithm.* PloS one, 2020. **15**(11): p. e0242428.
86. Behera, A., et al., *A novel cascaded PID controller for automatic generation control analysis with renewable sources.* IEEE/CAA Journal of Automatica Sinica, 2019. **6**(6): p. 1438-1451.
87. Aboelhassan, A., et al., *Design and Implementation of model predictive control based PID controller for industrial applications.* Energies, 2020. **13**(24): p. 6594.
88. Noordin, A., et al., *Adaptive PID controller using sliding mode control approaches for quadrotor UAV attitude and position stabilization.* Arabian Journal for Science and Engineering, 2021. **46**: p. 963-981.
89. Dogruer, T. and M.S. Can, *Design and robustness analysis of fuzzy PID controller for automatic voltage regulator system using genetic algorithm.* Transactions of the Institute of Measurement and Control, 2022. **44**(9): p. 1862-1873.
90. Pilla, R., et al. *Fuzzy PID controller for automatic generation control of interconnected power system tuned by glow-worm swarm optimization*. in *Applications of Robotics in Industry Using Advanced Mechanisms: Proceedings of International Conference on Robotics and Its Industrial Applications 2019 1*. 2020. Springer.
91. Pilla, R., A.T. Azar, and T.S. Gorripotu, *Impact of flexible AC transmission system devices on automatic generation control with a metaheuristic based fuzzy PID controller.* Energies, 2019. **12**(21): p. 4193.
92. Li, X.-G., et al., *Characterizing PID controllers for linear time-delay systems: a parameter-space approach.* IEEE Transactions on Automatic Control, 2020. **66**(10): p. 4499-4513.
93. Osadchyy, V., O. Nazarova, and M. Olieinikov. *The Research of a Two-Mass System with a PID Controller, Considering the Control Object Identification*. in *2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES)*. 2021. IEEE.
94. Suid, M. and M. Ahmad, *Optimal tuning of sigmoid PID controller using Nonlinear Sine Cosine Algorithm for the Automatic Voltage Regulator system.* ISA transactions, 2022. **128**: p. 265-286.

95. Verma, B. and P.K. Padhy, *Robust fine tuning of optimal PID controller with guaranteed robustness.* IEEE Transactions on Industrial Electronics, 2019. **67**(6): p. 4911-4920.
96. Suseno, E.W. and A. Ma'arif, *Tuning of PID controller parameters with genetic algorithm method on DC motor.* International Journal of Robotics and Control Systems, 2021. **1**(1): p. 41-53.
97. Zeng, D., et al., *Research on improved auto-tuning of a PID controller based on phase angle margin.* Energies, 2019. **12**(9): p. 1704.
98. Paul, S., et al., *Fuzzy tuned PID controller for envisioned agricultural manipulator.* International Journal of Automation and Computing, 2021. **18**(4): p. 568-580.
99. Habib, M.R., et al. *PID controller based automatic solar powerdriven grass cutting machine*. in *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*. 2019. IEEE.
100. Veerasamy, V., et al., *Design of single-and multi-loop self-adaptive PID controller using heuristic based recurrent neural network for ALFC of hybrid power system.* Expert Systems with Applications, 2022. **192**: p. 116402.
101. Salman, G.A., A.S. Jafar, and A.I. Ismael, *Application of artificial intelligence techniques for LFC and AVR systems using PID controller.* International Journal of Power Electronics and Drive Systems, 2019. **10**(3): p. 1694.
102. Weiszer, M., E.K. Burke, and J. Chen, *Multi-objective routing and scheduling for airport ground movement.* Transportation Research Part C: Emerging Technologies, 2020. **119**: p. 102734.
103. Weiszer, M., et al. *A heuristic approach to greener airport ground movement*. in *2014 IEEE congress on evolutionary computation (CEC)*. 2014. IEEE.
104. Chen, J., et al., *Toward a more realistic, cost-effective, and greener ground movement through active routing: a multiobjective shortest path approach.* IEEE Transactions on Intelligent Transportation Systems, 2016. **17**(12): p. 3524-3540.
105. Zhang, T., et al., *An online speed profile generation approach for efficient airport ground movement.* Transportation Research Part C: Emerging Technologies, 2018. **93**: p. 256-272.
106. Obajemu, O., et al., *Real-time four-dimensional trajectory generation based on gain-scheduling control and a high-fidelity aircraft model.* Engineering, 2021. **7**(4): p. 495-506.
107. Mitchell, T.M., *Machine learning.* 1997.
108. *Deep Learning Artificial Neural Network Intelligence Machine Neuron - Circuit - Networking Transparent PNG*. 2018, PNGHUT: webpage.
109. Yu, D.-Y. *Parallel robots pose accuracy compensation using artificial neural networks*. in *2008 IEEE International Conference on Mechatronics and Automation*. 2008. IEEE.
110. Nguyen, H., et al., *Prediction of blast-induced ground vibration in an open-pit mine by a novel hybrid model based on clustering and artificial neural network.* Natural Resources Research, 2020. **29**: p. 691-709.
111. Bianchi, F.M., D. Grattarola, and C. Alippi. *Spectral clustering with graph neural networks for graph pooling*. in *International conference on machine learning*. 2020. PMLR.
112. Müller, E., *Graph clustering with graph neural networks.* Journal of Machine Learning Research, 2023. **24**: p. 1-21.

113. Chattopadhyay, A., P. Hassanzadeh, and S. Pasha, *Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data.* Scientific reports, 2020. **10**(1): p. 1317.
114. Santos, T.T., et al., *Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association.* Computers and Electronics in Agriculture, 2020. **170**: p. 105247.
115. Bandara, K., C. Bergmeir, and S. Smyl, *Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach.* Expert systems with applications, 2020. **140**: p. 112896.
116. He, Y., L. Dai, and H. Zhang, *Multi-branch deep residual learning for clustering and beamforming in user-centric network.* IEEE communications letters, 2020. **24**(10): p. 2221-2225.
117. Khatir, S., et al., *An improved Artificial Neural Network using Arithmetic Optimization Algorithm for damage assessment in FGM composite plates.* Composite Structures, 2021. **273**: p. 114287.
118. Chegari, B., et al., *Multi-objective optimization of building energy performance and indoor thermal comfort by combining artificial neural networks and metaheuristic algorithms.* Energy and Buildings, 2021. **239**: p. 110839.
119. Satrio, P., et al., *Optimization of HVAC system energy consumption in a building using artificial neural network and multi-objective genetic algorithm.* Sustainable Energy Technologies and Assessments, 2019. **35**: p. 48-57.
120. Yadav, D., et al., *Optimization of FDM 3D printing process parameters for multi-material using artificial neural network.* Materials Today: Proceedings, 2020. **21**: p. 1583-1591.
121. Fetimi, A., et al., *Optimization and prediction of safranin-O cationic dye removal from aqueous solution by emulsion liquid membrane (ELM) using artificial neural network-particle swarm optimization (ANN-PSO) hybrid model and response surface methodology (RSM).* Journal of Environmental Chemical Engineering, 2021. **9**(5): p. 105837.
122. Fagundez, J., et al., *Joint use of artificial neural networks and particle swarm optimization to determine optimal performance of an ethanol SI engine operating with negative valve overlap strategy.* Energy, 2020. **204**: p. 117892.
123. Khalil, A.J., et al., *Energy efficiency prediction using artificial neural network.* 2019.
124. Zhou, G., et al., *Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings.* Journal of Cleaner Production, 2020. **254**: p. 120082.
125. Unke, O.T. and M. Meuwly, *PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges.* Journal of chemical theory and computation, 2019. **15**(6): p. 3678-3693.
126. Zakaulla, M., F. Parveen, and N. Ahmad, *Artificial neural network based prediction on tribological properties of polycarbonate composites reinforced with graphene and boron carbide particle.* Materials Today: Proceedings, 2020. **26**: p. 296-304.
127. Gadekallu, T.R., et al., *Deep neural networks to predict diabetic retinopathy.* Journal of Ambient Intelligence and Humanized Computing, 2020: p. 1-14.
128. Chen, Y., et al., *A review of the artificial neural network models for water quality prediction.* Applied Sciences, 2020. **10**(17): p. 5776.

129. Asteris, P.G., et al., *Application of artificial neural networks for the prediction of the compressive strength of cement-based mortars.* Computers and Concrete, 2019. **24**(4): p. 329.
130. Yang, B., et al., *Time series analysis and long short-term memory neural network to predict landslide displacement.* Landslides, 2019. **16**: p. 677-694.
131. Cabaneros, S.M., J.K. Calautit, and B.R. Hughes, *A review of artificial neural network models for ambient air pollution prediction.* Environmental Modelling & Software, 2019. **119**: p. 285-304.
132. Lau, E., L. Sun, and Q. Yang, *Modelling, prediction and classification of student academic performance using artificial neural networks.* SN Applied Sciences, 2019. **1**: p. 1-10.
133. Wen, J., et al., *Convolutional neural networks for classification of Alzheimer's disease: Overview and reproducible evaluation.* Medical image analysis, 2020. **63**: p. 101694.
134. Jang, B., I. Kim, and J.W. Kim, *Word2vec convolutional neural networks for classification of news articles and tweets.* PloS one, 2019. **14**(8): p. e0220976.
135. El-Khatib, M.J., B.S. Abu-Nasser, and S.S. Abu-Naser, *Glass classification using artificial neural network.* 2019.
136. Kwon, Y., et al., *Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation.* Computational Statistics & Data Analysis, 2020. **142**: p. 106816.
137. Hemanth, D.J., *EEG signal based modified Kohonen neural networks for classification of human mental emotions.* Journal of Artificial Intelligence and Systems, 2020. **2**(1): p. 1-13.
138. Tammina, S., *Transfer learning using vgg-16 with deep convolutional neural network for classifying images.* International Journal of Scientific and Research Publications (IJSRP), 2019. **9**(10): p. 143-150.
139. Kwon, O., et al., *A deep neural network for classification of melt-pool images in metal additive manufacturing.* Journal of Intelligent Manufacturing, 2020. **31**: p. 375-386.
140. Ma, F., et al., *Automated classification of atrial fibrillation using artificial neural network for wearable devices.* Mathematical Problems in Engineering, 2020. **2020**: p. 1-6.
141. Kujawa, S., J. Mazurkiewicz, and W. Czekała, *Using convolutional neural networks to classify the maturity of compost based on sewage sludge and rapeseed straw.* Journal of Cleaner Production, 2020. **258**: p. 120814.
142. Tang, R.-X., et al., *Evaluating landslide susceptibility based on cluster analysis, probabilistic methods, and artificial neural networks.* Bulletin of Engineering Geology and the Environment, 2020. **79**: p. 2235-2254.
143. Rohini, P., et al. *A study on the adoption of Wireless Communication in Big Data Analytics Using Neural Networks and Deep Learning*. in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2022. IEEE.
144. Kickingereder, P., et al., *Automated quantitative tumour response assessment of MRI in neuro-oncology with artificial neural networks: a multicentre, retrospective study.* The Lancet Oncology, 2019. **20**(5): p. 728-740.
145. Meng, X. and G.E. Karniadakis, *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems.* Journal of Computational Physics, 2020. **401**: p. 109020.

146. Gaby, N., F. Zhang, and X. Ye. *Lyapunov-net: A deep neural network architecture for lyapunov function approximation*. in *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022. IEEE.
147. Hanin, B., *Universal function approximation by deep neural nets with bounded width and relu activations.* Mathematics, 2019. **7**(10): p. 992.
148. Chen, M., et al., *Efficient approximation of deep relu networks for functions on low dimensional manifolds.* Advances in neural information processing systems, 2019. **32**.
149. Adcock, B. and N. Dexter, *The gap between theory and practice in function approximation with deep neural networks.* SIAM Journal on Mathematics of Data Science, 2021. **3**(2): p. 624-655.
150. Ohn, I. and Y. Kim, *Smooth function approximation by deep neural networks with general activation functions.* Entropy, 2019. **21**(7): p. 627.
151. Elfwing, S., E. Uchibe, and K. Doya, *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning.* Neural Networks, 2018. **107**: p. 3-11.
152. Pang, G., L. Yang, and G.E. Karniadakis, *Neural-net-induced Gaussian process regression for function approximation and PDE solution.* Journal of Computational Physics, 2019. **384**: p. 270-288.
153. Haykin, S. and N. Network, *A comprehensive foundation.* Neural networks, 2004. **2**(2004): p. 41.
154. Beale, M.H., M.T. Hagan, and H.B. Demuth, *Neural network toolbox.* User's Guide, MathWorks, 2010. **2**: p. 77-81.
155. Svozil, D., V. Kvasnicka, and J. Pospichal, *Introduction to multi-layer feed-forward neural networks.* Chemometrics and intelligent laboratory systems, 1997. **39**(1): p. 43-62.
156. Sazli, M.H., *A brief review of feed-forward neural networks.* Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 2006. **50**(01).
157. Nielsen, M.A., *Neural networks and deep learning*. Vol. 25. 2015: Determination press San Francisco, CA, USA.
158. Trask, A.W., *Grokking deep learning*. 2019: Simon and Schuster.
159. Chai, Y., L. Jia, and Z. Zhang, *Mamdani model based adaptive neural fuzzy inference system and its application.* International Journal of Computer and Information Engineering, 2009. **3**(3): p. 663-670.
160. Berenji, H.R., *Fuzzy logic controllers.* An introduction to fuzzy logic applications in intelligent systems, 1992: p. 69-96.
161. Walia, N., H. Singh, and A. Sharma, *ANFIS: Adaptive neuro-fuzzy inference system-a survey.* International Journal of Computer Applications, 2015. **123**(13).
162. Jang, J.-S., *ANFIS: adaptive-network-based fuzzy inference system.* IEEE transactions on systems, man, and cybernetics, 1993. **23**(3): p. 665-685.
163. Jang, J.-S. and C.-T. Sun, *Neuro-fuzzy modeling and control.* Proceedings of the IEEE, 1995. **83**(3): p. 378-406.
164. Stearns, S.D., *of Aldapfive Signal Processing.* 1985.
165. Wang, L.-X., *Adaptive fuzzy systems and control: design and stability analysis*. 1994: Prentice-Hall, Inc.
166. Suton, R. and A. Barto, *Reinforcement learning: an introduction. Adaptive computation and machine learning*. 2002, MIT Press, Cambridge (MA), USA.

167. Sutton, R.S. and A.G. Barto, *Reinforcement learning: An introduction*. 2018: MIT press.
168. Daw, N.D. and K. Doya, *The computational neurobiology of learning and reward.* Current Opinion in Neurobiology, 2006. **16**(2): p. 199-204.
169. Johnson, A., M.A.A. van der Meer, and A.D. Redish, *Integrating hippocampus and striatum in decision-making.* Current Opinion in Neurobiology, 2007. **17**(6): p. 692-697.
170. Dayan, P. and Y. Niv, *Reinforcement learning: the good, the bad and the ugly.* Current opinion in neurobiology, 2008. **18**(2): p. 185-196.
171. Johnson, A., M.A. van der Meer, and A.D. Redish, *Integrating hippocampus and striatum in decision-making.* Current opinion in neurobiology, 2007. **17**(6): p. 692-697.
172. O'DOHERTY, J.P., A. Hampton, and H. Kim, *Model-based fMRI and its application to reward learning and decision making.* Annals of the New York Academy of sciences, 2007. **1104**(1): p. 35-53.
173. Doya, K., *Modulators of decision making.* Nature neuroscience, 2008. **11**(4): p. 410-416.
174. Rushworth, M.F. and T.E. Behrens, *Choice, uncertainty and value in prefrontal and cingulate cortex.* Nature neuroscience, 2008. **11**(4): p. 389-397.
175. Kording, K., *Decision theory: what" should" the nervous system do?* Science, 2007. **318**(5850): p. 606-610.
176. Gold, J.I. and M.N. Shadlen, *The neural basis of decision making.* Annu. Rev. Neurosci., 2007. **30**: p. 535-574.
177. Lee, D., *Neural basis of quasi-rational decision making.* Current opinion in neurobiology, 2006. **16**(2): p. 191-198.
178. Niv, Y. and P.R. Montague, *Theoretical and empirical studies of learning*, in *Neuroeconomics*. 2009, Elsevier. p. 331-351.
179. Buşoniu, L., et al., *Reinforcement learning for control: Performance, stability, and deep approximators.* Annual Reviews in Control, 2018. **46**: p. 8-28.
180. Deisenroth, M.P., C.E. Rasmussen, and D. Fox, *Learning to control a low-cost manipulator using data-efficient reinforcement learning.* Robotics: Science and Systems VII, 2011. **7**: p. 57-64.
181. Bai, W., T. Li, and S. Tong, *NN reinforcement learning adaptive control for a class of nonstrict-feedback discrete-time systems.* IEEE Transactions on Cybernetics, 2020. **50**(11): p. 4573-4584.
182. Bøhn, E., et al. *Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization*. in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019. IEEE.
183. Sarmad, M., H.J. Lee, and Y.M. Kim. *Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
184. Raman, N.S., et al. *Reinforcement learning for control of building HVAC systems*. in *2020 American Control Conference (ACC)*. 2020. IEEE.
185. Wang, Z. and T. Hong, *Reinforcement learning for building controls: The opportunities and challenges.* Applied Energy, 2020. **269**: p. 115036.
186. Li, C. and Q. Chang, *Hybrid feedback and reinforcement learning-based control of machine cycle time for a multi-stage production system.* Journal of Manufacturing Systems, 2022. **65**: p. 351-361.

187. Wu, D., et al. *Nested reinforcement learning based control for protective relays in power distribution systems*. in *2019 IEEE 58th conference on decision and control (CDC)*. 2019. IEEE.
188. Procyk, T.J. and E.H. Mamdani, *A linguistic self-organizing process controller.* Automatica, 1979. **15**(1): p. 15-30.
189. Zhang, T., M. Weiszer, and J. Chen, *The feasibility of Follow-the-Greens for 4-dimensional trajectory based airport ground movements.* Transportation Research Part C: Emerging Technologies, 2020. **116**: p. 102632.
190. Wang, J., et al. *Empowering A* search algorithms with neural networks for personalized route recommendation*. in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019.
191. Liu, Y. and M. Hansen, *Predicting aircraft trajectories: a deep generative convolutional recurrent neural networks approach.* arXiv preprint arXiv:1812.11670, 2018.