OPTIMAL PATH PLANNING AND TRAJECTORY OPTIMIZATION FOR MULTIPLE AIRCRAFT LANDING USING RRT ALGORITHM AND PSEUDOSPECTRAL METHODS

By

KRITHIKA MOHAN

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2011

To my parents, Mr. D. Mohan and Mrs. Kusum Mohan; my sister Jyothika; all my friends and family members for motivating me and believing in me

ACKNOWLEDGMENTS

I express my sincere appreciation and gratitude to my supervisory committee chair and mentor, Dr. Anil V. Rao. I thank him for the constant support, encouragement and technical guidance that he provided me throughout my research work. I especially thank him for providing me the license to his software GPOPS which helped me accomplish my research work. I also thank Dr. Warren E. Dixon for lending his knowledge and support during the course of my study at the University of Florida. All that I have learned and accomplished would not have been possible without their support and belief in me. I thank all of my colleagues for helping me with my research work, motivating me towards my goal, and creating a friendly work atmosphere. I also extend my appreciation to Michael Patterson and Christopher Darby for sharing their knowledge and providing me with a better understanding of my research work. Most importantly, I would like to express my deepest appreciation to my parents, D Mohan and Kusum Mohan and my sister, Jyothika. Their love, understanding, motivation, and belief in me made this thesis possible.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

OPTIMAL PATH PLANNING AND TRAJECTORY OPTIMIZATION FOR MULTIPLE AIRCRAFT LANDING USING RRT ALGORITHM AND PSEUDOSPECTRAL METHODS

By

Krithika Mohan

May 2011

Chair: Anil V. Rao
Major: Mechanical Engineering

In this work, a method is presented for solving multiple aircraft landing problem using an RRT algorithm and pseudospectral method. The multiple aircraft landing problem involves generating a conflict-free trajectory for all the aircraft waiting to land on a single runway while avoiding obstacles. The trajectory generated for each aircraft should satisfy time-distance separation constraints between two successive aircraft landings and also the trajectories should not intersect. The multiple aircraft landing problem is solved as a multiple-phase optimal control problem using the open-source optimal control software GPOPS, and a nonlinear programming problem solver SNOPT. In order to improve computational efficiency, a rapidly exploring random tree (RRT) algorithm is used to generate initial guesses for the optimal control software. While the RRT algorithm determines the shortest path between the initial position and the target position, a novel algorithm has been presented which also combines other constraints like maintaining a minimum safe time-distance difference and avoiding intersecting flight paths; thus making it suitable for initializing the multiple aircraft landing problem.

CHAPTER 1
INTRODUCTION

Motivation

Currently, approximately 5000 aircraft land every hour at an airport in the United States. Due to this large increase in density of aircraft arrivals, it has become important to optimize the scheduling of aircraft landings in order to reduce wait time, improve airport efficiency, and minimize fuel consumption while maintaining safety. As the density of aircraft arrivals increases, so does the complexity of trajectory planning and generation.

Presently, the air traffic controller (ATC) uses its radar system to maintain the position of each of the aircraft arriving and waiting to land at the airport. The ATC provides information and support to the pilots to prevent collisions with other aircraft. The ATC makes use of the trajectory provided by the Instrument Landing System (ILS) which uses a combination of radio signals and high intensity light arrays to guide the aircraft for safe and precision landing. All these systems generate the trajectory using the present and predicted future position of the aircraft. Although these modern equipments are being used by the ATC for assistance, human error is always a possibility. In this work, aircraft landing trajectories are generated by taking into consideration various constraints such as the time difference between each aircraft landing, obstacle collision avoidance, dynamics, and minimum time for approach.

Literature Review

Conflict Detection and Resolution

A conflict is defined as the situation of loss of minimum safe separation between two aircraft [1]. The ATC is responsible for conflict detection and its resolution. It also

tries to fulfill the desired arrival and departure time of the aircraft. Conflict detection is the process of evaluating the probability of a conflict that will occur in the near future, based on the current position and flight path of an aircraft. Conflict resolution is the process of finding the maneuver required by one or multiple aircraft to avoid the predicted conflict. A great deal of research has been done to improve the performance of the air traffic controller. A review of conflict detection and resolution modeling techniques has been discussed by Kuchar, J.K. and Yang, L.C. [2].

The conflict detection and resolution process consists of predicting the conflict, communicating it to the pilot, and helping resolve the conflict. This process is shown in Figure 1-1. The surveillance is done by radars, GPS and other sensors, based on which the current state of each aircraft is estimated. The dynamic trajectory model and the flight plan are used to determine the future state. The current and future state is compared with the metrics for conflict detection and resolution. This information is given to the air traffic controller (human operator), who communicates with the pilot and helps resolve the conflict.

A number of conflict resolution algorithms that uses genetic algorithm, semi-definite programming, mixed integer programming (MIP) and sequential quadratic programming (SQP) methods have been proposed in literature [3-6].

A real-time aircraft conflict resolution approach that uses genetic algorithm is proposed by Durand, N. and others [6]. The authors solve a real variable combinatorial problem which combines both local and global optimization methods. The genetic algorithm is run initially and is followed by a local optimization hill-climbing algorithm to improve the best solution obtained for each chromosome.

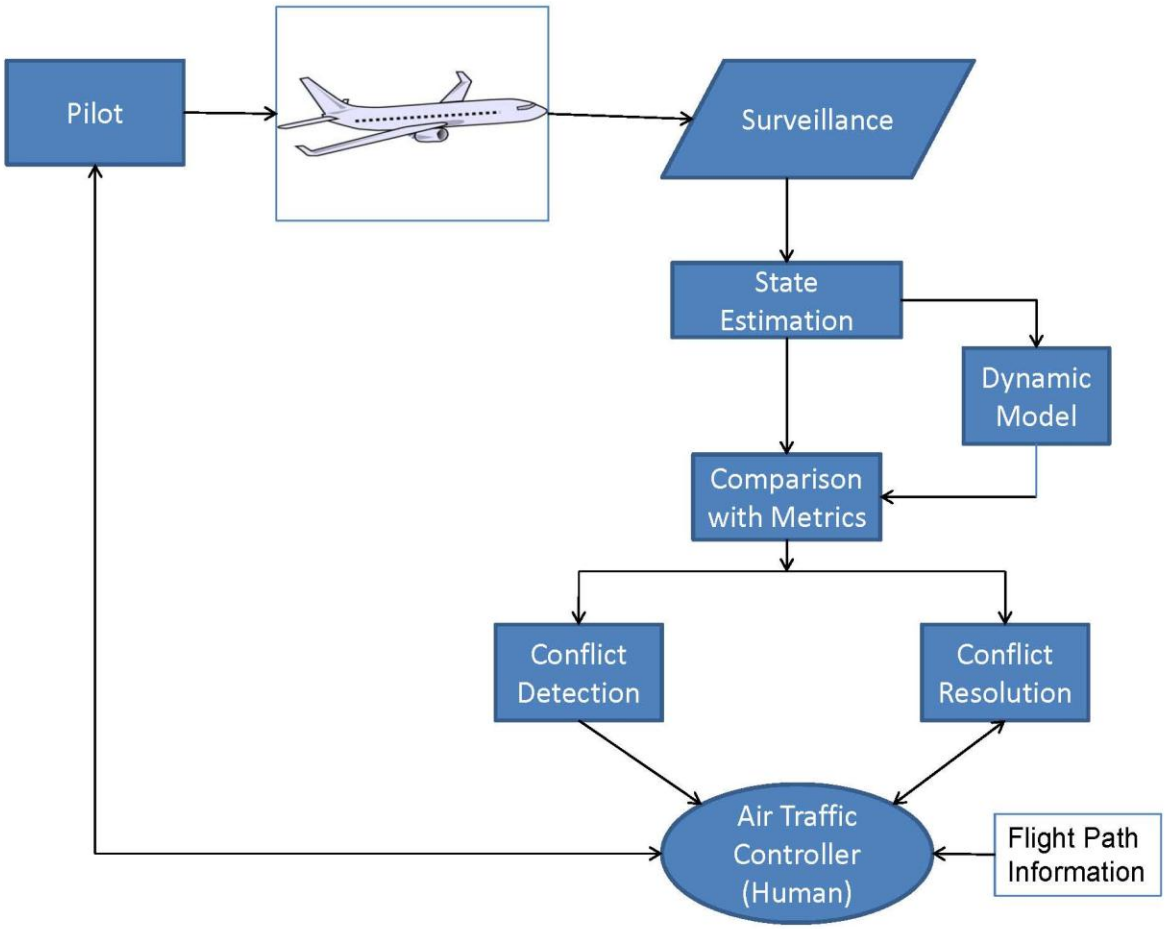

Figure 1-1. Conflict detection and resolution process

The conflict resolution problem is formulated as a non-convex quadratically constrained quadratic problem by Frazzoli, E. and others [3], which is approximated by a convex semi-definite program and solved. The optimal solution to this program is used to generate feasible and locally optimal maneuvers. The authors follow the resolution methodology where they try to minimize the deviation between the desired heading proposed by the aircraft and a conflict-free heading determined by the air traffic controller.

A multiple aircraft conflict avoidance problem is also discussed by Pallottino, L. and others [4], where a mixed-integer linear programming is used to formulate the problem with only velocity or heading changes as admissible maneuvers. The linear

formulation of the two problems is solved using the optimization software CPLEX to find the optimal solution to the mixed integer program (MIP).

A cost function which favors the lower priority aircraft for resolving the predicted conflicts by involving changes in heading, speed, and altitude was introduced by Hu, J. and others [5]. The vertical maneuvers are penalized for passenger comfort. The authors developed a constrained optimization problem for a two aircraft scenario, which is extended to multiple aircraft by approximating the optimization problem with a finite-dimensional convex optimization problem with linear waypoint constraints. Additional constraints on waypoint position were added to avoid sharp turns near the waypoints.

A three-dimensional trajectory optimization algorithm is developed by Menon, P.K. and others [7] to obtain a conflict-free flight path. The algorithm uses nonlinear point-mass model and includes realistic operational constraints on individual aircraft; making the resulting problem non-convex. Also the problem of optimal cooperative three-dimensional conflict resolution involving multiple aircraft is addressed by Raghunathan, A.U. and others [8]. Here the authors specify the initial and final locations along with the detailed point-mass aircraft dynamics. The infinite-dimensional optimal control problem is converted into a finite dimensional nonlinear program (NLP) by use of collocation on finite elements. This NLP is solved in IPOPT using an integer point algorithm which employs a line search method. The key concept in this paper is that it used the dynamic optimization by using the detailed dynamic model of the aircraft instead of using the kinematic model; thus generating feasible trajectories that are flyable. The Hu et al. procedure is used to solve the optimal control problem and obtain the trajectory of individual aircraft. This trajectory is used for initialization (initial guess) of the NLP.

In this work, a detailed point-mass dynamic model of the aircraft is used for generating flyable trajectories [9]. The infinite-dimensional optimal control problem here is converted into a finite-dimensional nonlinear program (NLP) by using the open-source optimal control software, GPOPS and solved using the NLP solver, SNOPT. An additional capability of the proposed method is that every aircraft can specify its individual dynamic model. This means that the trajectory can be generated simultaneously for different types of aircraft (including helicopters) that may occupy the airspace. Also for initialization of the NLP, a rapidly exploring random tree (RRT) algorithm is employed which finds the shortest path between two points avoiding any obstacles in the path. The shortest possible time required by each of the aircraft to reach the goal point maintaining the time difference constraint between successive aircraft landings (without considering the aircraft dynamics) is used to specify the final times in the optimal control software. The RRT algorithm is explained in detail in Chapter 3. As the speed of the aircraft is maintained constant in this problem, the only allowable change is the bank angle, heading angle and flight path angle of the aircraft by using the vertical and horizontal load factors as the control.

Trajectory Path Planning Algorithms

The main objective of a path planning algorithm is to find a path which satisfies certain conditions while avoiding obstacles in the path and preventing collisions with other moving objects. Such trajectory or motion planning algorithms have been primarily used in robotics, and dynamics and control. In dynamics and control, the guidance of vehicles depends on the dynamic behavior of the vehicle. Both applications need motion planning algorithms to find path to reach a goal in a partially known environment with obstacle avoidance and also collision avoidance with other vehicles. However, for

aerial vehicles, additional parameters needs to be considered such as the three-dimensional environment, non-trivial dynamics, safety due to disturbed operating conditions, and high level of uncertainty in the state knowledge due to sensor and radar measurements. An overview of all the existing motion planning algorithms used in unmanned aerial vehicle, (UAV) guidance is presented by Goerzen, C. and others [10].

A comprehensive study and comparison of various path planning methods currently in use was provided by Hurni, M.A. [11]. The author compared and graded algorithms like bug algorithm, potential fields, roadmaps, cell decomposition and optimal control based on its ability for path planning and control, optimality, obstacle avoidance, handling vehicle constraints, global vs. local information, computational complexity, feasibility, completeness, multiple vehicles, error and uncertainties. Based on his inference, the optimal control method was found to be the best in all the criteria except computational complexity. The research [11-13] also discussed the obstacle modeling and avoidance techniques for use in ground vehicle maneuver.

Some of the basic shortest path finding algorithms are given below:

1. Dijkstra's algorithm used to solve the single-source and single-destination shortest path problems with non-negative edge weights.
2. Bellman-Ford algorithm used to solve the single source problem with negative or non-negative edge weights.
3. A* search algorithm used to find the least-cost path using heuristics to increase the search speed.
4. RRT algorithm used to find shortest path with obstacle avoidance. It biases the search to the least explored regions, for effective and faster result.
5. Perturbation theory used to find the locally shortest path in worst case.

Dijkstra's algorithm

Dijkstra's algorithm is a single-source shortest-path planning algorithm which has non-negative weights in its edges. It is also known as a greedy algorithm. The start point is at a source vertex, S, from which a tree, T grows such that it ultimately spans all vertices reachable from S. The vertices that are closest to the tree are added first, i.e., first the source S is added, then the vertex closest to S, then the next closest, and so on.

The worst-case running time for the Dijkstra algorithm on a graph with n vertices and m edges is $O(n^2)$ because it allows for directed cycles. It even finds the shortest path from a source node, S to all other nodes in the graph.

Bellman-Ford algorithm

The Bellman-Ford algorithm is also a single-source shortest-path planning algorithm which can have negative weights in its edges. It does not follow the greedy approach used in Dijkstra's algorithm. However it initializes the distance of the source vertex to 0 and all other vertices to infinity, ∞. If V is the number of vertices, then it does |*V*| -1 passes over all edges relaxing, or updating, the distance to the destination of each edge. Thus it checks all the edges again for any negative weigh cycles, in which case it returns false. No shortest path exists during the negative cycle. The time complexity is O(|*V*|·|*E*|) time, where |*V*| and |*E*| are the number of vertices and edges respectively.

A* search algorithm

A* uses a best-first search algorithm to find the least-cost path from a given initial node to a goal node. It uses a heuristic function, $f(x)$ to determine the order of the

search. This heuristic function is a sum of two functions, one for minimizing the distance and other for minimizing the cost. It can be written as,

$$f(x) = g(x) + h(x) \quad (1\text{-}1)$$

where $g(x)$ is the path cost function, which denotes the cost to be minimized from the starting to the goal node; and $h(x)$ is estimate of the shortest distance from the starting to the goal node, which is usually denoted by a straight line distance to the goal.

A* can also be used as a bidirectional heuristic search algorithm like the RRT algorithm, which is explained in Chapter 3. The time complexity of A* will depend on the heuristic. In the worst case, the number of nodes added to the tree is exponential in the length of the shortest path solution. It becomes a polynomial if the search space is a tree with a single goal state, and if the heuristic function *h* meets the following condition,

$$|h(x) - h^*(x)| = O(\log h^*(x))$$

where h^* is the optimal distance cost to get from node *x* to the goal node. That is, the error of *h* will not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true optimal distance from node *x* to the goal node.

RRT algorithm

A new technique of path finding called rapidly random exploring trees (RRT) was proposed by Lavalle, S.M. [14]. The main advantage of this algorithm over other algorithm is that it can be used for solving problems which involves obstacles as path constraints and also kinodynamic or nonholonomic differential constraints of the vehicle. It can be used to generate trajectories for nonlinear systems with state and path constraints. It also biases the search to the least explored regions, for effective and faster result. The different types of RRT algorithm is explained in detail in Chapter 3.

The RRT algorithm is being used for various applications in recent years. It is used as a path planner for real time obstacle detection and avoidance in autonomous small-scale Helicopters [15]. Also Caves, A.D.J [16] used the RRT algorithm for human supervisory control of UAV missions in environments with dynamic obstacles fields of different densities. The algorithm is used to assist the human operator in path planning and re-planning for a group of UAVs. The RRT algorithm is also used by Aoudé, G.S. [17], for designing optimal reconfiguration maneuvers for multiple spacecrafts. The author proposed a two stage approach, where he used RRT algorithm in the first stage of the initializing the optimal control problem without the differential constraints and get a feasible initial guess.

The RRT algorithm is used in this work for initializing the multiple aircraft landing optimal control problem, due to its ability to find faster solutions especially in an environment with obstacles.

Aircraft Trajectory Optimization

The optimal flight trajectory generation in the presence of windshear is discussed by Miele, A. and others [18, 19]. The numerical solution of this optimal control problem of the Bolza type is solved using the dual sequential gradient-restoration algorithm (DSGRA). The author optimizes the take-off trajectories by minimizing the peak deviation of absolute path inclination from a reference value. Also the abort trajectories are optimized by minimizing the peak drop of altitude from a reference value. The survival capability of the optimal abort landing trajectory in a severe windshear was found superior to that of the other comparison trajectories like the constant pitch guidance trajectory and the maximum angle of attack guidance trajectory.

An automatic aircraft landing control algorithm is also developed using both fuzzy logic and neural networks models [20-22]. While the authors, Nho, K. and Agarwal, R.K. [20] used the linear and nonlinear aircraft model, the authors, Juang, J. and Chio, J. [21] used a linearized aircraft model for implementing fuzzy logic technique. An artificial neural network is used by Chaturvedi, D.K. and others [22] to deal with the nonlinearities in the aircraft model. Also Ruffier, F., and Franceschini, N. [23] developed a visual guidance based autopilot which enables the micro air vehicle to take-off, cruise, and land, while reacting adequately to wind disturbances.

An autonomous counter-hijack control of aircraft is proposed by Patel, R.B., and Goulart, P.J. [24], where the critical buildings are modeled as obstacles to be avoided in the aircraft path. These building are modeled by employing dualization of state exclusion regions to maintain continuity. The warm start method which uses prior solutions as initial guess is used for initialization of a direct multiple shooting method to solve the optimal control problem.

In recent years, pseudospectral methods, a type of direct trajectory optimization, has been used extensively in aerospace applications as an effective and faster way of solving optimal control problems. A pseudospectral method for real time motion-planning and obstacle avoidance is discussed by Lewis, L.R. and others [12, 13], for generating minimum-time trajectories for unmanned ground vehicles. The gauss pseudospectral method (GPM) is also used by Aoudé, G.S. [17] to solve the optimal control problem of reconfiguration maneuvers for multiple spacecrafts.

A Gauss pseudospectral method using orthogonal collocation which uses the LG points for collocation is developed by Benson, D.A. and others [25]. The orthogonal

collocation of the dynamics is performed only at these collocation points, and not the boundary points. It also properly approximates the costates, which leads to a set of KKT conditions that is identical to the discretized form of first-order optimality conditions at the collocation points. This helps in accurate costate estimation using the KKT multipliers of the NLP.

A method for direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using global collocation at Legendre-Gauss-Radau (LGR) points is presented by Garg, D. and others [26]. It is shown that the dual multipliers for the discrete scheme correspond to a pseudospectral approximation of the adjoint equation using polynomials, one degree smaller than that used for the state equation. Also it is shown that the inverse of the pseudospectral LGR differentiation matrix is precisely the matrix associated with an implicit LGR integration scheme. Hence, the method is a global implicit integration method or a pseudospectral method.

Consequently, a unified framework was designed for numerically solving optimal control problems using collocation at Legendre-Gauss(LG), Legendre-Gauss-Radau (LGR), and Legendre-Gauss-Lobatto (LGL) points [27]. It was shown that the LG and LGR differentiation matrices are rectangular and full rank whereas the LGL differentiation matrix is square and singular. Hence, the differential and integral forms are equivalent in the LG and LGR schemes while they are not equivalent in the LGL scheme. The LG and LGR discrete costate systems were found to be full rank while the LGL discrete costate system is rank-deficient. Also, the LGL costate approximation is

found to have an error that oscillates about the true solution due to the null space in the LGL discrete costate system.

An *hp*-adaptive pseudospectral method with collocation at Radau points is chosen in this work [28]. This method iteratively and simultaneously determines the number of segment breaks, the width of each segment, and the polynomial degree required in each segment for approximation, until the user-specified accuracy is achieved. It leads to higher accuracy solutions with less computational effort and memory than is required in global pseudospectral methods. As the problem in this work, has multiple aircraft with an obstacle wall in the path to be avoided, the problem is broken into several overlapping phases, for each aircraft to cross the obstacle at the desired location and then reach the goal. Due to the complexity of the problem, the *hp*-adaptive method is chosen.

About the Thesis

In this research work, an RRT algorithm is used for simultaneous path planning of multiple aircraft to avoid any collision with each other, as the aircraft reaches the goal one by one. While the RRT algorithm determines the shortest path between the initial position and the target position, a novel algorithm has been presented which also combines various other constraints like maintaining a minimum safe distance and time difference between each of the aircraft landings and also avoiding any intersecting flight paths. The number of aircraft and their different start positions is to be provided initially to the algorithm, which then generates a trajectory for these aircraft using the RRT algorithm to avoid any static obstacles in the path. These static obstacles can be buildings or other ground or air vehicles or a no-fly zone.

The final time taken by each aircraft to reach the goal and the path genearated is then used in initializing the multiple- phase optimal control problem that is converted into a nonlinear program (NLP) using GPOPS and solved using SNOPT. An *hp*-adaptive pseudospectral method that uses Radau collocation points is implemented to simultaneously generate the trajectory of three aircraft with overlapping phases. These trajectories satisfy the dynamics of the aircraft.

The research can be used to not only assist the ATC in planning the aircraft trajectory and avoiding conflicts with other aircraft, but also aims at avoiding obstacles (three-dimensional or two-dimensional) in its path. These obstacles can be important buildings or a no-fly zone. Hence it also helps in securing these areas and buildings. With future work on this research, real time implementation may also be possible.

Thesis Outline

This thesis is organized as follows. Chapter 1 is an introduction on the motivation for the research on multiple aircraft landing trajectory generation and a review of the previous work related to the research. It also describes the conflict detection and resolution problems, along with various path planning algorithms and trajectory optimization methods. Also the choice of using the RRT algorithm for initial path planning and pseudospectral method for optimization, along with a brief description of the research work has been presented. In Chapter 2, the multiple aircraft landing problem is formulated along with a general description of optimal control problems with its optimality and transversality conditions. Also the method for obstacle formulation in three-dimension is described. Chapter 3 discusses in detail the three different types of rapidly random exploring (RRT) algorithm. The advantages and differences of one over the other are discussed. The choice of using multiple RRT over the others is also

presented here. In Chapter 4, a novel algorithm is developed that uses multiple RRT algorithm for generating trajectories for multiple aircraft landing problem. Also examples of trajectories generated using three aircraft are shown. Chapter 5 gives a brief description of the Radau psedospectral and *hp*-adaptive methods initially and then describes the formulation of the optimal control problem using overlapping phases. It also gives examples for trajectory generated simultaneously for three aircraft using six phases.

CHAPTER 2
MULTIPLE AIRCRAFT LANDING CONTROL PROBLEM

This chapter gives a brief description of an optimal control problem and methods of solving this problem. The multiple aircraft landing problem is described along with the equations of motion of the aircraft and the formulated cost function. The three-dimensional obstacle formulation is also described in this chapter.

Structure of an Optimal Control Problem

Table 2-1. Notation

t_0	Initial Time
t_f	Final Time
$\mathrm{x}(\mathrm{t}_0)$	State value at the initial time, t_o
$\mathrm{x}(\mathrm{t}_\mathrm{f})$	State value at the final time, t_f
J	Cost Function
J_a	Augmented Cost Function
Q	Mayer Cost
L	Lagrange Cost
ϕ	Boundary Condition
g	Path Constraint
H	Hamiltonian
ϑ	Lagrange Multiplier for Boundary Condition
λ	Costate of the Differential Equation
V	Fixed Speed of the Aircraft, Nm/s
g	Acceleration due to Gravity, Nm/s^2
t_{obs}	Time taken by Aircraft to Reach the Obstacle Wall, sec

An optimal control problem is formulated with the dynamic equations of motion, the objective function, boundary conditions, and path constraints.

The dynamic equations should be continuous and differentiable. It is converted into the state space representation to get the individual state equations and its relation

with the control input. Hence the number of equations is equal to the number of states in the problem. These dynamic equations are represented as given in Equation (2-1),

$$\dot{x} = f(x, u, t) \tag{2-1}$$

where $\mathrm{x(t)} \in \mathbb{R}^n$ is the state, $\mathrm{u(t)} \in \mathbb{R}^p$ is the control, and t is the time.

The objective function also called as cost function, is the parameter that has to be optimized (minimized or maximized) while solving the problem. The cost function is a function of state variables, control variables, and/or time. That is it can be the end point states, end point time or the state and/or control over the entire period of time. The end point cost is known as Mayer cost and the cost function consisting of state or control variables, for the entire period of time is known as Lagrange cost.

$$J = Q\big(x(t_0), t_0, x(t_f), t_f\big) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt \tag{2-2}$$

In the above cost function, J, $Q\big(x(t_0), t_0, x(t_f), t_f\big)$ is the Mayer cost and $L(x(t), u(t), t)$ is the Lagrange cost.

Boundary conditions are the initial and terminal values of states and time, known at the beginning of the problem or to be achieved at the end. They become the initial and terminal constraints of the problem.

$$\phi\big(x(t_0), t_0, x(t_f), t_f\big) = 0 \tag{2-3}$$

The path constraints are the linear or nonlinear constraints to be satisfied by the system in its path. They can be equality constraint or an inequality constraint.

The nonlinear inequality constraint is represented as

$$g(x(t), u(t), t) \leq 0 \tag{2-4}$$

The nonlinear equality constraint is represented as

$$g(x(t), u(t), t) = 0 \tag{2-5}$$

First Order Optimality Conditions

The augmented cost function is formulated as,

$$J_a = Q - \vartheta^T \phi + \int_{t_0}^{t_f} [L + \lambda^T (f - \dot{x})] dt$$

The Hamiltonian is represented as

$$H = L + \lambda^T f = H(x, \lambda, u)$$

The augmented function in terms of Hamiltonian is given by,

$$J_a = Q - \vartheta^T \phi + \int_{t_0}^{t_f} [H - \lambda^T \dot{x}] dt$$

The optimality condition for costate dynamics is given as,

$$\dot{x} = \left[\frac{\partial H}{\partial \lambda}\right]^T$$

The optimal control, u^* is found using the condition given as,

$$\left[\frac{\partial H}{\partial U}\right]^T = 0$$

Transversality Conditions

The first variation of the augmented cost function J_a is used to find the transversality conditions. These conditions can be used to solve the differential equations which are found from the first order optimality conditions. There are different transversality equations for different boundary conditions on state and time of the system, which are given below:

- For no boundary conditions on state and time of the system, $\delta\vartheta = 0$, the boundary condition is $\phi\big(x(t_0), t_0, x(t_f), t_f\big) = 0$
- For fixed initial state, $\delta x_0 = 0$, the costate at initial time is given by,

$$\lambda(t_0) = -\left[\frac{\partial Q}{\partial x(t_0)}\right]^T + \left[\frac{\partial \phi}{\partial x(t_0)}\right]^T \vartheta$$

- For fixed final state, $\delta x_f = 0$, the costate at final time is given by,

$$\lambda(t_f) = \left[\frac{\partial Q}{\partial x(t_f)}\right]^T - \left[\frac{\partial \phi}{\partial x(t_f)}\right]^T \vartheta$$

- For fixed initial time, $\delta t_0 = 0$, the Hamiltonian at initial time is given by,

$$H(t_0) = \frac{\partial Q}{\partial t_0} - \vartheta^T \frac{\partial \phi}{\partial t_0}$$

- For fixed final time, $\delta t_f = 0$, the Hamiltonian at the final time is given by,

$$H(t_f) = -\frac{\partial Q}{\partial t_f} + \vartheta^T \frac{\partial \phi}{\partial t_f}$$

Methods of Solving Optimal Control Problem

An optimal control problem is one where differential equations need to be solved subject to constraints while optimizing a performance index. The two main classifications of methods to solve such optimal control problems with path constraints and boundary conditions are indirect methods and direct methods.

Indirect Method

This method emanates from variational calculus. It requires solving of a two-point boundary value problem. Here the optimality conditions for solving of optimal trajectory are to be derived. The first order optimality conditions are formed using variation of calculus, by taking the first variation of the cost functional. This gives the first-order differential equations for the states and the costates. The transversality conditions usually give nonlinear equations which cannot be solved with Ricatti equation. Thus they form the nonlinear constraints in the problem.

Indirect method converts the optimal control problem into a purely differential-algebraic system of states, costates and dynamics. Hence it becomes a root finding problem, where a set of differential equations has to be solved and roots are found to

satisfy the constraints. The different types of indirect method are Indirect shooting and Indirect multiple shooting. In Indirect multiple shooting method, the time interval is divided into several smaller intervals with indirect shooting implemented in each interval.

Direct Method

In direct method, all the functions (states and control) in the optimal control problem are approximated and transcribed into a nonlinear optimization problem. By making this approximation, the continuous time infinite-dimensional problem is converted into a nonlinear programming problem, which is a finite-dimensional problem. The quadrature approximation of the integral is used in the continuous cost function as given below,

$$J = Q + \int_{t_0}^{t_f} Ldt \cong Quadrature\ Approximation\ of\ Integral$$

$$J \approx J_a$$

The path constraints are evaluated only at specific points of the quadrature. Pseudospectral method is a direct method of solving an optimal control problem, which is explained in detail in Chapter 5.

Problem Formulation

In this work, three aircraft are taken into consideration to generate the landing trajectories. Initially, the position of all the three aircraft is recorded, assuming this time to be the start time, t_0 of the problem. The objective is to make all these aircraft land (reach a common goal point) on a single runway, such that there is a safe time difference between each aircraft landings. The aircraft that is nearest to the runway should land first, then the next nearest aircraft, and so on. In the aircraft dynamics, the speed of all the aircraft is considered to be constant. Hence, only the trajectory of these

aircraft (heading angle, flight path angle, and bank angle) can be manipulated to achieve the objective. Also the path has obstacles, which is formulated like a wall of some thickness with a small opening, through which the aircraft has to pass to reach the runway. This enforces additional constraints, like obstacle avoidance and also avoiding collision with each other.

A figurative illustration of the problem with three aircraft and the obstacle wall is shown in Figure 2-1. It shows that, the trajectory of the farthest aircraft has to take a circuitous path to maintain the time difference between successive aircraft landings, prevent collision with other aircraft, and also should avoid the obstacle wall.

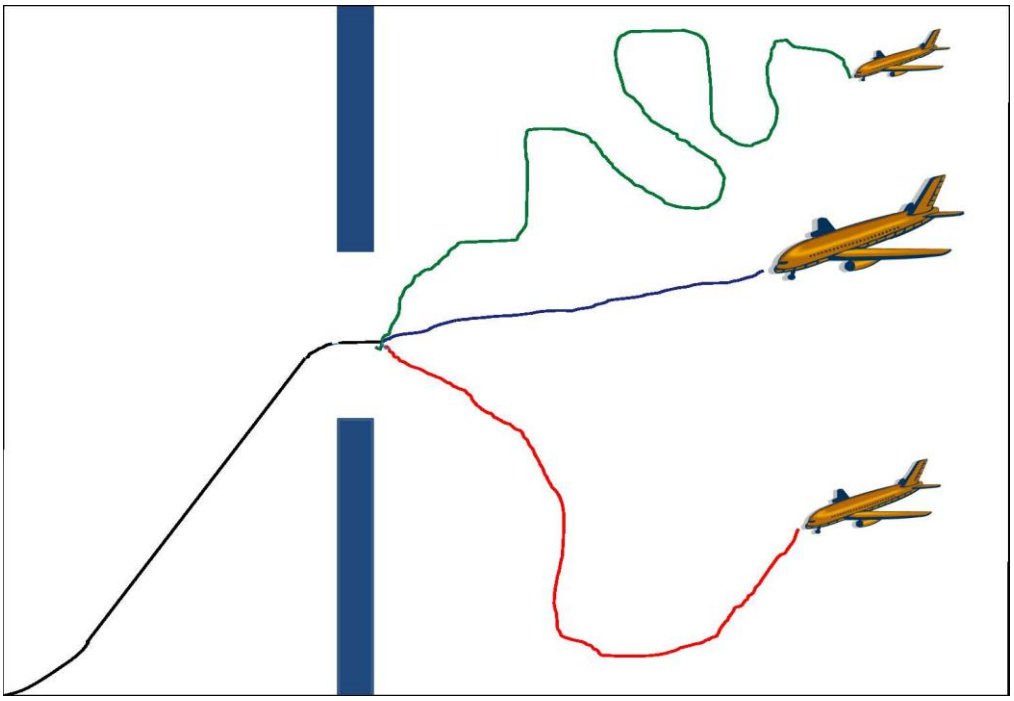

Figure 2-1. Figurative illustration of the multiple aircraft landing problem

The aircraft dynamics in three-dimension is considered in this work. Though all the three aircraft are considered to have the same 3D dynamics in this problem, the way the pseudospectral implementation is formulated, each aircraft can have its own dynamic equations, which is explained in Chapter 5.

Aircraft Dynamics

States

There are five states considered for each aircraft. They are –

1. x - x position
2. y - y position
3. z - z position
4. γ - Flight path angle
5. φ - Heading angle

Here the first three states are the x, y, and z position of the aircraft in three-dimension. The fourth state is the flight path angle, which is defined as the angle between the aircraft's velocity vector and the horizontal reference plane vector. The fifth state is the heading angle, which is the yaw angle (rotation about the z-axis, which is the upward/downward facing axis). It decides the left or right direction of the aircraft.

Control

There are two control variables in this problem. They are –

1. n_h - Horizontal load factor
2. n_v - Vertical load factor

The total load factor is, $n = \sqrt{n_h^2 + n_v^2}$

This total load factor, n determines the ratio of lift to weight.

$$n = \frac{Lift}{Weight}$$

The control is also related to the bank angle, μ with the relation,

$$\mu = \tan^{-1} \frac{n_h}{n_v} \tag{2-6}$$

The bank angle is the angle between the horizontal reference plane vector towards the left and the right wing of the aircraft in the lateral plane, which is positive

(clockwise) when the right wing is facing down and negative (anti-clockwise) when it is facing up, with the horizontal plane vector towards the right.

State-space dynamic equations

The state-space representation for each of the state is given below. The acceleration due to gravity, g is taken as 0.0052952 Nm/s^2 and the fixed speed value, V is taken as 0.04 Nm/s which is about 1.5 times the stall speed of a generic medium range aircraft. This speed is near landing speed of an aircraft.

$$\dot{x} = Vcos\,\gamma\cos\varphi$$

$$\dot{y} = Vcos\,\gamma\sin\varphi$$

$$\dot{z} = V\sin\,\gamma$$

$$\dot{\gamma} = \frac{g}{V}\,(n_v - \cos\,\gamma)$$

$$\dot{\varphi} = \frac{g}{V}\left(\frac{n_h}{\cos\,\gamma}\right)$$

Cost Function

The cost function is formulated and explained in Chapter 5. The Mayer cost function is the time for the aircraft to reach the obstacle wall, which is being maximized in this problem. The Lagrange cost in this problem is minimizing the change in flight path angle and heading angle of the aircraft. The formulated cost is given as,

$$J = \sum_{i=1}^{N} -t_{obs}^{ii} + \sum_{j=1}^{2N}\sum_{k=1}^{n}\int_{t_o}^{t_f}\left(\frac{d\gamma^{jk}}{dt}\right)^2 + \left(\frac{d\varphi^{jk}}{dt}\right)^2 dt$$

where t_{obs} is the time taken by the aircraft to reach the face of the obstacle, N is the total number of aircraft taken into consideration, n is the number of aircraft present in

Phase j, and 2N is the total number of phases formulated in the problem as seen in Chapter 5.

Obstacle Formulation

The obstacles form the path constraints of the problem. All the functions in an optimal control problem should be continuous and differentiable. Hence the obstacles should be represented by smooth function. It can be approximated by geometric shapes. In literature most of the obstacles are formulated as a circle in two-dimension and as a sphere in three-dimension. But Hurni, M.A. [11] represented polygonal obstacles in two-dimension by using the p-norm method.

Two–Dimensional Obstacle Modeling

The two-dimensional obstacles can be modeled using the p-norm method [11-13, 29]. Using the p-norm method, obstacles can be modeled as circle, ellipse, square, rectangle of different types. These shapes can be used to fit most of the obstacles. It is very complex to model the obstacle to the exact shape and size.

The general equation used to model these shapes is given below,

$$q\big(x(t), y(t)\big) = \left|\left(\frac{x(t) - x_c}{a}\right)^p\right| + \left|\left(\frac{y(t) - y_c}{b}\right)^p\right| - |c^p| = 0$$

The variables x_c and y_c are the center point of the obstacle. The value of a, b, c indicate the size of the obstacle, where ‘a’ is the distance along the x-axis, ‘b’ is the distance along the y-axis from the center of the obstacle and ‘c’ is the radius of the obstacle. The Figure 2-2 shows the various shapes generated by using different values for p, a, and b in the general equation given above, with c=1.

For example to represent a rectangle with center at (2, 2) and length and breadth as 8 and 4 respectively, the equation can be written as,

$$q\big(x(t),y(t)\big) = \left|\left(\frac{x(t)-2}{4}\right)^{50}\right| + \left|\left(\frac{y(t)-2}{2}\right)^{50}\right| - |1^{50}| = 0$$

$x_c = 2, y_c = 2, a = 4, b = 2, p = 50$. Assume c to be 1 here.

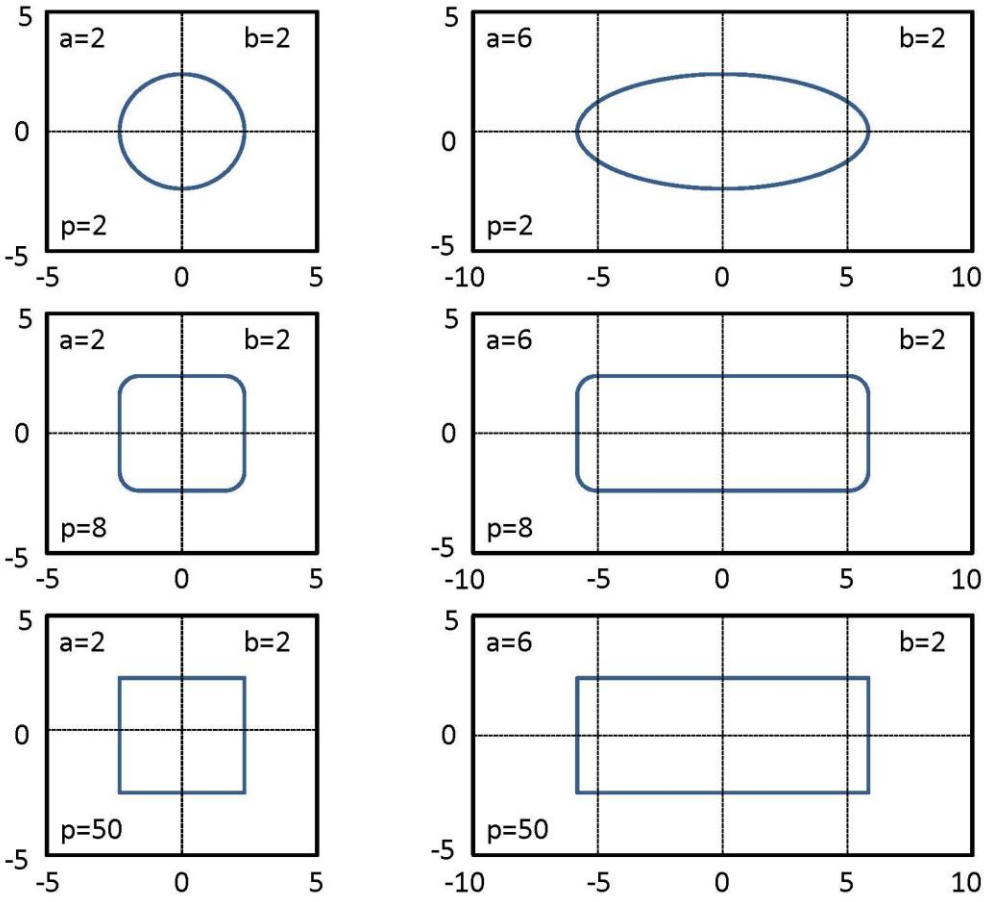

Figure 2-2. Shapes generated for different values of p, a, and b

The ‘p’ value is always taken as even number, to get the absolute non negative value of the power. Also as the power of 50 can become a large constraint to handle, the natural logarithm is taken on both sides of the example equation,

$$q\big(x(t),y(t)\big) = ln\left(\left(\frac{x(t)-2}{4}\right)^{50} + \left(\frac{y(t)-2}{2}\right)^{50}\right) \geq 0$$

Three–Dimensional Obstacle Modeling

In this work, the problem is defined in a three-dimensional environment; hence the obstacle should also be defined in three-dimension. Thus the p-norm method is used to represent the obstacle in three-dimensions with x, y, and z axis. The general equation for 3D obstacle representation is given below,

$$q\big(x(t), y(t)\big) = \left|\left(\frac{x(t) - x_c}{a}\right)^p\right| + \left|\left(\frac{y(t) - y_c}{b}\right)^p\right| + \left|\left(\frac{z(t) - z_c}{c}\right)^p\right| - |d^p| = 0$$

The center point of the obstacle is now represented as (x_c, y_c, z_c). The distance along the x, y, and z axis from the center is given as a, b, c. The radius of the obstacle now is given by d.

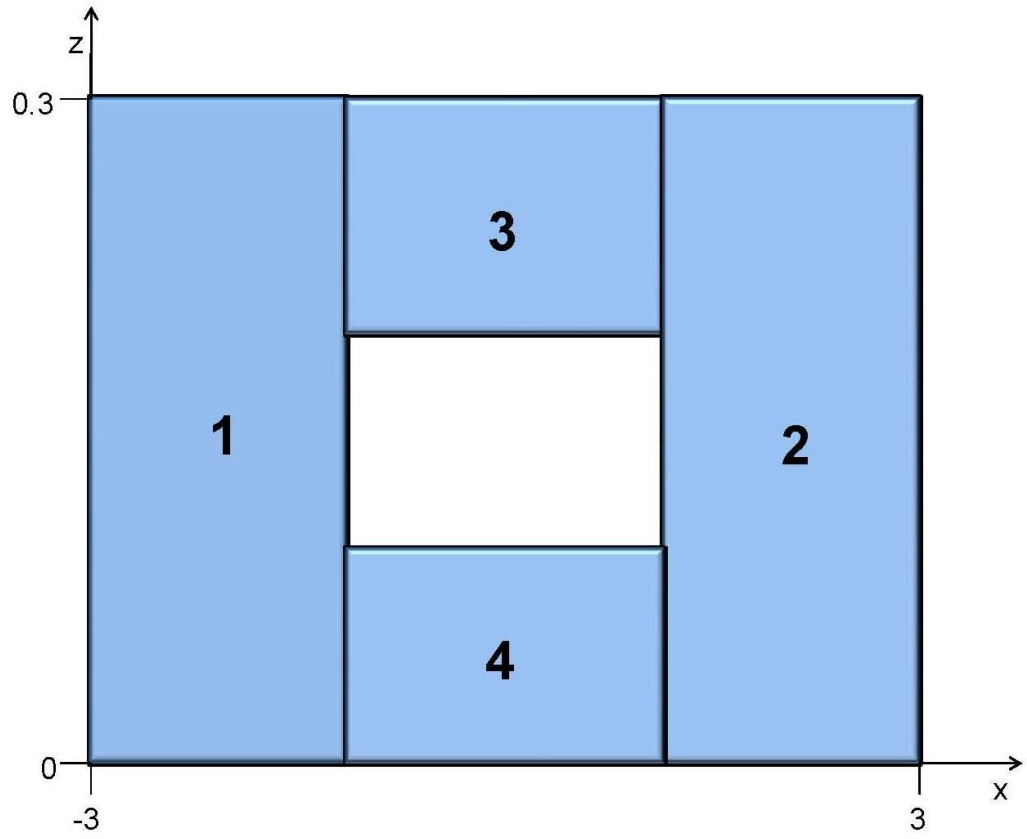

Figure 2-3. Obstacle wall representation with center flyable hole.

This method is used to model an obstacle, which is of the form of a wall with a square hole in the center. The wall has a small thickness, which can be modeled in three-dimension. The wall with hole is constructed by taking four obstacles, such that a square flyable hole is formed by them. This is shown in the Figure 2-3.

The four obstacle walls are numbered as 1, 2, 3 and 4 in the Figure 2-3. Each of the walls is taken as an individual obstacle to be modeled. They are modeled as a function using the general equation, to make it smooth and differentiable. These functions form the nonlinear path constraints in the problem. The limits on x, y, and z axis of the environment considered are taken as, -3 to 3, 0 to 6 and 0 to 0.3 respectively.

Table 2-3. Three-dimensional obstacle parameters

Obstacle	Center	p	Distance along x-axis	Distance along y-axis	Distance along z-axis
1	(-2, 3.0005, 0.15)	50	1	0.0005	0.15
2	(2, 3.0005, 0.15)	50	1	0.0005	0.15
3	(0, 3.0005, 0.25)	50	1	0.0005	0.25
4	(0, 3.0005, 0.05)	50	1	0.0005	0.05

Table 2-2 displays the parameters of the obstacles used in the problem. The center, p value and the distance along each axis (a, b, and c value) of all the obstacles are displayed. All the distance is taken in nautical miles.

The user-specified parameters of the obstacles are then modeled into functions using the general equation for three-dimensional obstacles. The obstacles of the example problem solved in this work is modeled as below-

$$obs1 = \left(\frac{x(t)+2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50} - 1$$

$$obs2 = \left(\frac{x(t)-2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50} - 1$$

$$obs3 = \left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.25}{0.05}\right)^{50} - 1$$

$$obs4 = \left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.05}{0.05}\right)^{50} - 1$$

Natural logarithm is taken on both the sides to make the constraint with the power of 50 smaller. The set of obstacles are modeled in the problem as below,

$$obs1 = ln\left(\left(\frac{x(t)+2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50}\right)$$

$$obs2 = ln\left(\left(\frac{x(t)-2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50}\right)$$

$$obs3 = ln\left(\left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.25}{0.05}\right)^{50}\right)$$

$$obs4 = ln\left(\left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.05}{0.05}\right)^{50}\right)$$

The obstacle path constraint is then taken as,

$$PathConst_{obs} = [obs1; obs2; obs3; obs4]$$

These path constraints are given a minimum value of 0 to prevent any collision of the aircraft with these obstacles. Hence the path generated should satisfy these path constraints (i.e. its value should always be greater than 0) at all the collocation points.

The multiple aircraft landing problem is formulated along with the obstacles in the path. This problem is to be solved as an optimal control problem, to get the desired trajectory satisfying all the constraints. Chapter 3 describes in detail about the RRT algorithm which is used to generate the initial trajectory guess that avoids the obstacles, which acts as the nonlinear path constraints of the problem.

CHAPTER 3
RRT AND MULTIPLE RRT ALGORITHMS

This chapter describes the rapidly exploring random tree (RRT) algorithm, which is a type of randomized shortest path finding algorithm. RRT is especially good for solving problems which involve obstacles as path constraints and kinodynamic or nonholonomic differential constraints of the vehicle. It can be used to generate trajectories for nonlinear systems with state and path constraints. It biases the search to the least explored regions, for effective and faster result. That is, the probability that a point is selected and added to the tree is proportional to the area of its Voronoi region. The Voronoi diagram is formed by partitioning an area with multiple points into convex polygons such that each polygon has only one generating point within it, and all the other point is closer to this generating point than any other point.

There have been many improvements made to the RRT algorithm over the years as seen in literature. This chapter will discuss about these algorithms including the advantages and differences of one over the other. There are three main algorithms described in this chapter: (1) Original RRT algorithm, (2) Bi-directional RRT algorithm, and (3) Multiple RRT algorithm.

The RRT algorithm that is used in this work is called multiple RRT algorithm, which has multiple trees simultaneously generating towards the goal point and trying to connect with each other at every iteration, to ultimately reach the goal point in shortest possible time.

Original RRT Algorithm

The main objective of RRT algorithm is to find a path from point A to point B with obstacles defined in the environment. They are widely used in path planning of robot

arm to avoid an obstacle while reaching another point. The robot arm has to change its orientation and position simultaneously to make the movement. The RRT algorithm is also used in path planning of vehicles while maneuvering through a set of obstacles. The obstacle can be defined in 2D or 3D space. The Figure 3-1 gives a visualization of the RRT tree growing denser as the iterations increase.

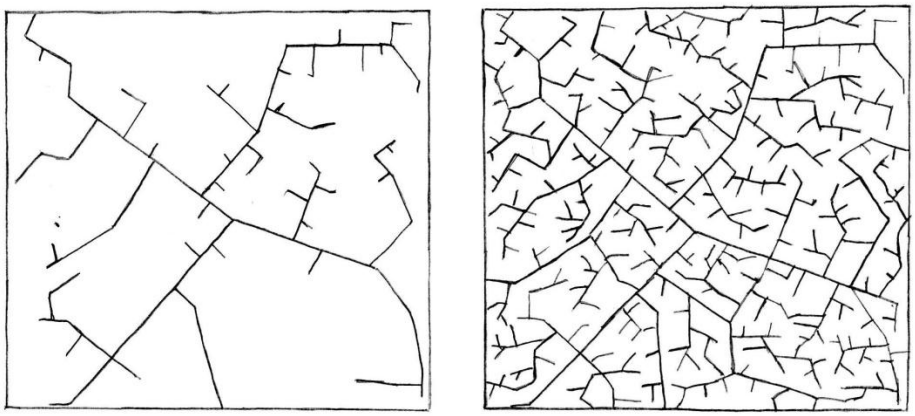

Figure 3-1. A visualization of a RRT graph growing denser with increasing iterations.

In the RRT algorithm that was developed originally, a configuration space, C is defined, in which $p\epsilon\ C$ specifies the position and orientation of the object in its path. C_{free} is the set of free configuration space, where the object can move freely without colliding with the obstacle.

The start point A is taken as p_{start} and final goal point is taken as p_{goal}. The obstacle is defined by specifying its vertex points. The algorithm generates intermediate points randomly, biasing those points towards the Voronoi regions with larger area. It then tries to connect these points with the nearest neighbor vertex/node of the tree, such that no collision occurs while making the connection. The collision is checked by a Collision_Check function. If no collision occurs, the intermediate point becomes a part of

the main tree by adding it as its vertex/node, and creating an edge connecting it to the tree. However if a collision occurs, the algorithm rejects that intermediate random point and selects another point to continue the above mentioned steps, until it reaches the final goal point. The tree is represented by T. Also n is the number of iterations for which the algorithm repeats itself. It represents the maximum number of times the algorithm should find a new intermediate point, to reach the goal point.

Algorithm 3-1 RRT Original (p_{start} , p_{goal})

```
CONSTRUCT_RRT( p_start, p_goal)
 1  T.start ( p_start);
 2  for j=1 to n do
 3        p_rand ← RANDOM_CONFIG( ) ;
 4       ADD ( T, p_goal);
 5  end for
 6  Return T;
```

```
EXPAND ( T, p_goal)
 1  p_near ← NEAREST_NEIGHBOR( p_goal , T);
 2  if NEW_CONFIG ( p_goal, p_near, p_new) then
 3        T.add_vertex ( p_new);
 4        T.add_edge ( p_near, p_new);
 5        if p_new = p_goal then
 6              Return Reached;
 7        else
 8              Return Advance;
 9        end if
10  end if
11  Return Trapped;
```

The Algorithm 3-1 is developed by Kuffner, J.J., and LaValle, S.M. [30]. The RRT grows as follows. A random configuration in the configuration space with position as p_{rand} is chosen initially using the RANDOM_CONFIG function. Then a vertex from the RRT, p_{near} that is nearest to this configuration is found using the

NEAREST_NEIGHBOR function. The NEW_ CONFIG function is called to expand the tree towards the random configuration with a small epsilon value. The new vertex p_{new} and the new edge connecting p_{new} with p_{near} are added to the tree.

If this new vertex is the goal vertex, p_{goal}, then the goal point is reached and a path between the start and the goal vertex is formed. If the new vertex is not the goal vertex, the tree keeps growing, until the goal is reached. Also, if the new vertex cannot join with the nearest vertex due to collision with an obstacle in the path, then the new configuration is rejected and the algorithm returns a 'trapped' message. Thus a new configuration is chosen at every iteration until the RRT reaches the goal vertex.

Bi-Directional RRT Algorithm

Bi-directional RRT is a variation of the original RRT algorithm. It has two trees growing towards each other; one starting from the start point and the other from the goal point. This makes it more efficient and fast in finding the solution.

The main difference between the function RRT_BIDIREC and RRT are listed below-

- Bi-directional RRT grows two trees originating from the source and from the destination, until they meet each other. So both the trees have to be tracked simultaneously at every iteration.
- Both the trees grow towards each other instead of growing towards some random configuration. This increases the probability of finding a solution and also makes it faster.
- In original RRT algorithm, the tree grows towards some random configuration with a single step of epsilon length in each iteration. In bi-directional RRT, the trees grow towards each other with multiple epsilon length steps, thus making the greediness of the algorithm stronger.

The Algorithm 3-2 is also developed by Kuffner, J.J., and LaValle, S.M. [30], which is similar to the original RRT algorithm except for the difference in the number of trees

and an additional function for connecting the two trees. Figure 3-2 depicts a bi-directional RRT algorithm with its two trees growing from the start and goal node.

Figure 3-2. Bi-directional RRT algorithm solution with four obstacles and two trees

The tree, T_a has the start point as its initial vertex and the tree, T_b has the goal point as its initial vertex. The RRT-BIDIREC function calls an additional function called CONNECT. This CONNECT function only iterates over the EXPAND function until the connection is made and the EXPAND function returns 'Advanced' message.

First a random configuration is chosen. The EXPAND function is called passing T_a and the goal point, p_{goal} as its parameters. The nearest vertex to p_{rand} is chosen from the tree, T_a such that it tries to grow the tree, T_a towards the goal point, p_{goal}. If there are no collisions with the obstacles, it goes to the next step.

In this step the tree, T_b is passed to the CONNECT function which calls the EXPAND function iteratively to grow T_b towards the new vertex, p_{new} that is added to the tree, T_a. If the CONNECT function fails to reach this new vertex due to an obstacle in between (i.e. the two trees fails to connect with each other), then the roles of the two trees are exchanged by using the SWAP function, which swaps the two trees. The SWAP function which is called in RRT_BI-DIREC function, takes two parameters, T_a and T_b trees.

Algorithm 3-2 RRT Bi-directional (p_{start}, p_{goal}, T_a, T_b)

```
RRT_BI-DIREC(p_start, p_goal)
 1  T_a.start (p_start); T_b.start (p_goal);
 2  for j=1 to n do
 3      p_rand ← RANDOM_CONFIG( ) ;
 4      if not (ADD (T_a , p_goal)= Trapped) then
 5          if (CONNECT(T_b, p_new) = Reached) then
 6              Return PATH (T_a, T_b);
 7          end if
 8      end if
 9      SWAP (T_a, T_b);
10  end for
11  Return Failure;
```

```
EXPAND ( T, p_goal)
 1  p_near ← NEAREST_NEIGHBOR( p_goal , T);
 2  if NEW_CONFIG ( p_goal, p_near, p_new) then
 3      T.add_vertex ( p_new);
 4      T.add_edge ( p_near, p_new);
 5      if p_new = p_goal then
 6          Return Reached;
 7      else
 8          Return Advance;
 9      end if
10  end if
11  Return Trapped;
```

```
CONNECT ( T, p_goal)
 1  repeat;
 2      Q ← EXPAND ( T, p_goal);
 3  until not ( Q = Advanced);
 4  Return Q;
```

Now the tree with initial vertex as p_{goal} tries to grow towards the goal point and the tree with initial vertex as p_{start}, tries to connect with the new vertex of the other tree. Hence the roles of the 2 trees are swapped in each iteration, with the first tree trying to

connect with the goal point and the second trying to connect with the first tree; until the two trees connect with each other.

An important advantage of this approach is that it is very flexible with its multiple epsilon steps and the role swapping. There can be many possible variations in the solution. It explores a different solution every time and so it not a single solution approach. Also the solution is obtained much faster in a cluttered environment and slightly faster in a more cluttered environment. It can find solution to 2D or 3D problem with obstacles in seconds.

Hence Bi-directional RRT is an improved version of the original RRT algorithm which is developed for faster convergence and for use in 2D and 3D problems. However the main drawback of this approach is that there a lot of nearest neighbor and collision check function calls to be made, in order to grow the trees and connect them in each iteration; thus making it computationally more expensive.

Multiple RRT Algorithm

The RRT and Bi-directional RRT algorithm are adept at solving very difficult and high-dimensional space path planning problems. But increasing the obstacles in the environment can severely affect the convergence in these algorithms. A single or two RRTs does not affect the rate of convergence to a solution, which is largely determined by shape and position of the obstacle in the environment. For example, if a start and goal point is separated by a wall with a narrow hole or passageway in it, the RRT can get lost on one side of the obstacle. This can increase the number of iterations exponentially, thus increasing the calls to the nearest neighbor and collision check function. This can increase the computation effort and time significantly.

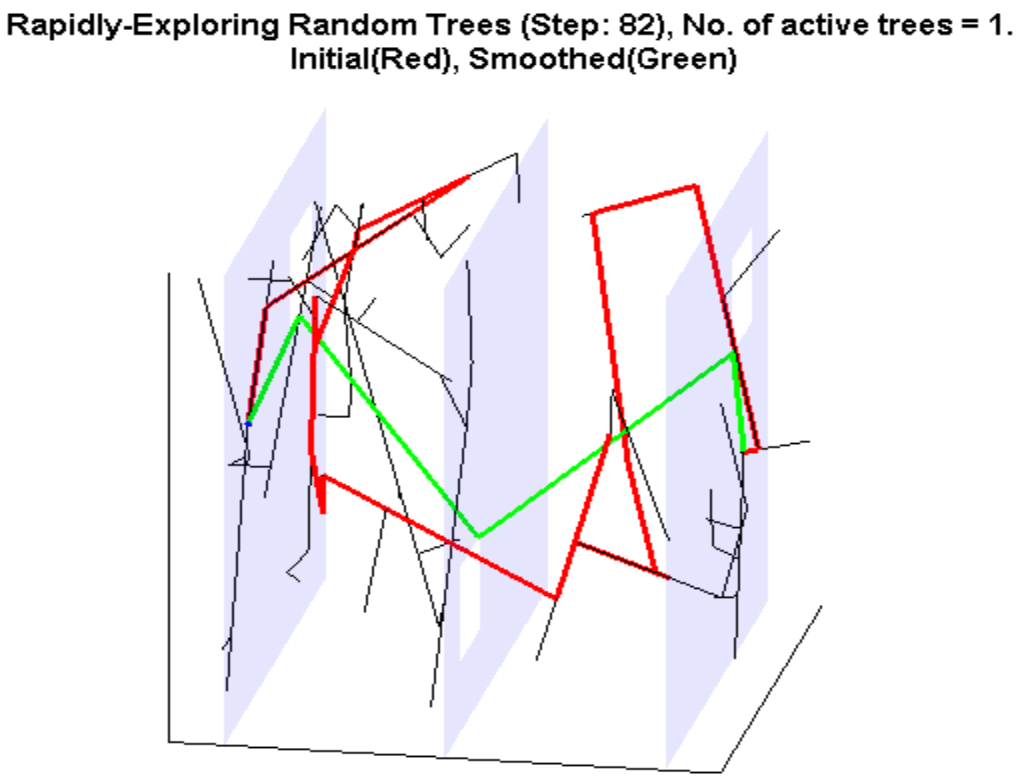

Figure 3-3. Multiple RRT algorithm solution with three wall obstacle

The multiple RRT algorithm is proposed by Clifton, M. and others [31] to address these issues. The paper introduces the method of using multiple trees and thus reducing the logarithmic complexity of the search in a high obstacle density environment. Figure 3-3 shows a multiple RRT algorithm solution with three wall obstacles. The green colored solution is the path found after smoothening of the initial red colored path is done.

The pseudo code of an RRT and multiple RRT algorithm is given in Algorithms 3-3 and 3-4. The RRT pseudo code has only a single tree, which grows in every iteration towards the goal point. In multiple RRT algorithm, after every collision check, a new tree is created if a collision occurs. These trees try to connect with only the nearest neighbor node of the other trees at each iteration. This way it eliminates the need to connect with

all the nodes of all the trees and hence reduces the computationally expensive collision checks for each connection. Also it only creates a new tree if it is required and tries to merge these trees at each iteration, to reduce the total number of trees.

Algorithm 3-3 RRT

```
do while Path not found
    NEW_POINT
    NEAREST_NEIGHBOR
    COLLISION_CHECK
    if no Collision
        CONNECT
        Path Found
    end if
end do
```

Algorithm 3-4 Multiple RRT

```
do while Path not found
    NEW_POINT
    for each tree
        NEAREST_NEIGHBOR
        COLLISION_CHECK
        if no Collision
            CONNECT
        else
            NEW TREE
        end if
    end for
end do
```

The algorithm initially places discretely positioned seeds throughout the environment, which may or may not become the roots of new trees. It can be used for both 2D and 3D search space with obstacles. It tries to automatically connect to the goal and does path smoothing to find the shortest possible path to the goal. The user can specify the maximum number of trees to be created, with a minimum of 2. The user

can also specify the number of seeds to be placed on each axis. Also the user can specify the maximum number of RRT search iterations to be done to find the path. The search space is specified by giving its limits in each axis, and also the start and goal locations have to be specified within this search space. The obstacles can be specified in the form of walls, with more than one wall allowed in the environment. The vertices of the obstacles are specified in an obstacle array file, which can be updated with new obstacles at any time.

New_Point function finds a point randomly from the search space. The Nearest_Neighbor function finds the vertex of the tree that is nearest to the new point. The Collision_Check functions is called to check if there is any collisions with the obstacles while connecting the new point with the nearest neighbor vertex. If no collision occurs, the new point is added to the tree as its new vertex and along with the edge joining it with the nearest neighbor. The tree also tries to connect with the nearest neighbors of the other trees in the process. If a collision occurs, it creates a new tree with the new point as its root. The process repeats until a tree reaches the goal point. Once the goal point is reached, the path is smoothened to give the shortest path to the goal.

The rate of convergence is greatly improved by using multiple tree RRT algorithm especially in environments with high obstacle density. Also the computational efficiency is better when compared to the bi-directional RRT algorithm due to fewer collision checks and less number of iterations. Another advantage of multiple RRT algorithm is that, the search is not restricted to near the start and goal points, and hence it can be useful for path planning with multiple goal points.

Application of RRT in Multiple Aircraft Landing Problem

Multiple RRT method has the advantage of reducing the computational complexity by using multiple trees which try to connect with each other in every iteration. In the application proposed, this multiple RRT is used to find the trajectory to be taken by multiple aircraft while they are about to land. A new algorithm is developed which takes into account the path constraints of avoiding the obstacles initially and is discussed in Chapter 4. The aircraft are sorted in order of their distance from the goal and the nearest aircraft lands first followed by the next nearest aircraft. This algorithm not only finds the shortest distance path of all the aircraft but also satisfies additional constraints of keeping a minimum distance and time between aircraft landings.

CHAPTER 4
INITIAL TRAJECTORY GENERATION USING RRT

Parameters for the Problem

The problem in this work is three-dimensional, so multiple RRT with 3D trajectory generation is used. The obstacles are defined in the obstacle text file of the RRT algorithm as vertices of a quadrilateral. This text file can be updated anytime with a new set of obstacles, and the proposed algorithm can be executed again to get a new trajectory, taking into consideration the new obstacles. A simple RRT algorithm is used to find the shortest path from a start point to a goal point avoiding the obstacles. However in this work, the RRT algorithm is used to find the trajectory of each of the aircraft, avoiding obstacles and maintaining the minimum safe distance between each landing.

But the problem into consideration has to include many constraints in the trajectory generated by the RRT algorithm. Also the trajectories of the aircraft are related to each other. That is the trajectory of all the aircraft has to be generated taking into consideration the trajectory of the previous aircraft, to include the minimum distance and time difference constraint in its landing. Also intersecting flight paths should be avoided.

So another algorithm is developed in this work, which takes into consideration all these constraints for any number of aircraft. However, increasing the number of aircraft would increase the computation time. In this work, three aircraft and eight obstacles were taken into consideration. The user has to specify the number of aircraft, the obstacles and the initial positions of the aircraft. The initial position of all the aircraft can be found using the radar positioning system. The time at which the initial position is found, is taken as the initial time, $t = 0$ in the problem.

Algorithm for Initial Trajectory Generation

Initialization

- The number of aircraft approaching to land is given as n. (n=3 , in this problem)
- The initial location of all the aircraft is measured using radars signals. This initial location is calculated such that it gives the position in which the aircraft would be after 60 seconds of moving in its own trajectory. This may be the computation time taken to pre-calculate trajectory for the aircraft.
- The goal location is specified, which is the position the aircraft should reach to start its descent to the runway. This is considered as the same for all the aircraft (taking into consideration only one runway).
- The limit, lim specifies the airspace within which all the aircraft should reach to be considered for landing. (i.e the start positions of all the aircraft should be within this limit).

Identification of the Nearest Aircraft

- Initially the RRT algorithm is used to find the shortest distance path from each of the start points of the aircraft to the goal point.
- The RRT algorithm gives only the way points of the path as its output, so the total distance from the start to the goal point is found by adding up the distance between two successive points.
- Based on the total distance between start and goal point, the aircraft with the nearest distance from the goal point is found. These aircraft are then sorted based on the minimum distance required to reach the runway while avoiding obstacles.

Specification of the Constraints

- Keeping the path of the first nearest aircraft as the same, the trajectory of rest of the aircraft has to be manipulated based on the constraints specified.
- For this, a waypoint is chosen randomly within the limit specified by the user initially, so that the rest of the aircraft (leaving the first aircraft) reach this way point first, before reaching for the goal point.
- This point is chosen such that the total distance for these aircraft to reach the goal point, should satisfy the constraints given below -

    1. Minimum distance between two successive aircraft landing.
    2. Maximum distance between two successive aircraft landing.

3. The distance from the start point to way point should be greater than half the total distance from start to goal point.

- The first two constraints gives a margin of safety that can be specified, while the third constraints is used to see if the way point is chosen wisely, instead of just choosing it randomly. It helps in preventing intersecting trajectories at the same time, thus avoiding collisions.

Selection of the Way Point

- The waypoint is chosen repeatedly to satisfy these constraints. To improve this search, it is made sure that none of the waypoints selected are the same as those selected before. For this the waypoints are stored in an array, and its elements are compared to the new random waypoint, to check that they are not equal.

- The RRT algorithm is used to find the trajectory from the start point to the waypoint. Also the distance from the start to the way point is found as mid-distance.

- This RRT algorithm is used again to find the trajectory from the waypoint to the goal point. The distance from waypoint to the goal point is found. This distance is summed with the mid distance to obtain the full distance.

- If all the constraints are met, it exits out of the loop; else it keeps searching for a suitable waypoint to satisfy all the constraints.

- The total trajectory of the aircraft is found by adding the trajectory generated by both the RRT algorithm. This is repeated for all the aircraft except for the nearest aircraft, to find the new trajectory, which satisfies all the constraints.

Aircraft Trajectory and Time

- The new trajectory of the aircraft, except for the one nearest to goal point is found.

- As the speed is assumed to be constant for all the aircraft, the time at the end of each point can be found based on the distance travelled from one point to another, by using the speed, distance, time relationship.

- However, the new trajectory may have one or more repeated points, (the waypoint will be repeated at least twice due to summing the trajectories). Such repetitions of points in the trajectory are removed and the new times associated with these points are found.

- Also the time at which the waypoint is to be reached is found, and the index value of this time, from the time array is found.

Advantages and Disadvantages of the Initial Trajectory Generated

The initial trajectory of the aircraft and the time associated with it is found using the developed algorithm. The time taken by aircraft is the minimum possible time it takes to reach the goal avoiding the obstacle and without considering the aircraft dynamics. It can be used to find the initial trajectories of any number of aircraft, which has to be specified along with the start positions of the aircraft. Also any number of 2D or 3D obstacles can be added in the trajectory. In real time, the obstacles can be updated to generate a new initial trajectory.

As the way points are selected randomly, the trajectory generated every time the algorithm is executed, will not be the same. Hence the algorithm does not restrict itself, to only one solution. Instead it searches for a new trajectory each time it is executed. However, this also means that the computation time to find the initial trajectory may vary and is not fixed.

Trajectory Plots

Three-Dimensional Trajectory

The 3D trajectory of all the aircraft can be plotted using the plot3 option in Matlab. The plot in Figure 4-1 shows the trajectory of the three aircraft in three-dimension. The original trajectory and the new trajectory that is generated for the second and the third aircraft using the algorithm developed are shown in Figure 4-1. It is clearly seen, that by going through the way point, the trajectory of the second and third aircraft becomes longer than their original trajectory. The new trajectory satisfies all the minimum and maximum distance difference constraints between each aircraft landing.

Check for Intersecting Trajectories

The new trajectory generated should not only avoid obstacles in the path, but also should ensure that the trajectories do not intersect at any point of time, thus avoiding collisions. To check this, the x, y, and z positions of all the aircraft with respect to time is plotted.

From Figures 4-2, 4-3, and 4-4 it can be seen that though there may be overlaps in the y and z positions at different points of time, the x position does not intersect at any point of time. This shows that the trajectory does not intersect with the same x, y, and z positions at any instance of time. Also even if the x position intersects at any point of time, the instant at which the intersection occurs should match with the instant of intersection in the y and z positions too, to have intersections in the trajectories. This condition almost never occurs because of the way the constraints are formulated. Hence the algorithm also avoids any intersecting trajectories and thus avoids any collisions between aircraft during landing.

The initial trajectory generated can be used for initializing the optimal control problem. The aircraft can be arranged in the order of its distance from the goal point, so that the time constraints can be specified accordingly. The initial trajectory takes care of the obstacle avoidance constraint, which forms the path constraint in the optimal control problem. This helps in better convergence of the problem.

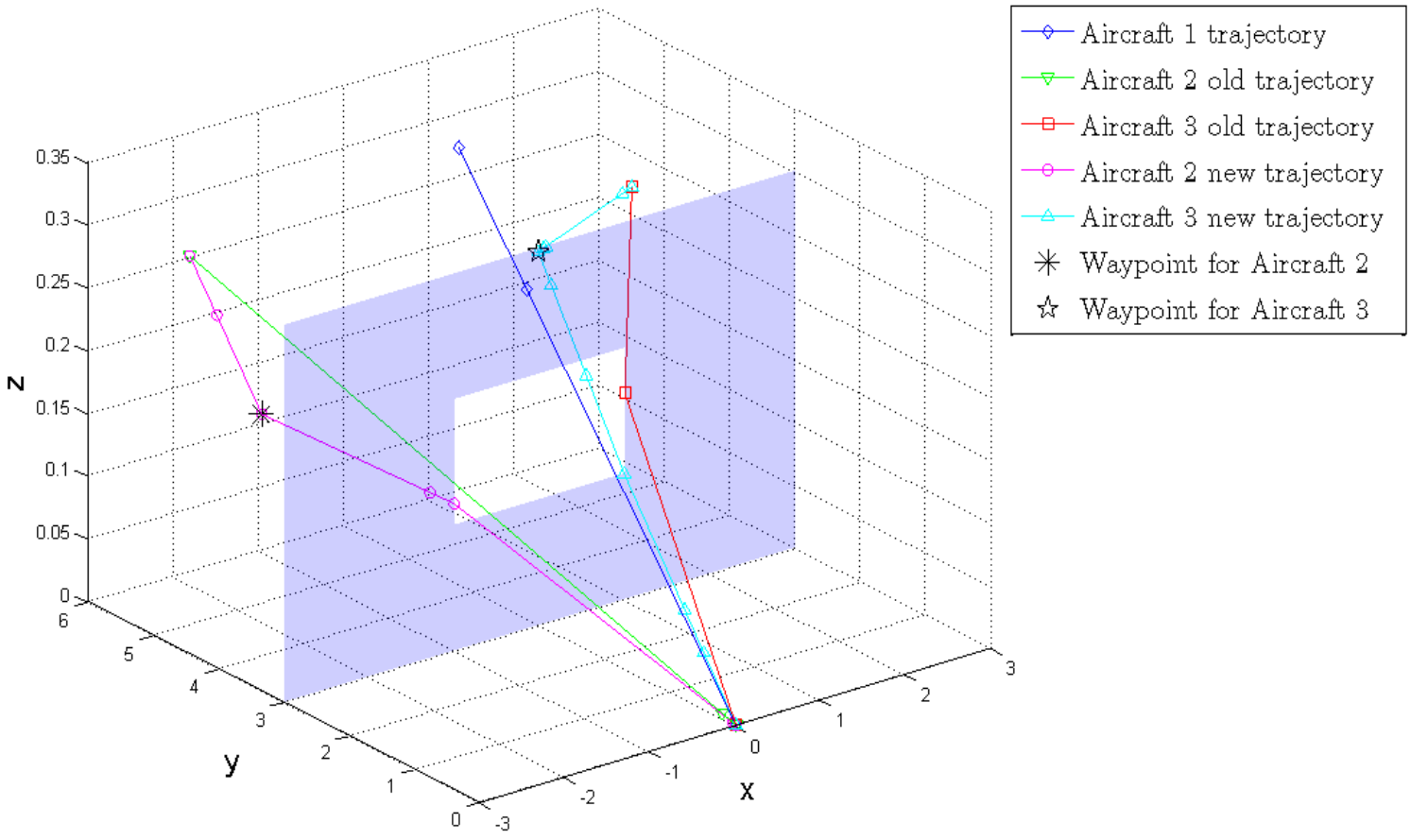

Figure 4-1. Initial three-dimensional trajectory of the three aircraft using the RRT and proposed algorithm

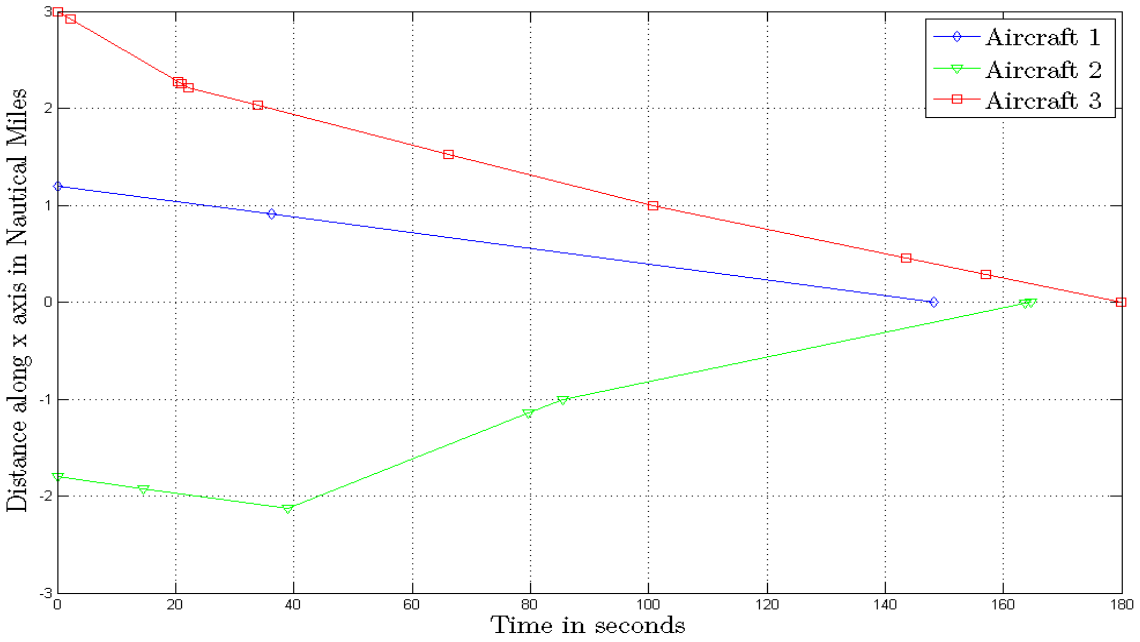

Figure 4-2. X-axis vs. Time plot for the three aircraft using the RRT and proposed algorithm

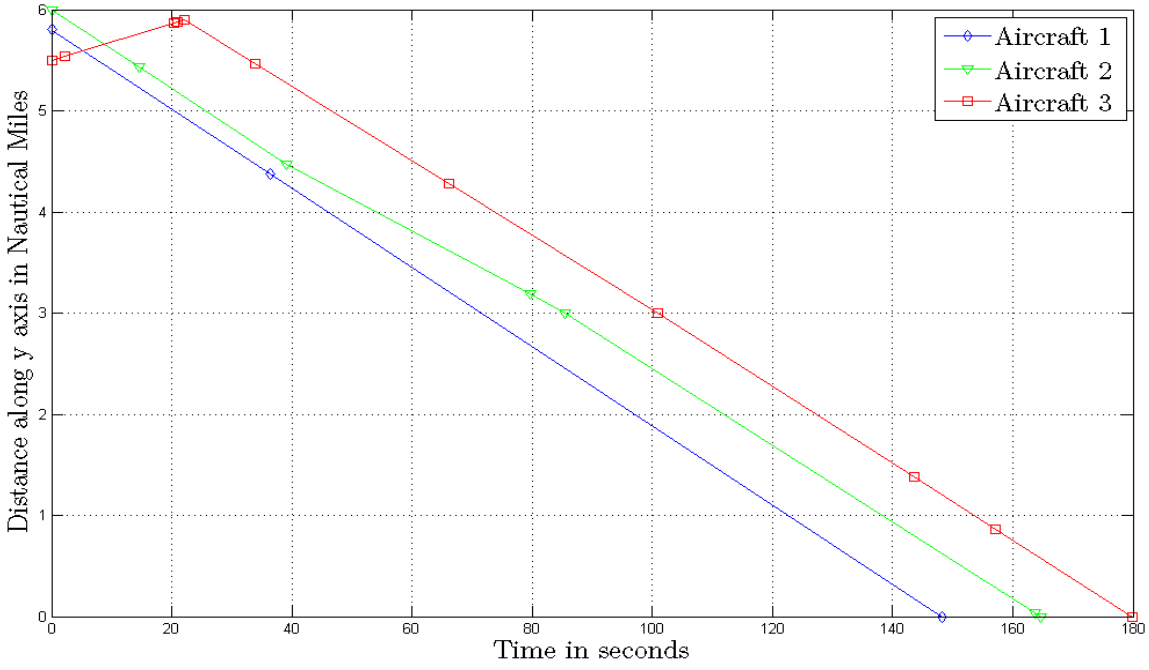

Figure 4-3. Y-axis vs. Time plot for the three aircraft using the RRT and proposed algorithm

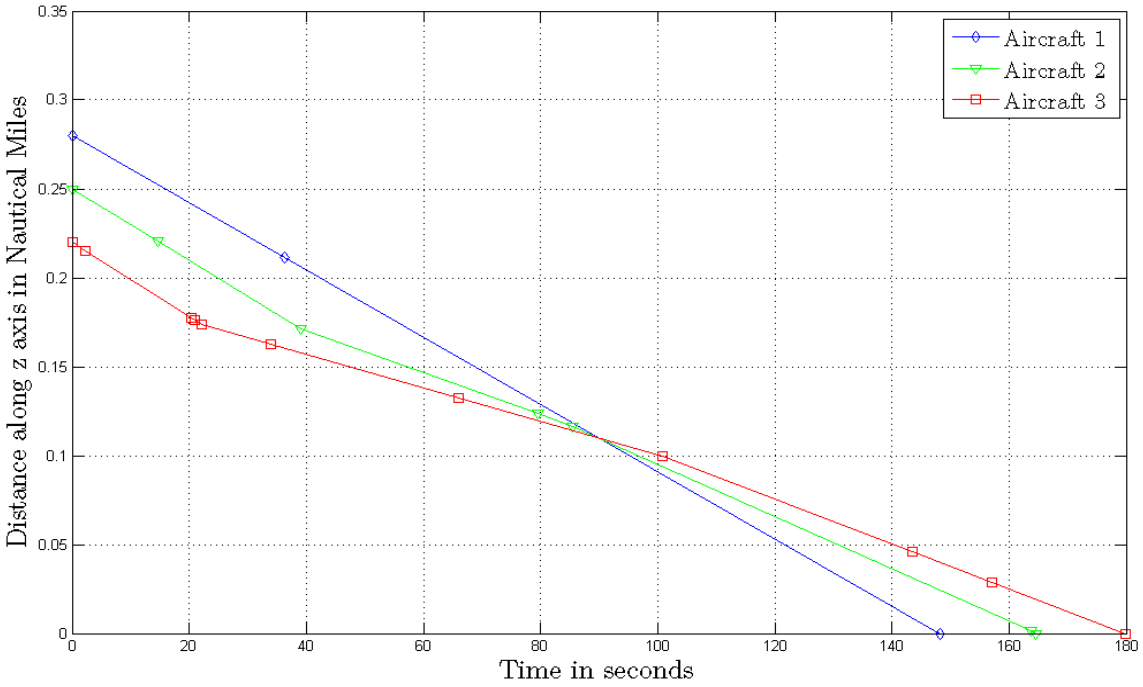

Figure 4-4. Z-axis vs. Time plot for the three aircraft using the RRT and proposed algorithm

CHAPTER 5
PSEUDOSPECTRAL METHODS AND GPOPS

Pseudospectral methods are a direct trajectory optimization method which uses direct collocation to transcribe the optimal control problem to a nonlinear programming problem (NLP). The optimal control problems can be solved using collocation at Legendre-Gauss(LG), Legendre-Gauss-Radau (LGR), and Legendre-Gauss-Lobatto (LGL) points is presented by Garg, D. and others [27].

Pseudospectral methods uses global polynomials to parameterize the state and control and then uses the nodes of the Gaussian quadrature to collocate the differential algebraic equations. It provides accurate solutions for problem with smooth solutions. For problem with solutions which are not smooth, the time interval from [-1,1] is broken into several intervals, so that a different global polynomial approximation is used over these intervals.

LG, LGR, and LGL Collocation Points

There are three most common sets of collocation point which are obtained from the roots of a Legendre polynomial and/or linear combinations of a Legendre polynomial and its derivatives; they are Legendre-Gauss (LG), Legendre-Gauss-Radau (LGR), and Legendre-Gauss-Labatto (LGL) points. All the three sets are defined in the domain of [-1,1]. The major difference is that the LG points do not include any endpoints, the LGR points include one of the endpoints, and the LGL points include both of the endpoints. Also the LGR points are asymmetric relative to the origin and can be defined using the endpoint as either the initial point or the end point. Figure 5-1, shows a schematic of the LGL, LGR and LG collocation points.

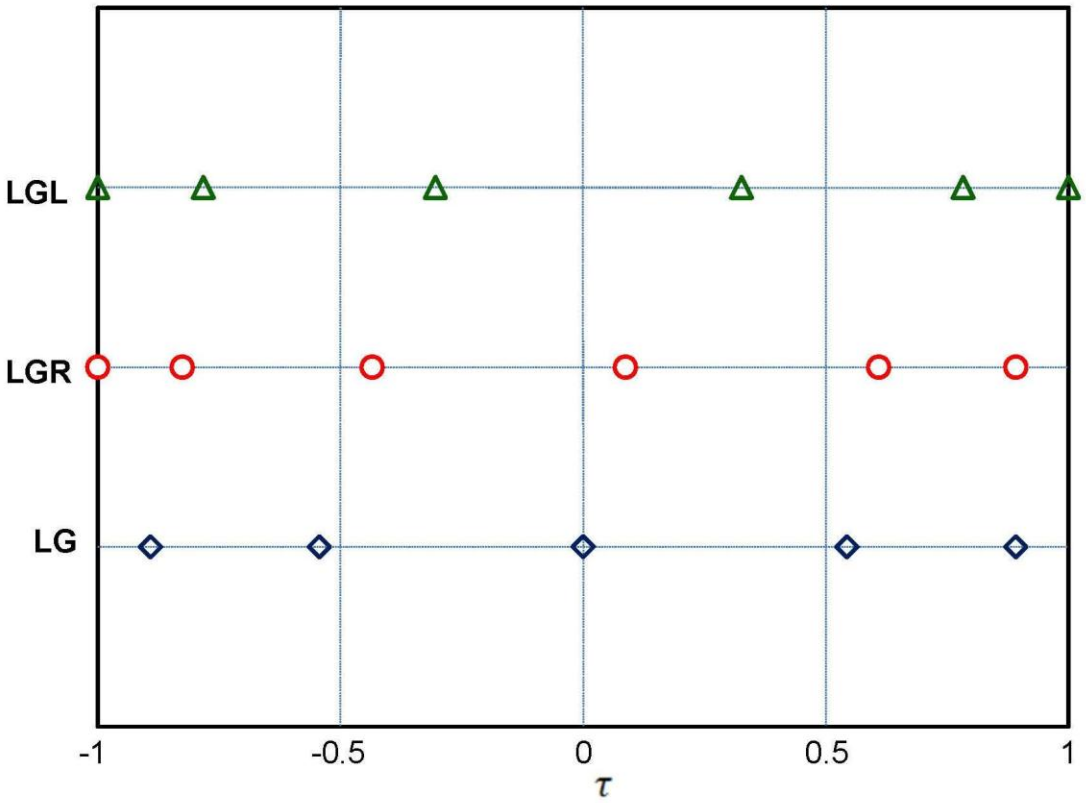

Figure 5-1. Schematic showing LGL, LGR and LG collocation points

Let N be the number of collocation points and $P_N(\tau)$ be the N^{th}-degree Legendre polynomial, then the LG, LGR and LGL collocation points are obtained from the roots of the polynomial. LG points are the roots obtained from $P_N(\tau)$, LGR points are the roots obtained from $P_{N-1}(\tau) + P_N(\tau)$, and LGL points are the roots obtained from $P_{N-1}(\tau)$ together with the points -1 and 1

Formulation of Pseudospectral Method Using LGR Points

A method for direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using global collocation at Legendre-Gauss-Radau (LGR) points is shown by Garg, D. and others [26]. The authors have shown that the use of LGR collocation helps in determining accurate primal and dual

solutions for both finite and infinite-horizon optimal control problems. So the Radau Pseudospectral method is used in this work.

To simplify the problem, an unconstrained optimal control problem on the time interval ∈ [−1,+1] with a terminal cost is taken. The time interval can be transformed from [−1, 1] to the time interval $[t_0,\ t_f]$ via the affine transformation,

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}$$

The goal is to determine the state $x(\tau) \in \mathbb{R}_n$ and the control u(τ) $\in \mathbb{R}_m$ which minimize the cost functional,

$$J = x(1)$$

subject to the constraints,

$$\frac{dx}{d\tau} = f\big(x(\tau), u(\tau)\big), \qquad x(-1) = x_0,$$

where $f: \mathbb{R}^n * \mathbb{R}^m \to \mathbb{R}^n$ and x_0 is the known initial condition.

Let there be N LGR collocation points, $\tau_1, \tau_2, \tau_3, \ldots\ldots \tau_N$ described in the interval [-1,1], where $\tau_1 = -1$ and $\tau_N < +1$. A new non-collocated point $\tau_{N+1} = 1$ which is used to approximate the state variable is introduced [26]. Each component of state x is approximated by a Lagrange polynomial, L_i as given in equation below, with $i = 1, \ldots, N+1$.

$$L_i(\tau) = \prod_{\substack{j=1 \\ j \neq i}}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j},$$

The j-th component of the state is approximated as a series,

$$x_j(\tau) \approx \sum_{i=1}^{N+1} x_{ij} L_i(\tau)$$

Differentiating the above series at the collocation points τ_k gives,

$$\dot{x}_j(\tau_k) \approx \sum_{i=1}^{N+1} x_{ij}\dot{L}_i(\tau_k) = \sum_{i=1}^{N+1} D_{ki}x_{ij},$$

where $D_{ki} = \dot{L}_i(\tau_k)$

The N by N+1 matrix D is called the radau differentiation matrix. Collocating the dynamics at the 'N' LGR collocation points gives,

$$DX = F(X^{LGR}, U^{LGR})$$

The finite-dimensional NLP is then formulated as below,

Minimize $\phi(X_N)$

subject to the system dynamics, $DX = F(X^{LGR}, U^{LGR})$, $X_0 = x_0$

This system dynamics is then rewritten as,

$$D_{1:N}X_{1:N} = F(X^{LGR}, U^{LGR}) - D_0x_0$$

where $D_{1:N}$ is the N by N differentiation matrix which is invertible.

hp-Adaptive Pseudospectral Method

An *hp*-adaptive pseudospectral method with collocation at Radau points is chosen in this work. This method is proposed by Darby, C.L. and others [28], which iteratively and simultaneously determines the number of segment breaks, the width of each segment, and the polynomial degree required in each segment for approximation, until the user-specified accuracy is achieved. It leads to higher accuracy solutions with less computational effort and memory, than is required in global pseudospectral methods. Also the Radau points are used, as it is more computationally efficient and well posed. As the problem in this work, has multiple aircraft and also has an obstacle wall in the path to be avoided, the problem is broken into several overlapping phases, for each

aircraft to cross the obstacle at the desired location and then reach the goal. Due to the complexity of the problem, the *hp*-adaptive method is chosen.

Multiple Aircraft Landing Multiple-Phase Problem Formulation

Objective and Input Parameters

The objective is to land the three aircraft one after the other, while maintaining a minimum time and distance separation between successive aircraft landings. Also the initial and final desired position and orientation specified in the problem has to be achieved.

The final position in this problem is the same for all the aircraft, as they have to land on a common single runway. This position is [0, 0, 0], and the limits of the environment within which the aircraft has to be is [-3 3; 0 6; 0 0.3] with the distance considered in nautical miles. Along x-axis, it ranges from -3 to 3 nautical miles (Horizontal breadth), along y-axis it ranges from 0 to 6 nautical miles (Horizontal length) and along the z-axis it ranges from 0 to 0.3 nautical miles (Vertical height). The initial and final time is taken in seconds. The flight path and heading angle of the aircraft are taken in radians in the problem and specified in degree in Table 5-1.

Table 5-1. User-specified initial and final states of the Aircraft 1, 2, and 3

Aircraft	t_o	x_0	y_0	z_0	γ_0	φ_0	t_f	x_f	y_f	z_f	γ_f	φ_f
1	0	1.2	5.8	0.28	0	-15	155	0	0	0	0	-90
2	0	-1.8	6	0.25	5	-10	170	0	0	0	0	-90
3	0	3	5.5	0.22	0	-105	185	0	0	0	0	-90

Methodology

The problem of aircraft landing is solved using three aircraft. The problem is divided into 6 overlapping phases. In Phase I, all the three aircraft are hovering before the obstacle wall and are assumed to start at the same time at t=0. At the end of the

Phase I, the nearest aircraft reaches the obstacle wall. In Phase II, only the second and third aircraft are hovering before the obstacle wall. At the end of Phase II, second aircraft reaches the obstacle wall. Phase III has only the third aircraft still hovering before the obstacle wall. Phases IV, V and VI depict the three aircraft landing and reaching the goal (runway). In Phase IV, the first aircraft is landing; the Phase V has the second aircraft landing, while the final Phase VI has the third aircraft landing. Thus the problem is divided into the six phases.

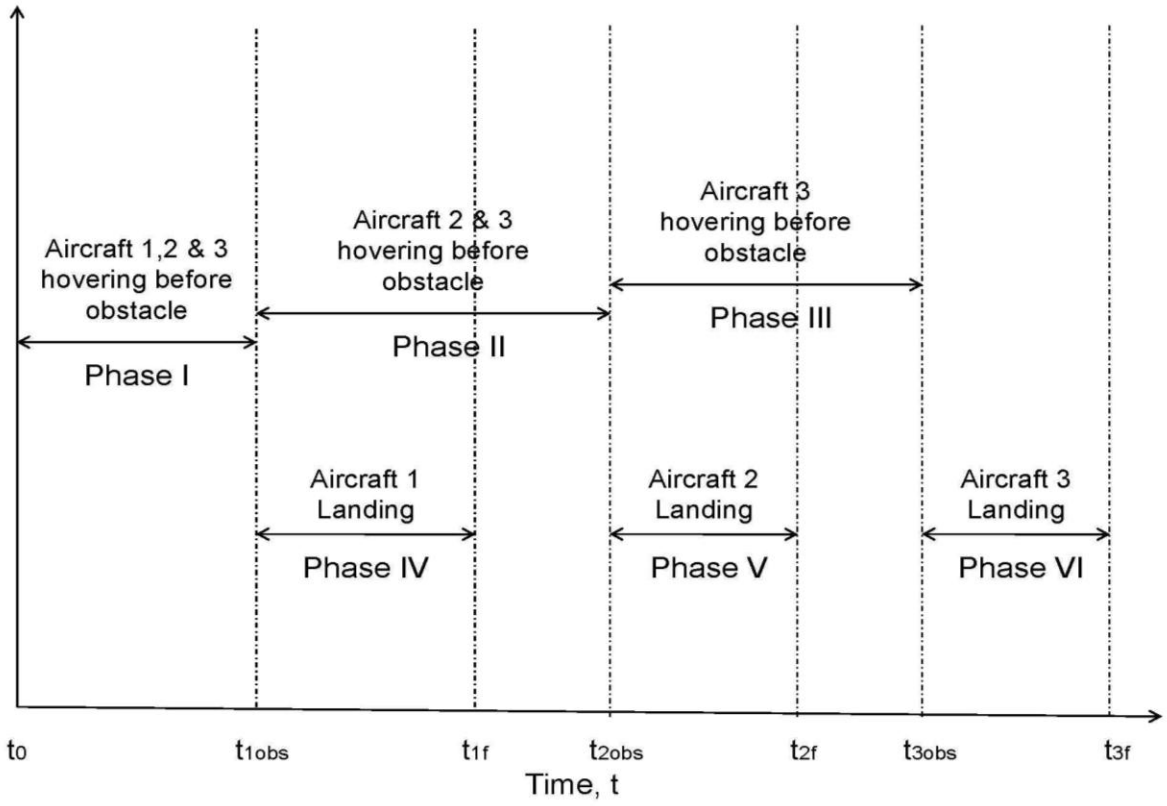

Figure 5-2. Schematic showing the six phases in the problem

These phases can be seen in Figure 5-2, where t_{1f}, t_{2f} and t_{3f} are the final landing time of Aircraft 1, 2 and 3 respectively; t_{1obs}, t_{2obs} and t_{3obs} is the time taken by Aircraft 1, 2 and 3 to reach the obstacle wall respectively.

These phases are linked with each other in GPOPS using the linkage constraints, in order to make the states continuous. In this problem, as seen in Figure 5-1, at time, t_{1obs}, the first five states of Aircraft 1 in Phase I is linked with Phase IV and the last ten states of Aircraft 2 and 3 in Phase I is linked with Phase II. At time, t_{2obs}, the first five states of Aircraft 2 in Phase II is linked with Phase V and the last five states of Aircraft 3 in Phase II is linked with Phase III. At time, t_{3obs}, all the states of Aircraft 3 in Phase III is linked with Phase VI.

The dynamics of each of the aircraft are given separately. This gives an advantage of generating trajectories for aircraft with different dynamics such as different types of aircraft, helicopter, small UAVs and other air vehicles. In the example problem, Phase I has totally fifteen states, with five states for each of the three aircraft and has total of six controls (two for each aircraft). Similarly Phase II has a total of ten states and four controls for Aircraft 2 and 3. The Phases III, IV, V, and VI have five states and two controls for a single aircraft.

Cost Function Formulation

The cost function for the problem with 'N' number aircraft is formulated as,

$$J = \sum_{i=1}^{N} -t_{obs}^{ii} + \sum_{j=1}^{2N} \sum_{k=1}^{n} \int_{t_o}^{t_f} \left(\frac{d\gamma^{jk}}{dt}\right)^2 + \left(\frac{d\varphi^{jk}}{dt}\right)^2 dt$$

where t_{obs} is the time taken by the aircraft to reach the face of the obstacle, N is the total number of aircraft taken into consideration, n is the number of aircraft present in Phase j, and 2N is the total number of phases formulated in the problem.

The Mayer cost function is the time for the aircraft to reach the obstacle wall, which is being maximized in this problem. The Lagrange cost in this problem is

minimizing the change in flight path angle and heading angle of the aircraft. Maximizing the time to reach the obstacle and fixing the final time of landing of the aircraft, ensures that the aircraft satisfies the time difference constraints by hovering around before it reaches the obstacle wall, and takes an almost straight path to the goal point, after it passes the obstacle wall. Also minimizing the change in flight path and heading angle ensures less control effort required by the aircraft and also generates smooth aircraft trajectories.

Constraints Formulation

There are different constraints specified in each of the phases depending on the number of aircraft in that phase, the position of aircraft (reaching the obstacle wall or landing at the runway), and the aircraft dynamics. There are four main constraints that are taken into consideration (1) Obstacle path constraints, (2) Intersection constraints, (3) Event Constraints, and (4) Bank angle constraint.

Obstacle path constraints formulation

The path constraint that is common to all the phases is the obstacle wall which has to be avoided in the aircraft trajectory. As each aircraft has different dynamic states, the obstacle constraints have to be modeled for every aircraft that present in each phase. In the example discussed, Phase I has twelve obstacle path constraints, four for each of the three aircraft. Similarly, Phase II has eight obstacle path constraints, and the rest has four obstacle path constraints.

As seen before, the obstacles are modeled as below,

$$obs1 = \ ln\left(\left(\frac{x(t)+2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50}\right) \geq 0$$

$$obs2 = ln\left(\left(\frac{x(t)-2}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.15}{0.15}\right)^{50}\right) \geq 0$$

$$obs3 = ln\left(\left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.25}{0.05}\right)^{50}\right) \geq 0$$

$$obs4 = ln\left(\left(\frac{x(t)+0}{1}\right)^{50} + \left(\frac{y(t)-3.0005}{0.0005}\right)^{50} + \left(\frac{z(t)-0.05}{0.05}\right)^{50}\right) \geq 0$$

Intersection constraints formulation

The intersection constraints has to be taken care of in the Phases I and II, as there are more than one aircraft flying in these phases. The example has all the three aircraft flying in Phase I. The intersection constraints are modeled by finding the distance between the aircraft at every point of time, which has to be always greater than the safe separation distance. In this problem, the safe distance is taken to be 0.25 nautical miles. The distance between the first and second aircraft forms the first intersection constraint; similarly distance between first and third aircraft forms the second intersection constraint; and the distance between second and third aircraft forms the third intersection constraint. In Phase II, however there is only one intersection constraint which is the distance between the second and the third aircraft. Let A1 be Aircraft 1, A2 be Aircraft 2 and A3 be Aircraft 3. Now let,

inter11 = Minimum distance between A1 and A2 in Phase I,

inter12 = Minimum distance between A1 and A3 in Phase I,

inter13 = Minimum distance between A2 and A3 in Phase I,

inter21 = Minimum distance between A2 and A3 in Phase II.

then the intersection constraint in Phase I is specified as,

$PathConstI_{inter} = [inter11;\ inter12;\ inter13] \geq$ [0.25; 0.25; 0.25]

and intersection constraint in Phase II is specified as,

$$PathConstII_{inter} = [inter21] \geq [0.25]$$

Event constraints formulation

The event constraints are enforced at the end of Phase I, II, and III on Aircraft 1, 2 and 3 respectively. These constraints are formulated to ensure that the aircraft passes only through the hole in the obstacle wall and it should be oriented parallel to the ground and perpendicular to the wall, when it reaches the face of the obstacle wall.

To ensure the x, y, z positions are within the limit of the hole in the wall, the constraints on these positions are specified at the time the aircraft reaches the face of the obstacle wall.

$$-1 \leq x \leq 1$$

$$3 \leq y \leq 3$$

$$0.1 \leq z \leq 0.2$$

To satisfy the orientation constraint of the aircraft, the flight path angle has to be oriented parallel to the ground and the heading angle should be exactly perpendicular to the obstacle wall. This is given by,

$$\gamma = 0°$$

$$\varphi = -90°$$

Bank angle constraints formulation

The control is constrained by constraining the bank angle of the aircraft in its trajectory. The bank angle, μ is related to the control as given below,

$$\mu = \tan^{-1}\left(\frac{n_h}{n_v}\right)$$

where n_h is the horizontal load factor and n_v is the vertical load factor of the aircraft. In the example, the bank angle of all the three aircraft is constrained as $-45° \leq \mu \leq 45°$ in degree angle. To enforce this constraint, the inverse tangent of ratio of the two controls has to be in the range, $[-45°, 45°]$.

$$-45° \leq \tan^{-1}\left(\frac{n_h}{n_v}\right) \leq 45°$$

As Phase I has three aircrafts, there are three bank angle constraints to be satisfied, Phase II has two aircrafts with two bank angle constraints, and all the other phases has a single aircraft with a single bank angle constraint.

The results are plotted and discussed in Chapter 6. The three-dimensional trajectory is plotted, along with all the states and controls of the three aircraft. These results are analyzed and validated.

CHAPTER 6
RESULTS AND DISCUSSION

The multiple aircraft landing problem with three aircraft is solved by converting the optimal control problem into a nonlinear programming (NLP) problem using GPOPS. This NLP is then solved using SNOPT. The solution obtained for the optimal control problem is plotted. Also the results are analyzed and validated in this chapter.

Analysis of Results

The trajectory of the three aircraft is plotted in a three-dimensional plot in Figure 6-1. It portrays the trajectory of each of the three aircraft from start to the goal point. The blue window shows the obstacle wall in the environment which the aircraft has to avoid in its trajectory. Aircraft 1 has two phases it its trajectory, which are shown in different colors. Similarly, Aircraft 2 has three phases in different colors and Aircraft 3 has four phases in different colors. Also it can be seen that, Phase I, which has all the three aircraft, is depicted in blue color; Phase II with two aircraft is depicted in magenta color; Phase III with one aircraft is depicted in green color, similarly Phase IV is red color; Phase V is cyan color and Phase VI is black color.

Aircraft 1 reaches the obstacle at (1, 3, 0.2); Aircraft 2 reaches the obstacle at (1, 3, 0.1); and Aircraft 3 reaches the obstacle at (-1, 3, 0.15). Hence it satisfies the path constraints of flying through the obstacle hole and avoiding collisions with the obstacle wall.

The x, y and z- axis plot of all the three aircraft is plotted in Figures 6-3 to 6-5. They reach the goal at 165, 170, 185 seconds, which is about 5 seconds added to the solutions obtained using the initial trajectory guess using the RRT algorithm. Hence the

software finds the shortest possible time taken by the aircraft to reach the runway while avoiding the obstacles and also taking into consideration the dynamics of the aircraft.

The flight path angle and the heading angle of all the aircraft is plotted in Figures 6-6 and 6-7. The smooth curve indicates the cost of minimizing the change in the flight path angle and heading angle is satisfied. Also the event constraints at the end of the Phases I, II, and III for the flight path angle and the heading angle are satisfied.

Validation of Results

Cost Function Validation

The Mayer cost of maximizing the time taken by the aircraft to reach the obstacle is satisfied and can be seen in the Figures 6-3 to 6-13. This can also be verified from Table 6-1, which displays the time taken by aircraft to reach the obstacle, t_{obs} and the final fixed time to land on the runway, t_f for all the three aircraft. The smooth curve of flight path angle and heading angle as seen in Figures 6-6 and 6-7 indicates that the Lagrange cost of minimizing the change in flight path angle and heading angle is also satisfied.

Table 6-1. Mayer cost validation

Aircraft Number	t_{obs} in seconds	t_f in seconds
1	79.77	155
2	94.96	170
3	109.96	185

Constraints Validation

The constraints formulated in Chapter 5 are validated from the plotted results.

Obstacle path constraints validation

The path constraint is the obstacle wall to be avoided in the trajectory. The values of each of these obstacles are found at each phase of the aircraft. These values satisfy

the obstacle path constraints specified in Chapter 5. This can also be seen in the three-dimensional trajectory of the three aircrafts shown in Figures 6-1 and 6-2, where the blue window is the obstacle wall with the hole in the center.

Event constraints validation

The aircraft satisfies the event constraints of flying only through the hole in the obstacle wall and being within the limit of [-1, 1] along x-axis, [0.1, 0.2] along z-axis, when it reaches the face of the obstacle wall. The obstacle wall face is placed at 3 nautical miles along the y-axis. Also the state value of heading path angle and flight path angle for Aircraft 1, 2 and 3 at the end of Phases I, II, and III respectively, is found to be 0 and -1.5708, as seen in Figures 5-5 to 5-8. This satisfies the event constraints of flight path angle and heading angle specified in Chapter 5.

Intersection constraints validation

The Figure 6-14 shows the minimum distance between the pairs of aircraft in each phase. The minimum safe separation distance value is taken as 0.25 nautical miles, but the bar graph shows that the actual minimum distance for the four intersection constraints specified in Chapter 5 is much more than the minimum safe separation distance of 0.25 nautical miles.

Bank angle constraints validation

The bank angles of the three aircraft are plotted in Figures 6-11 to 6-13. It is seen that the maximum and minimum bank angle values of the three aircraft is $45°$ and $-45°$. Also it can be noted that only the first aircraft reaches these maximum and minimum bounds. However the maximum and minimum bank angle value bound of the second and third aircraft is much smaller than that specified in Chapter 5.

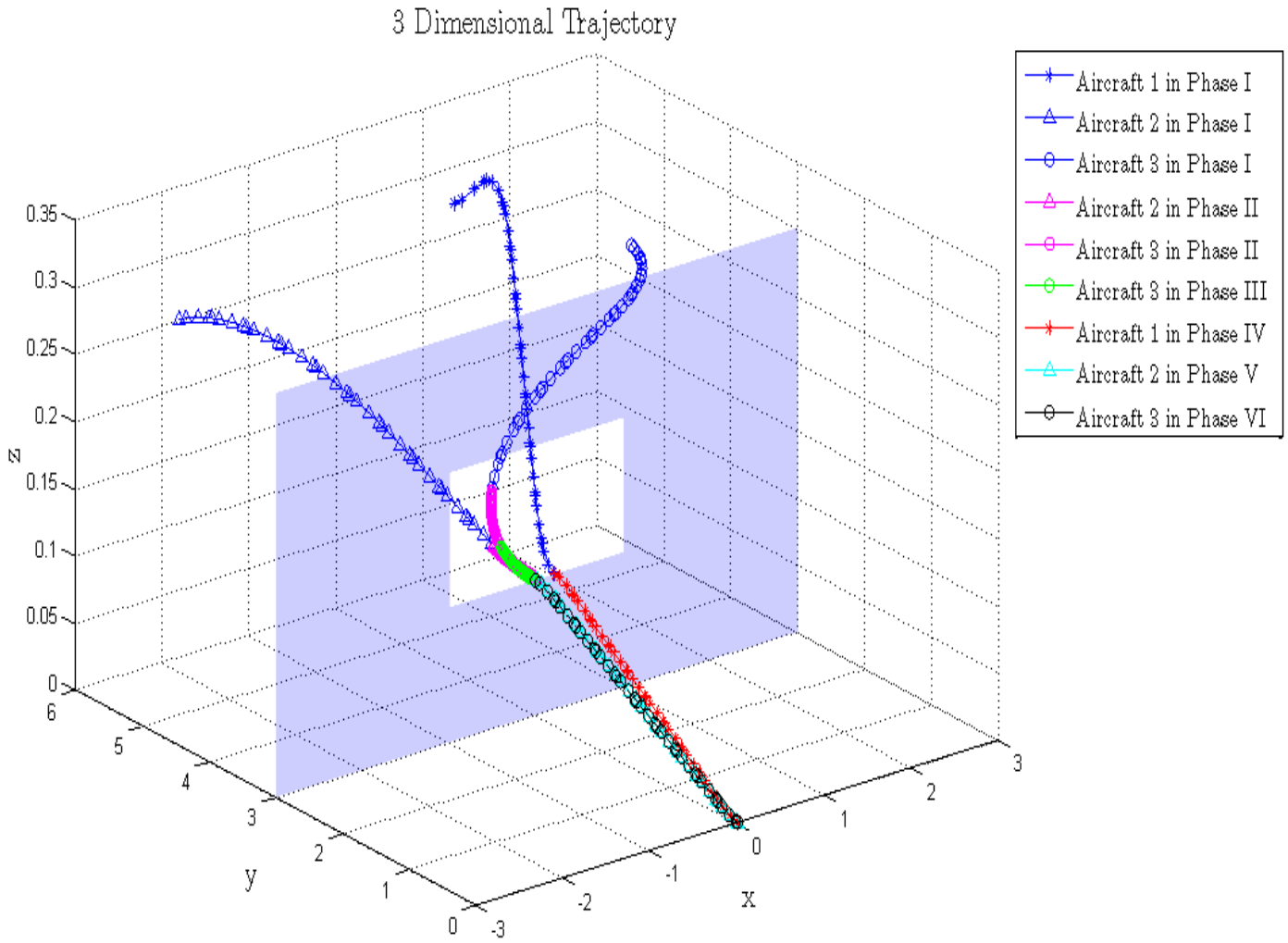

Figure 6-1. Three-dimensional trajectory with the obstacle wall obtained using GPOPS

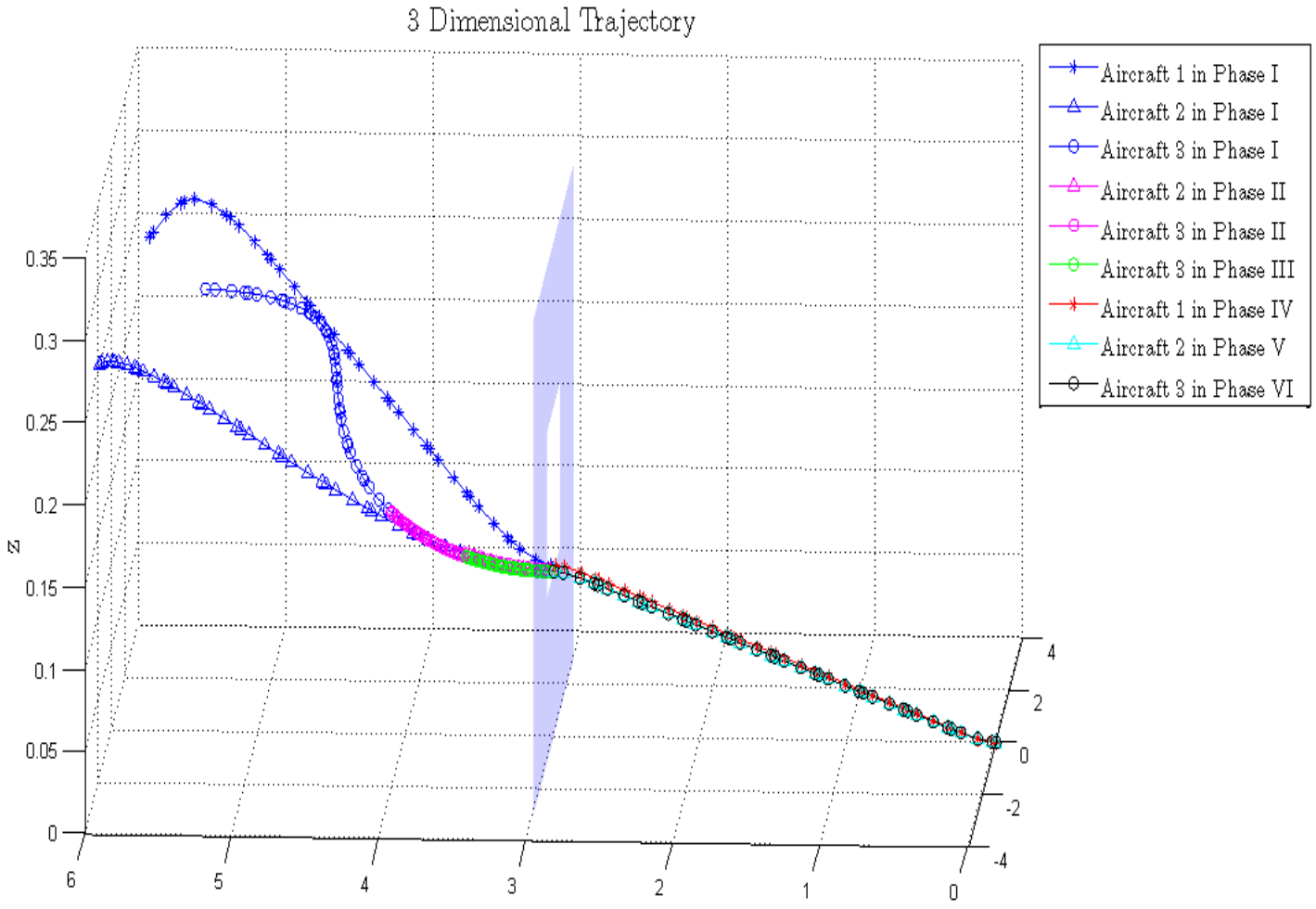

Figure 6-2. Y-Z axis view of the three dimensional trajectory

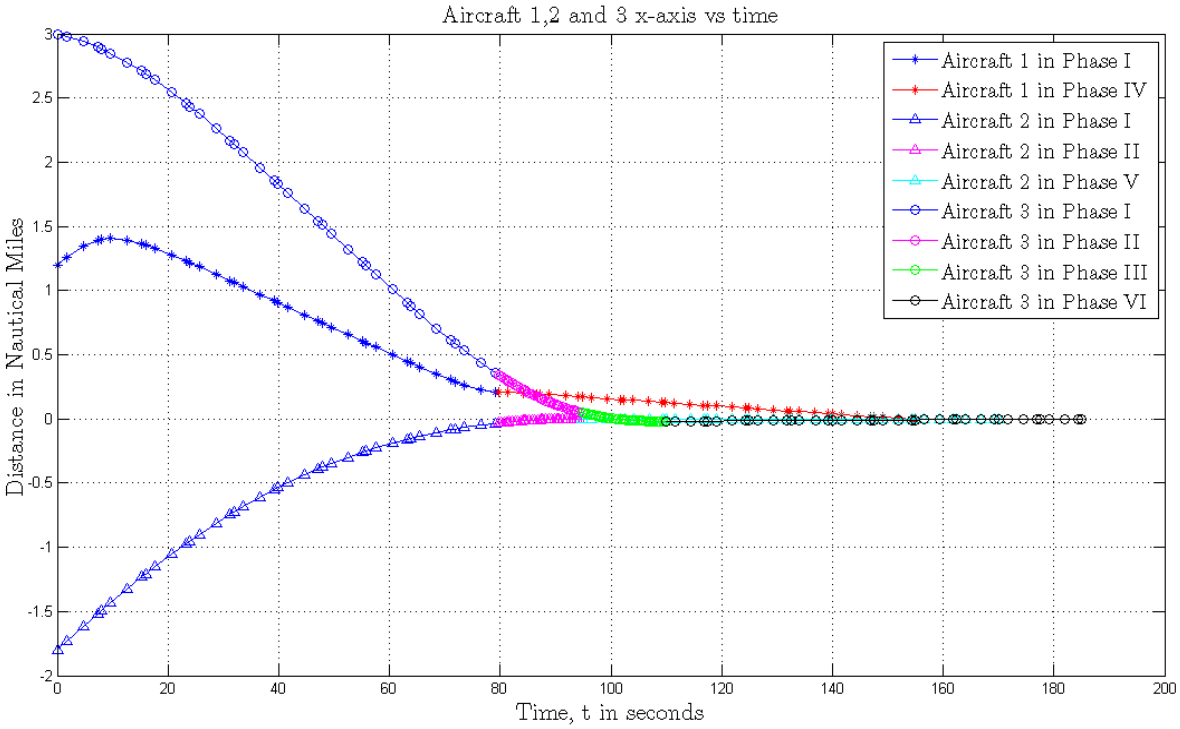

Figure 6-3. X-axis vs. Time plot of the three aircraft in different phases

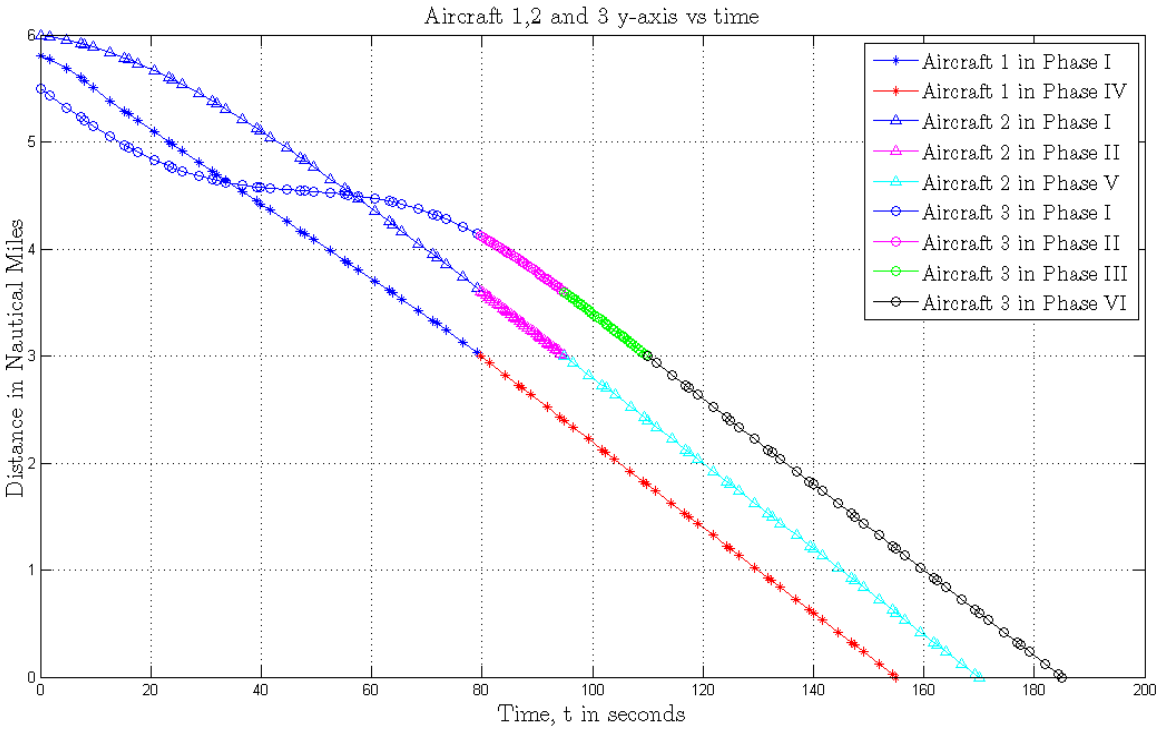

Figure 6-4. Y-axis vs. Time plot of the three aircraft in different phases

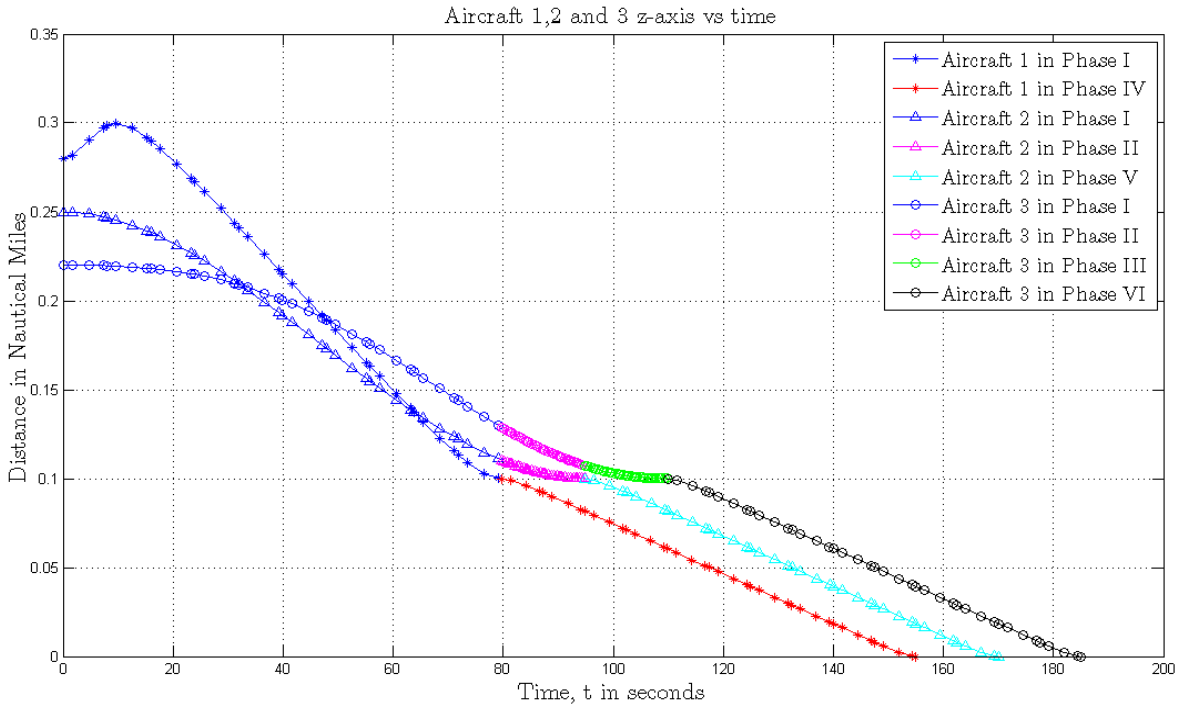

Figure 6-5. Z-axis vs. Time plot of the three aircraft in different phases

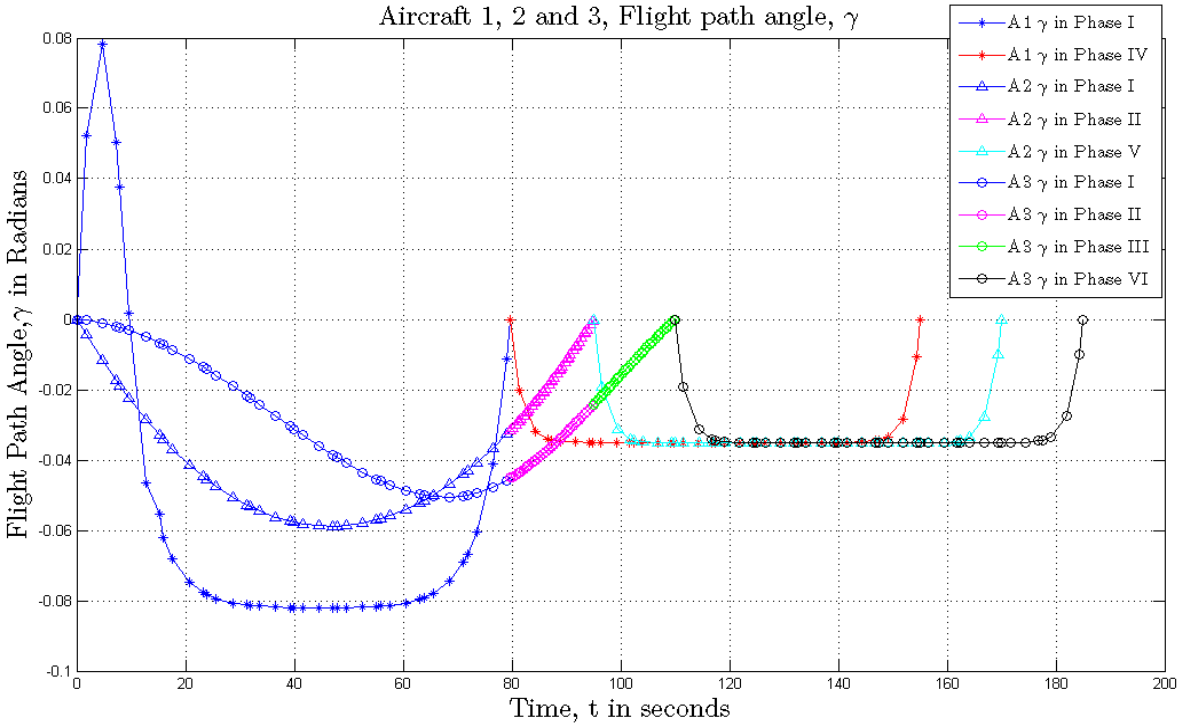

Figure 6-6. Flight path angle vs. Time plot of the three aircraft in different phases

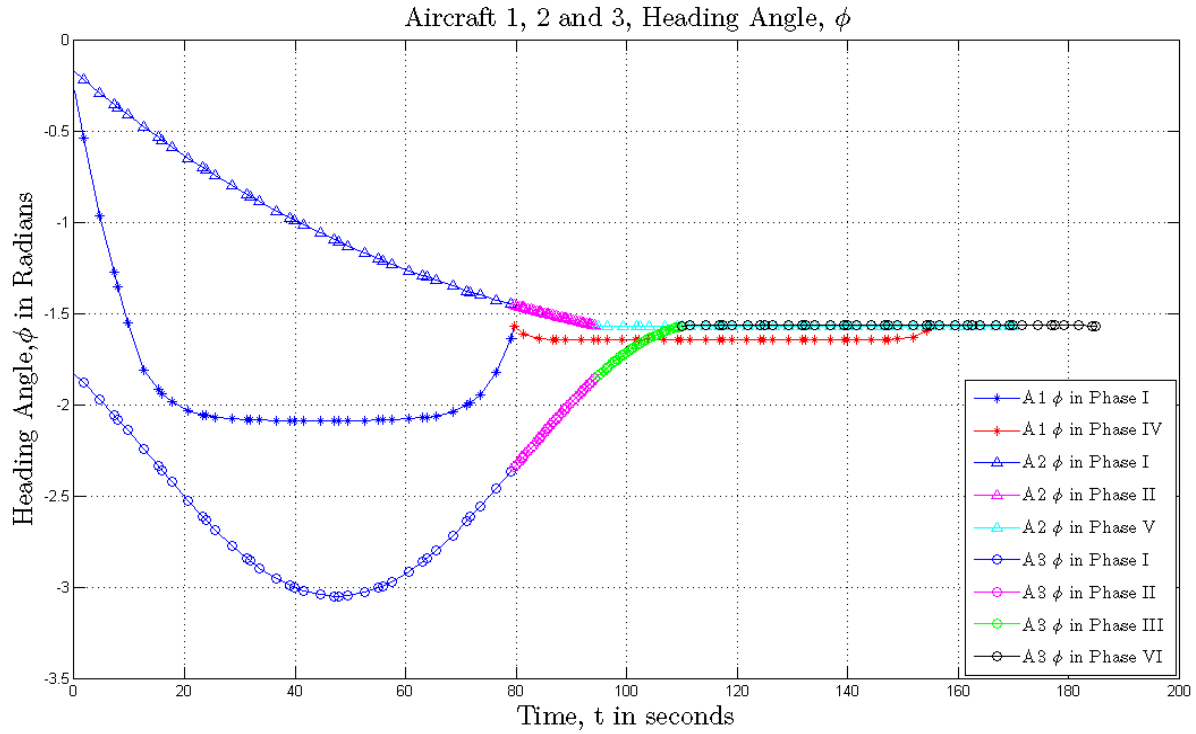

Figure 6-7. Heading angle vs. Time plot of the three aircraft in different phases

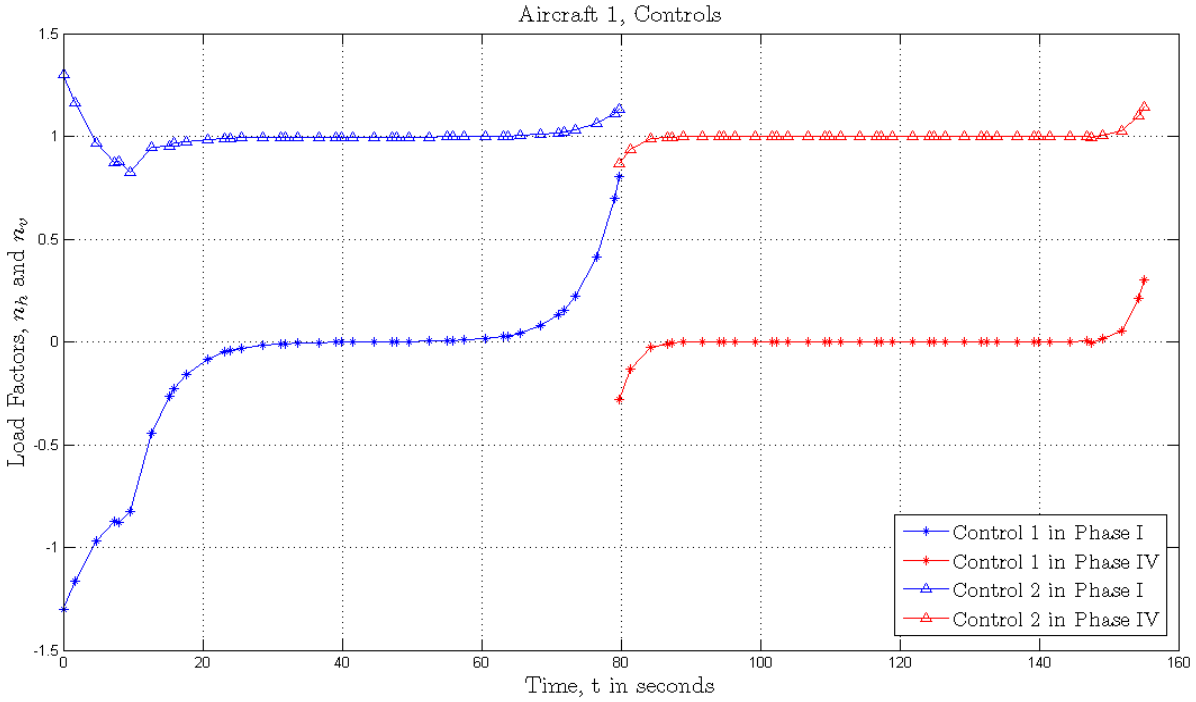

Figure 6-8. Load factor controls, n_{h1} and n_{v1} of Aircraft 1

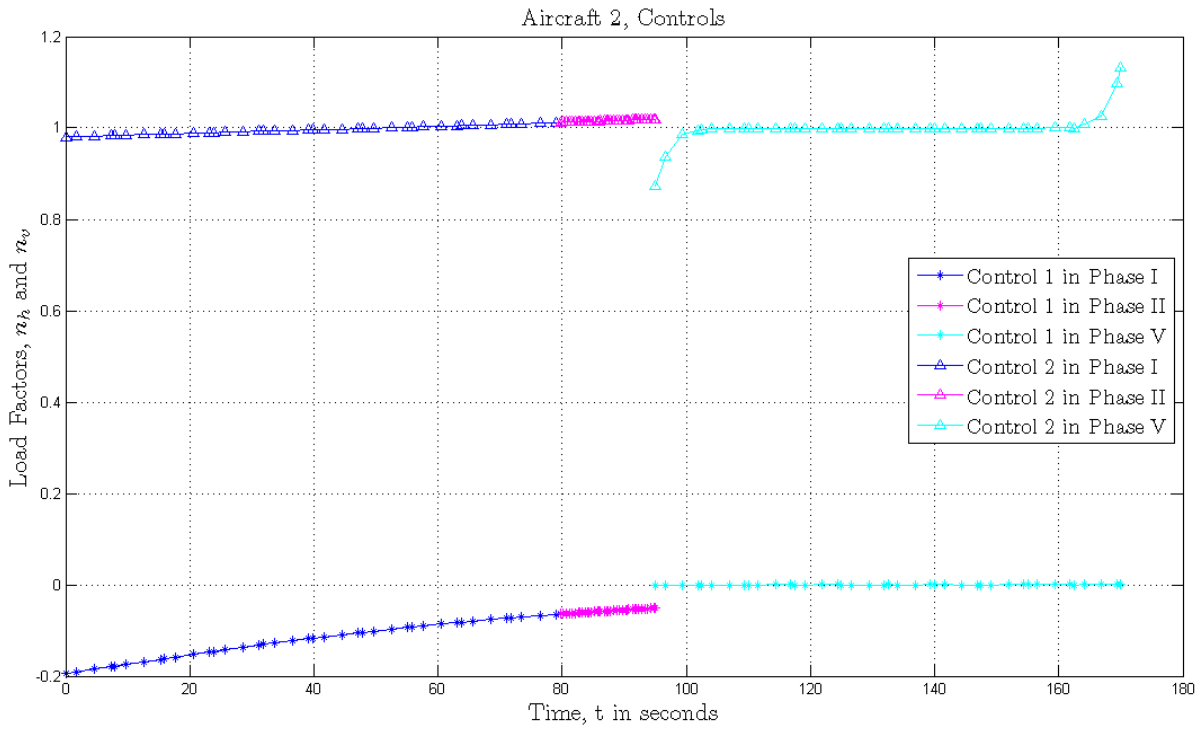

Figure 6-9. Load factor controls, n_{h2} *and* n_{v2} of Aircraft 2

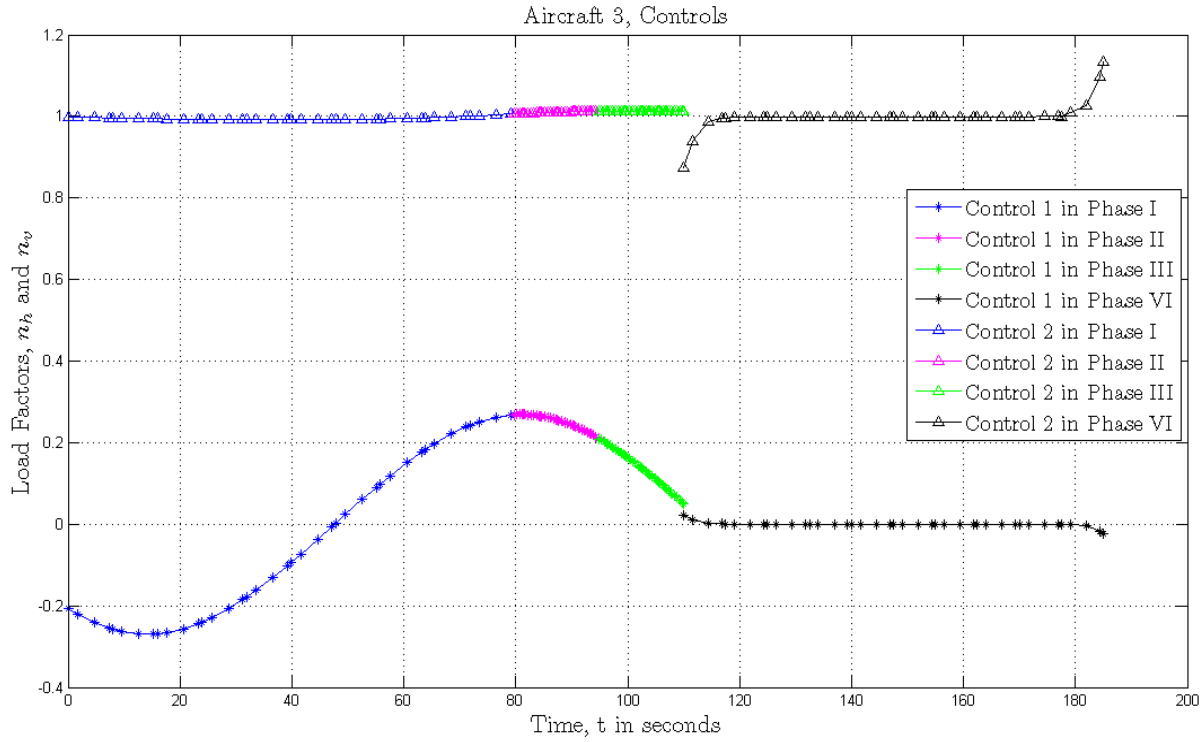

Figure 6-10. Load factor controls, n_{h3} *and* n_{v3} of Aircraft 3

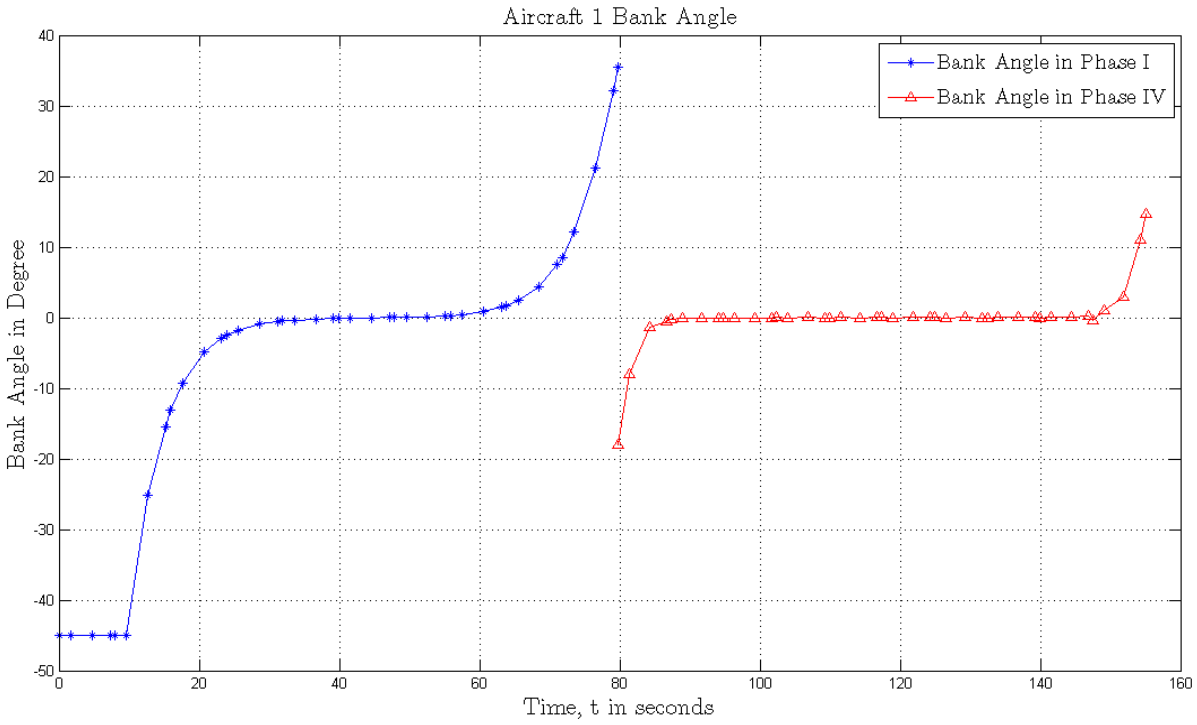

Figure 6-11. Bank angle of Aircraft 1 in different phases

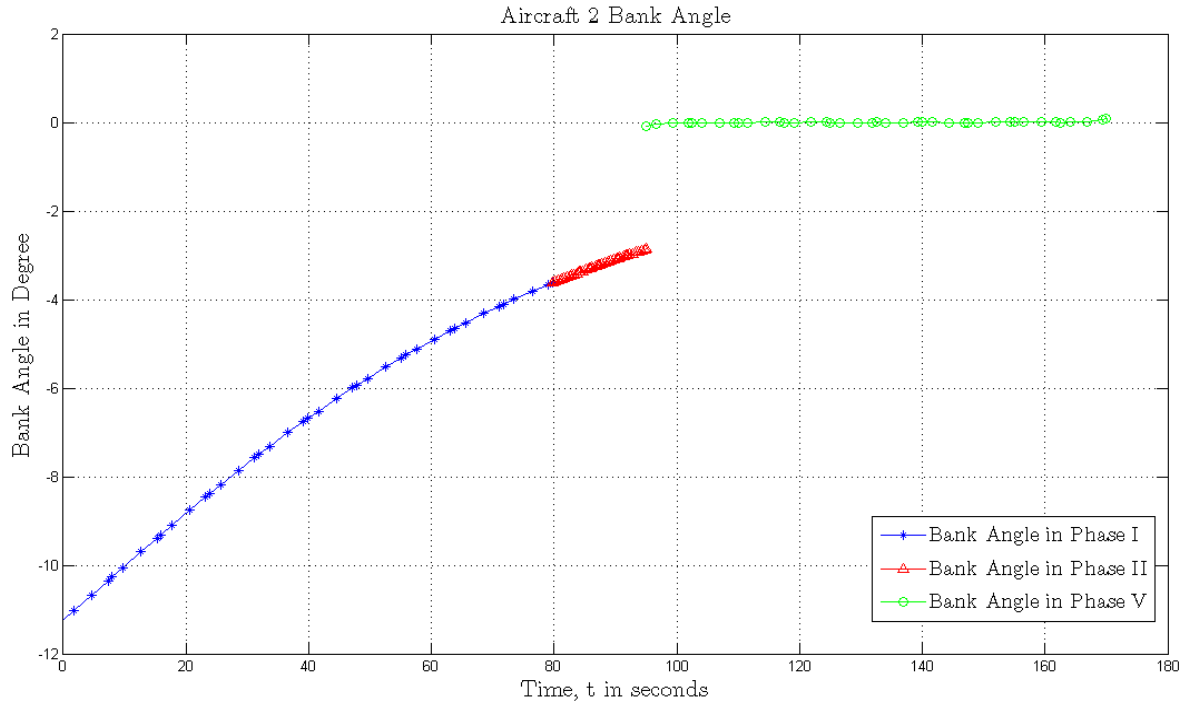

Figure 6-12. Bank angle of Aircraft 2 in different phases

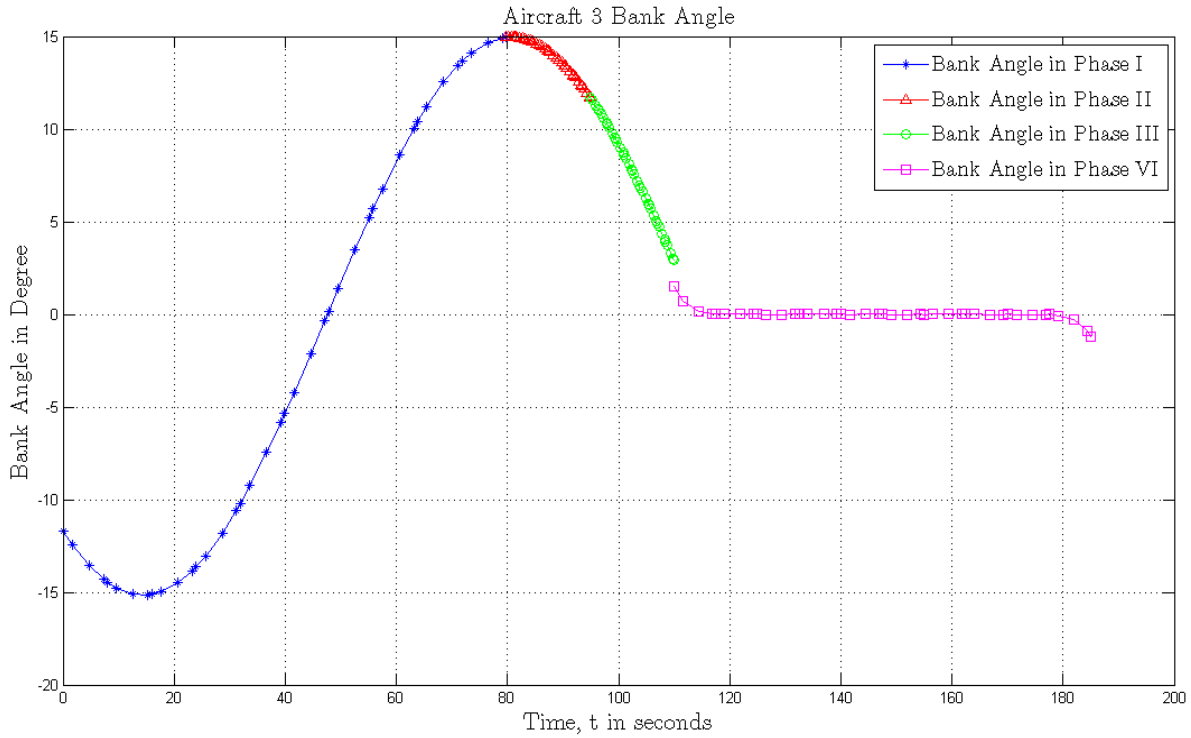

Figure 6-13. Bank angle of Aircraft 3 in different phases

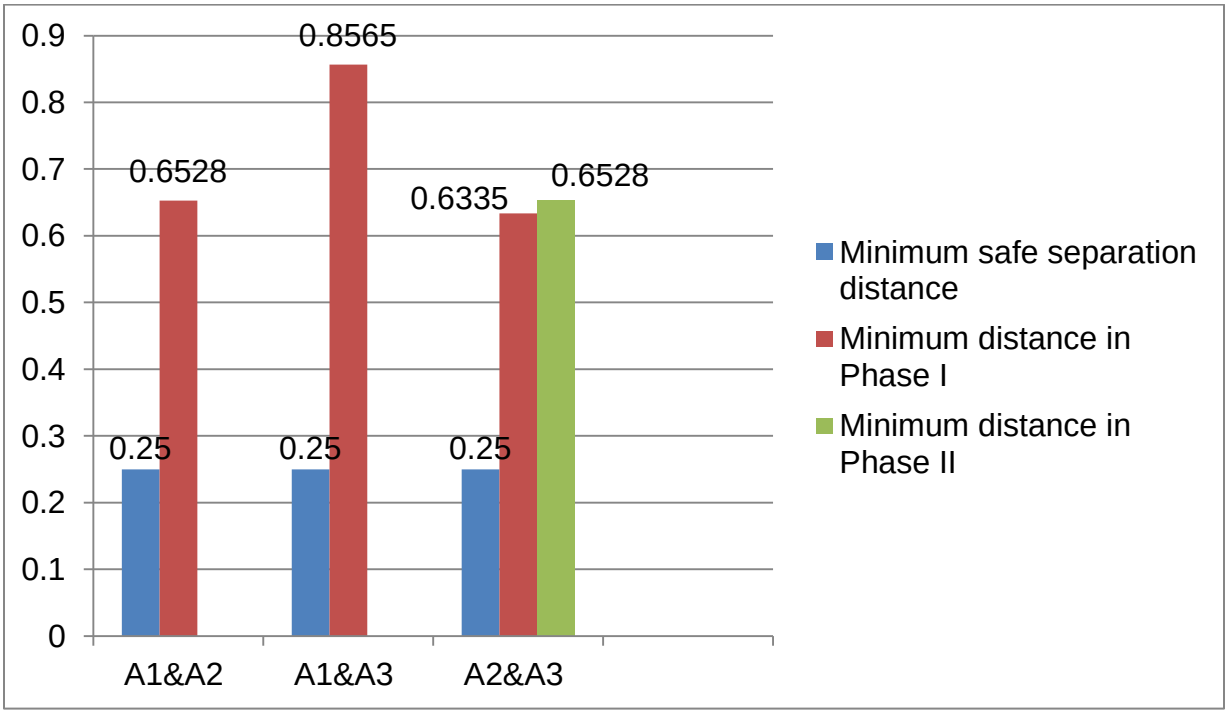

Figure 6-14. Intersection constraints validation- Minimum distance between aircraft in Phase I and II with A1- Aircraft 1, A2- Aircraft 2, and A3- Aircraft 3

CHAPTER 7
CONCLUSION

The work presented an approach to solve the multiple aircraft landing problem using a combination of RRT algorithm and pseudospectral methods. The RRT algorithm has been used to provide initial guess to the optimal control software, GPOPS, which helped in better convergence of the solution. An example problem to generate the landing trajectory of three aircraft has been solved. The problem and the obstacles has been formulated in Chapter 2. The initial trajectory generated using the proposed algorithm satisfied the minimum distance and time difference constraint of landing as seen in Chapter 4. The multiple-phase optimal control problem has been formulated in Chapter 5. From the results plotted in Chapter 6, it was found that the solution satisfied all the constraints in the problem and gave an optimal solution. It was also found that the aircraft final landing time was only 5 seconds more than that obtained using the proposed algorithm, used for initial guess. So the aircraft takes only 5 seconds more than the shortest possible time required by aircraft to reach the runway while avoiding the obstacle wall. The extra 5 seconds were used to consider the aircraft's dynamic constraints, event constraints, intersection constraints and bank angle constraints in the flight path. Thus the initial guess for the final landing time of the aircraft was found to be very close to the actual optimal solution to the problem. This was helpful as the problem has been formulated with a fixed landing time for the three aircraft and a fixed time difference between successive aircraft landings.

This research work can be used to assist the air traffic controllers in generating the optimal trajectories for aircraft which are ready to land and to improve the efficiency of the airports. Also it can help the ATC to avoid any obstacles such as buildings or no

fly zones while generating these trajectories. Future work on this research can decrease the computation time in generating these trajectories by optimizing the number of collocation points used to solve the optimal control problem. Also faster NLP solvers and better CPU processors can improve the computation time significantly. This can help in real time implementation of the work

LIST OF REFERENCES

[1] Lecchini, A., Glover, W., and Lygeros, J., "Air-Traffic Control in Approach Sectors: Simulation Examples and Optimisation," *Hybrid Systems: Computation and Control, Series-Lecture notes in Computer Science,* Vol. 3414, Jan. 2005, pp. 433-448.

[2] Kuchar, J.K., and Yang, L.C., "A Review of Conflict Detection and Resolution Modeling Methods," *IEEE Transactions on Intelligent Transportation Systems,* Vol. 1, No. 4, Dec. 2000, pp. 179-189.

[3] Frazzoli, E., Mao, Z.H., and Oh, J.H., "Resolution of Conflicts Involving Many Aircraft via Semidefinite Programming," *Journal of Guidance, Control, and Dynamics,* Vol. 24, No. 1, Jan-Feb. 2001, pp. 79-89.

[4] Pallottino, L., Feron, E.M., and Bicchi, A., "Conflict Resolution Problems for Air Traffic Management Systems Solved With Mixed Integer Programming," *IEEE Transactions on Intelligent Transportation Systems,* Vol. 3, No. 1, Mar. 2002, pp. 3-11.

[5] Hu, J., Prandini, M., and Sastry, S., "Optimal Coordinated Maneuvers for Three-Dimensional Aircraft Conflict Resolution," *Journal of Guidance, Control, and Dynamics,* Vol. 25, No. 5, Jan. 2002, pp. 888-899.

[6] Durand, N., Alliot, J., and Noailles, J., "Automatic Aircraft Conflict Resolution Using Genetic Algorithm," *Proceedings of the 1996 ACM symposium on Applied Computing*, 1996, pp. 289-298.

[7] Menon, P.K., Sweriduk, G.D., and Sridhar, B., "Optimal Strategies for Free-Flight Air Traffic Conflict Resolution," *Journal of Guidance, Control, and Dynamics,* Vol. 22, No. 2, Jan. 1999, pp. 202-212.

[8] Raghunathan, A.U., Gopal, V., and Subramanian, D., "Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft," *Journal of Guidance, Control, and Dynamics,* Vol. 27, No. 4, Jul-Aug. 2004, pp. 586-594.

[9] Dai, R., and Cochran, J.E., "Three-dimensional trajectory optimization in constrained airspace," Dissertation, Auburn Univ., 2007, pp. 1-162.

[10] Goerzen, C., Kong, Z., and Mettler, B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," *Journal of Intelligent and Robotic Systems,* Vol. 57, No. 1-4, Jan. 2010, pp. 65-100.

[11] Hurni, M.A., "An information-centric approach to autonomous trajectory planning utilizing optimal control techniques," Dissertation, Naval Postgraduate School, Sept. 2009, pp. 1-296.

[12] Lewis, L.R., Ross I.M., and Gong Q., "Pseudospectral motion planning techniques for autonomous obstacle avoidance," *Proceedings of the 46th IEEE Conference on Decision and Control,* 12–14 Dec. 2007, pp. 5997-6002.

[13] Lewis, L.R., and Ross, I.M., "A Pseudospectral Method for Real-Time Motion Planning and Obstacle Avoidance," *AVT-SCI Joint Symposium on Platform Innovations and System Integration for Unmanned Air, Land and Sea Vehicles*, Florence, Italy, May 2007, pp. 1-23.

[14] Lavalle, S.M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Computer Science Dept., Iowa State Univ., URL: http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf [cited 11 Nov. 2010].

[15] Redding, J., Amin, J., and Boskovic, J., "A Real-Time Obstacle Detection and Reactive Path Planning System for Autonomous Small-Scale Helicopters," *AIAA Guidance, Navigation and Control Conference and Exhibit,* Aug. 2007, pp. 1-22.

[16] Caves, A.D.J., "Human-automation collaborative RRT for UAV mission path planning," Thesis, Massachusetts Institute of Technology, May 2010, pp. 1-111.

[17] Aoudé, G.S., "Two-stage path planning approach for designing multiple spacecraft reconfiguration maneuvers and application to SPHERES onboard ISS," Thesis, Massachusetts Institute of Technology, Sep. 2007, pp. 1-149.

[18] Miele, A., "Optimal Trajectories of Aircraft and Spacecraft," Rice Univ., N91109686, Houston, TX. Sponsor: National Aeronautics and Space Administration, Washington, DC; United States, 1990.

[19] Miele, A., Wang, T., and Tzeng, C.Y., "Optimal abort landing trajectories in the presence of windshear," *Journal of Optimization Theory and Applications,* Vol. 55, No. 2, 1987, pp. 165-202.

[20] Nho, K., and Agarwal, R.K., "Automatic Landing System Design Using Fuzzy Logic," *Journal of Guidance, Control, and Dynamics,* Vol. 23, No. 2, Mar-Apr. 2000, pp. 298-310.

[21] Juang, J., and Chio, J., "Fuzzy modelling control for aircraft automatic landing system," *International Journal of Systems Science,* Vol. 36, No. 2, Jan. 2005, pp. 77-87.

[22] Chaturvedi, D.K., Chauhan, R., and Kalra, P.K., "Application of generalised neural network for aircraft landing control system," *Soft Computing,* Vol. 6, No. 6, 2002, pp. 441-448.

[23] Ruffier, F., and Franceschini, N., "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction," *Proceedings. ICRA '04. IEEE International Conference on Robotics and Automation,* Vol. 3, Apr. 2004, pp. 2339-2346.

[24] Patel, R.B., and Goulart, P.J., "Trajectory Generation for Aircraft Avoidance Maneuvers Using Online Optimization," *Journal of Guidance, Control, and Dynamics,* Vol. 34, No. 1, Jan-Feb. 2011, pp. 218-230.

[25] Benson, D.A., Huntington, G.T., and Thorvaldsen, T.P., "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics,* Vol. 29, No. 6, Nov-Dec. 2006, pp. 1435-1440.

[26] Garg, D., Patterson, M., and Francolin, C., "Direct trajectory optimization; costate estimation of finite-horizon; infinite-horizon optimal control problems using a Radau pseudospectral method," *Computational Optimization and Applications,* Oct. 2009, pp. 1-24.

[27] Garg, D., Patterson, M., and Hager, W.W., "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica,* Vol. 46, No. 11, Nov. 2010, pp. 1843-1851.

[28] Darby, C.L., Hager, W.W., and Rao, A.V., "An *hp*-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods,* Aug. 2010, pp. 1-41.

[29] Lewis, L.R., and Naval Postgraduate School (U.S.), "Rapid motion planning and autonomous obstacle avoidance for unmanned vehicles," Thesis, Naval Postgraduate School, 2006, pp. 1-161.

[30] Kuffner, J.J., and LaValle, S.M., "RRT-Connect: An Efficient Approach to Single-Query Path Planning," *Proceedings IEEE International Conference on Robotics and Automation,* Vol. 2, Apr. 2000, pp. 995-1001.

[31] Clifton, M., Paul, G., and Kwok, N., "Evaluating Performance of Multiple RRTs," *IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications,* Jun. 2008, pp. 564-569.

BIOGRAPHICAL SKETCH

Krithika Mohan was born in Delhi, India in 1987. She received her bachelor's degree in Electronics and Instrumentation Engineering from Velammal Engineering College (Anna University), Chennai, India in April 2009. She was awarded 14th rank among 1531 students, who graduated with bachelor's degree in Electronics and Instrumentation Engineering from Anna University affiliated institutions in April, 2009. She also graduated with a Master of Science degree in Mechanical Engineering from the Department of Mechanical and Aerospace Engineering at the University of Florida in May 2011. She completed her research work and master's thesis under the supervision of Dr. Anil V. Rao and was co-advised by Dr. Warren E. Dixon. Her interests lie in the field of path planning, trajectory optimization, guidance and control of nonlinear systems and nonlinear controls.