An Optimal Control Approach to Aircraft Conflict Avoidance

Sakirudeen Akinkunmi Abdulsalaam (abdulsalaam@aims.ac.za)
شاكردين اكيكومي بن عبدالسلام
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Montaz Ali
University of the Witwaterstrand, South Africa

22 May 2014
Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Aircraft conflict avoidance is an important issue arising in air traffic management. This work is based on Cellier et al. (2012) where a time dependent mathematical model that will keep a given separation in distance for a number of aircraft along their trajectories was developed. The main objective of this work is to reformulate the optimal control problem so that the constraint on the state variable is avoided. This was achieved by modifying the performance index of the control problem by writing the constraints as a penalty function in the objective function and applying direct technique of solving NLP to the resulting model.

Keywords: Optimal control, dynamical system, optimization, non-linear programming, conflict avoidance, separation distance, shooting method, penalty function, Air Traffic Management (ATM).

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Sakirudeen Akinkunmi Abdulsalaam, May 22, 2014

Contents

1. Introduction to Optimal Control

The meaning of the word *control* can be ambiguous. In the first case it may mean checking whether a product is up to standard or not while on the other hand it can mean controlling the behaviour of a system to get the desired output. In optimal control theory, the goal is to determine the input to a dynamical system which yields the optimal value of a specific cost function while satisfying any constraints imposed on the system (Rao, 2009).

Optimal control (OC) problem is that of choosing the best *path* among all the admissible paths for the system with respect to the given performance index (Liberzon, 2012). In this sense, the problem is *infinite–dimensional*, because the space of the paths is an infinite–dimensional function space. This problem is also a *dynamic* optimization problem, in the sense that it involves a dynamical system and time. It can be argued, according to Liberzon (2012), that optimality is a universal principle of life in the sense that many – if not most – processes in nature are governed by solutions to some optimization problems. OC problems are generally solved using numerical methods. The idea of using numerical methods to solve OC problem started around 1950 with the work of Bellman (Rao, 2009).

Liberzon (2012) lists the following as some examples of optimal control problems arising in applications:

- Send a rocket to the moon with minimal fuel consumption.
- Produce a given amount of chemical in minimal time and/or with minimal amount of catalyst used (or maximise the amount produced in a given time).
- Bring sales of a new product to a desired level while minimizing the amount of money spent on advertising campaign.
- Maximize throughput or accuracy of information transmission over a communication channel with a given bandwidth/capacity.

1.1 Basic Definitions

1.1.1 Dynamics. This is sometimes referred to as a *control system* (Liberzon, 2012). It is an ordinary differential equation (ODE) of the form

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad (t > 0), \\ \mathbf{x}(0) = x^0. \end{cases} \tag{1.1.1}$$

Here, we are given the initial point $x_0 \in \mathbb{R}$ and the function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$. The unknown is the curve $\mathbf{x} : [0, \infty) \to \mathbb{R}^n$, which we interpret as the dynamical evolution of the state of the system.

1.1.2 Controlled Dynamics. Suppose that $\mathbf{f}$ depends also upon some "control" parameters belonging to a set $A \subset \mathbb{R}^m$; so that $\mathbf{f} : \mathbb{R}^n \times A \to \mathbb{R}^n$. Then if we select some value $a \in A$ and consider the corresponding dynamics

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), a), \quad (t > 0), \\ \mathbf{x}(0) = x^0, \end{cases} \tag{1.1.2}$$

we obtain the evolution of the system for the constant value a. It is also possible that we change the value of the parameter as the system evolves. For instance, suppose we define the function $\alpha : [0, \infty) \to A$ this way:

$$\alpha(t) = \begin{cases} a_1 & 0 \le t \le t_1 \\ a_2 & t_1 < t \le t_2 \\ a_3 & t_2 < t \le t_3 \quad \text{etc.} \end{cases} \tag{1.1.3}$$

for times $0 < t_1 < t_2 < t_3 < \cdots$ and parameter values $a_1, a_2, a_3, \cdots \in A$; and we then solve the dynamical equation

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(\mathbf{t}), \alpha), \quad (t > 0), \\ \mathbf{x}(0) = x^0. \end{cases} \tag{1.1.4}$$

The resulting evolution is of the form shown in Figure 1.1. We call a function $\alpha : [0, \infty) \to A$ a *control* and corresponding to each control, we consider the ODE (1.1.4). We regard the trajectory $x(\cdot)$ as the corresponding *response* of the system.

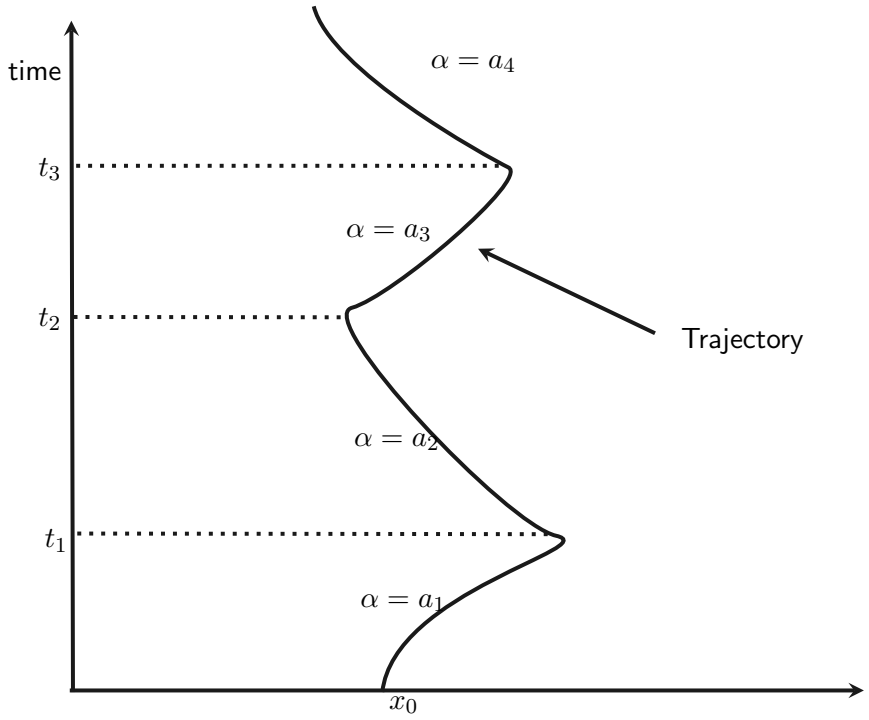


Figure 1.1: Control Dynamics

1.1.3 Payoffs. The payoff functional, J, is defined as

$$J[\alpha(\cdot)] := \int_{t_0}^{t_f} \mathcal{L}((\mathbf{x}(t), \alpha(t))dt + \Phi(\mathbf{x}(t_f)), \tag{1.1.5}$$

where $x(\cdot)$ solves (1.1.4) for the control $\alpha(\cdot)$. $\mathcal{L} : \mathbb{R}^n \times A \to \mathbb{R}$ and $\Phi : \mathbb{R}^n \to \mathbb{R}$ are given, and we call $\mathcal{L}$ the *running* payoff and Φ the *terminal* payoff. The initial and terminal time $t_0 \geq 0, t_f > 0$ are given as well.

1.1.4 End points and transversality condition. A transversality condition describes what must be satisfied at the end of a time horizon (Woodward, 2013). The nature of the transversality condition depends greatly on the statement of the problem. For example, it may be that the state variable must be equal to zero at the terminal time, t_f, i.e., $x(t_f) = 0$, or it might be that it must be less than some function of t, $x(t_f) \leq \phi(t_f)$.

Transverslity condition for various end points

- Free endpoints problems: in a free endpoint type problem, t_f is fixed and $x(t_f)$ can take any value.
- Fixed endpoint: in this case, there is no fixed endpoint for t but the ending state variable must have a certain value.
- Fixed terminal point: here, both $x(t_f)$ and t_f are fixed.

These various conditions are illustrated in Figure 1.2

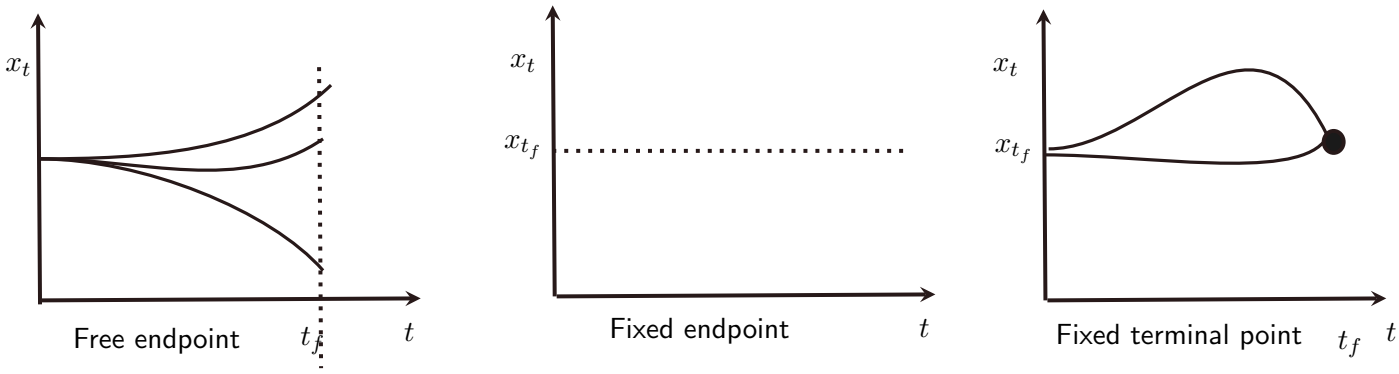


Figure 1.2: Travnsversality Conditions

2. Techniques of Optimal Control

2.1 Statement of Control Problem

In Rao (2009), optimal control problem is defined as the process of determining $\mathbf{x}(\mathbf{t}), \mathbf{u}(t), \mathbf{p}$ over the time interval $[t_0, t_f]$ which optimizes the cost function

$$\mathbf{J} = \Phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}) + \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}) dt, \tag{2.1.1}$$

subject to dynamic constraints (differential equation constraints)

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}), \quad \mathbf{x}(t_0) = \mathbf{x}^0, \tag{2.1.2}$$

the path constraints

$$\mathbf{C}_{min} \leq \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}) \leq \mathbf{C}_{max}, \tag{2.1.3}$$

and the boundary conditions

$$\phi_{min} \leq \phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}) \leq \phi_{max}. \tag{2.1.4}$$

Where $\mathbf{x}(\mathbf{t})$ is the optimal trajectory, $\mathbf{u}(\mathbf{t})$ is the optimal control and $\mathbf{p}$ is a set of parameters. The trajectory, control and parameters can be written as

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}; \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix}; \quad \mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_q \end{bmatrix}. \tag{2.1.5}$$

The ODE (2.1.2) describes the trajectory of the system, while the cost functional is a measure of the optimality of the trajectory. For a minimization problem, a smaller value of the cost functional J is desired while a higher value of J is better for a maximization problem.

2.2 Tools for Solving Optimal Control Problems

Only a few OC problems can be solved analytically. Solutions of OC problems rely mostly on numerical methods. To solve optimal control problems, there are three fundamental components to be considered namely: methods for solving differential equation and integrating functions, methods for solving non-linear algebraic equations and methods for solving non-linear optimization problem (Rao, 2009).

Numerical methods for solving ODE and integrating functions are required for numerical methods in optimal control. Indirect method involves combination of the numerical solution of ODE and non-linear equation while the direct methods consist of the numerical solution of ODE and NLP (Rao, 2009).

2.2.1 Numerical Solution of Differential Equations. Given the *initial-value problem* (IVP)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.2.1}$$

over the time interval $[t_i, t_{i+1}]$. Suppose at time t_i, $x(t_i) \equiv x_i$. We wish to obtain the solution at t_{i+1}, i.e., $x(t_{i+1}) \equiv x_{i+1}$. Integrating (2.2.1), we can write

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \dot{\mathbf{x}}(s)ds = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds. \tag{2.2.2}$$

In what follows, we consider two categories of numerical methods of solving ODEs, namely; time marching and collocation.

Time Marching

The time marching method involves obtaining the solution of a differential equation at a particular discretized time step t_{k+1} using the solution at the previous time step(s). It is of two categories, namely; (1) multi-step methods and (2) multi-stage methods.

1. *Multi-Step Methods*: To obtain the solution at the time step t_{k+1} in a multi step method, previous solutions at time $t_{k-j}, \ldots, t_k$, where j is the number of previous steps, are required.The most easy to implement multi-step method is the single-step method, whereby $j = 1$. Theta methods which is of the form

$$\mathbf{x}_{k+1} = \mathbf{x_k} + h_k(\theta \mathbf{f}_k + (1-\theta)\mathbf{f}_{k+1}), \tag{2.2.3}$$

where $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_{t_k}, \mathbf{u}(t_k), t_k)$ and $\theta \in [0, 1]$ are the most popular single step methods. If the value of θ in equation (2.2.3) equals to 1, then we have the forward Euler method, $\theta = \frac{1}{2}$ gives the Crank Nicolson method while $\theta = 0$ yields the backward Euler method. A more complicated multi-step method requires solutions from more than one previous time step. Examples of multi-step methods are Adams' methods. They are as stated below:

$$\text{Adams} - \text{Bashforth}: \quad \mathbf{x}_{k+1} = \mathbf{x}_k + h_k \left[\mathbf{f}_k + \frac{1}{2} \bigtriangledown \mathbf{f}_k + \frac{5}{12} \bigtriangledown^2 \mathbf{f}_k + \frac{3}{8} \bigtriangledown^3 \mathbf{f}_k + \frac{251}{720} \bigtriangledown^4 \mathbf{f}_k + \cdots \right], \tag{2.2.4}$$

$$\text{Adams} - \text{Moulton}: \quad \mathbf{x}_{k+1} = \mathbf{x}_k + h_k \left[\mathbf{f}_k - \frac{1}{2} \bigtriangledown \mathbf{f}_k - \frac{1}{12} \bigtriangledown^2 \mathbf{f}_k - \frac{1}{24} \bigtriangledown^3 \mathbf{f}_k - \frac{19}{720} \bigtriangledown^4 \mathbf{f}_k + \cdots \right], \tag{2.2.5}$$

and $\bigtriangledown$ is the backward difference operator,

$$\bigtriangledown \mathbf{f} = \mathbf{f}_k - \mathbf{f}_{k-1}. \tag{2.2.6}$$

An implicit method, e.g. backward Euler and Crank Nicolson, has the value $\mathbf{x}(t_{k+1})$ appearing implicitly on the right hand side of its equation whereas explicit method does not. An example of explicit method is the forward Euler method.

2. *Multi-stage Methods*:Assuming the time interval $[t_i, t_{i+1}]$ is divided into K subintervals $[\tau_j, \tau_{j+1}]$ where

$$\tau_j = t_i + h_1\alpha_j, \quad (j = 1, \ldots, K), \quad h_i = t_{i+1} - t_i, \tag{2.2.7}$$

and $0 \leq \alpha_j \leq 1, (j = 1, \ldots, K)$. τ_j is known as *stage*. We can approximate the integral t_i to t_{i+1} by using a numerical quadrature

$$\int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}(\mathbf{x}_j, \tau_j), \tag{2.2.8}$$

where $\mathbf{x}_j \equiv \mathbf{x}(\tau_j)$. The values of the integral in-between the two time step can be obtained using the solution at time step t_i and the approximation of the integral within the interval using

$$\mathbf{x}(\tau_j) - \mathbf{x}(t_i) = \int_{t_i}^{\tau_j} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{l=1}^{K} \gamma_{jl}\mathbf{f}(\mathbf{x}_l, \tau_l). \tag{2.2.9}$$

If we combine equation (2.2.8) and (2.2.9), we obtain the family of K-stage *Runge - Kutta* methods

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds &\approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}_{ij}, \\ \mathbf{f}_{ij} &= \mathbf{f}\left(\mathbf{x}_i + h_i \sum_{l=1}^{K} \gamma_{jl}\mathbf{f}_{il}, t_i + h_i\alpha_j\right). \end{aligned} \tag{2.2.10}$$

A Runge-Kutta scheme, in summary, involves using the solution at the beginning of a time step with the approximations within the time step to obtain the solution at the end of the time step. The *Butcher array*, shown in Figure 2.1, is a concise way of presenting The RK methods:

$$\begin{array}{c|ccc} \alpha_1 & \gamma_{11} & \cdots & \gamma_{1K} \\ \vdots & \vdots & & \vdots \\ \alpha_K & \gamma_{K1} & \cdots & \gamma_{KK} \\ \hline & \beta_1 & \cdots & \beta_K \end{array}$$

Figure 2.1: The Butcher Array

Runge-Kutta method is an example of Euler method of first order. The *classical Runge-Kutta method*, (see equation (2.2.11)) is the most popular of the RK methods.

$$\begin{aligned} k_1 &= h_i f(x_i, t_i), \\ k_2 &= h_i f(x_i + \frac{h_i}{2}k_1, t_i + \frac{h_i}{2}), \\ k_3 &= h_i f(x_i + \frac{h_i}{2}k_2, t_i + \frac{h_i}{2}), \\ k_4 &= h_i f(x_i + h_i k_3, t_i + h_i), \\ x_{i+1} &= x_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{2.2.11}$$

Hermite-Simpson method of equation (2.2.12) is another well-known Runge-Kutta scheme.

$$\begin{aligned} \hat{x} &= \frac{1}{2}(x_i + x_{i+1}) + \frac{h_i}{8}(f_i - f_{i+1}), \\ \hat{f} &= f(\hat{x}, t_i + \frac{h_i}{2}), \\ x_{i+1} &= x_i + \frac{h_i}{6}(f_i + 4\hat{f} + f_{i+1}). \end{aligned} \tag{2.2.12}$$

The classical Runge-Kutta and Hermite simpson scheme are fourth and third order scheme respectively.

Collocation

Collocation is another means of solving differential equations. Consider the subinterval $[t_i, t_{i+1}]$, we approximate the state using a $K^{th} - degree$ piecewise polynomial

$$\mathbf{X}(t) \approx \sum_{k=0}^{K} \mathbf{a}_i (t - t_i)^k, \quad t \in [t_i, t_{i+1}]. \tag{2.2.13}$$

Furthermore, we suppose that the coefficients $(a_0, \ldots, a_K)$ of the piecewise polynomial are chosen to match-up the value of the function at the beginning of each step, i.e.,

$$\mathbf{X}(t_i) = \mathbf{x}(t_i). \tag{2.2.14}$$

And finally, we suppose that we match-up the derivative of the state at the points defined in equation (2.2.8). This means that we have

$$\dot{\mathbf{X}}(\tau_j) = \mathbf{f}(\mathbf{x}(\tau_j), \tau_j), (j = 1, \ldots, K). \tag{2.2.15}$$

Equation (2.2.15) is called a *collocation condition*. This is because the approximate value of the derivative is set equal to the right-hand side of the ODE evaluated at each of the intermediate points $(\tau_1, \ldots, \tau_K)$. Collocation methods are of three categories, namely; Gauss methods, Radau methods and Lobatto methods (Rao, 2009).

2.2.2 Non-linear Optimization. One important tool required to solve OC problems is the ability to solve *non-linear optimization* otherwise known as *non-linear programming problems* (NLP) (Rao, 2009) or parameter optimization (Betts, 2001). Non-linear optimization is an optimization problem whose objective function or constraints or both is/are neither linear nor convex (Boyd and Vandenberghe, 2004). The NLP problem requires finding the value of finite number of variables that optimize a cost function without violating the set constraints on the cost function (Betts, 2001). A problem with about ten variables which may look simple oversight can be challenging while trying to solve it, while large scale problems may be intractable. Methods for solving NLP takes several approaches, each with pros and cons. Some special cases of the NLP problem are *linear programming (LP), quadratic programming (QP)* and *least squares* problems.

NLP, according to Rao (2009), has the general form: determine the values of decision variables $\mathbf{z} \in \mathbb{R}^n$ which optimize (minimize or maximize) the cost function

$$f(\mathbf{z}), \tag{2.2.16}$$

subject to

$$\mathbf{g}(\mathbf{z}) = \mathbf{0}, \tag{2.2.17}$$

$$\mathbf{h}(\mathbf{z}) \leq \mathbf{0}, \tag{2.2.18}$$

where $\mathbf{g}(\mathbf{z}) \in \mathbb{R}^m$ and $\mathbf{h}(\mathbf{z}) \in \mathbb{R}^p$. The first order optimality conditions of the NLP (2.2.16), called *Karush-Kuhn-Tucker conditions* (KKT), are given as

$$g_i(\mathbf{z}) = 0, (i = 1, \ldots, m); \quad h_i(\mathbf{z}) \leq 0, (i = 1, \ldots, p); \quad \nu_i \geq 0, (i = 1, \ldots, p),$$
$$\nu_i h_i(\mathbf{z}) = 0, (i = 1, \ldots, p), \tag{2.2.19}$$

$$\bigtriangledown f(\mathbf{z}) + \sum_{i=1}^{m} \lambda_i \bigtriangledown g_i(\mathbf{z}) + \sum_{i=1}^{p} \nu_i \bigtriangledown h_i(\mathbf{z}) = \mathbf{0}. \tag{2.2.20}$$

The NLP may either be *dense* or *sparse*. There are two categories of solution approaches to NLP problems, namely: *gradient-based methods* and *heuristic methods*.

Gradient Methods. In a gradient based method, we are required to guess the initial value of the decision variables, $\mathbf{z}$. $\mathbf{p}_k$ (search direction) and α_k (step length) are determined at each k^{th} iterative step. The search direction provides a direction in $\mathbb{R}^n$ along which to change the current value $\mathbf{z}_k$ and the step length provides the size of the change to $\mathbf{z}_k$. The update from $\mathbf{z}_k$ to $\mathbf{z}_{k+1}$ has the form

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{p}_k. \tag{2.2.21}$$

The common way of getting search direction is by using the negative gradient $\triangle \mathbf{z} = - \bigtriangledown f(\mathbf{z})$ (Boyd and Vandenberghe, 2004). We thereby get the algorithm known as the gradient algorithm or gradient descent method. The most popular gradient based NLP solution methods are *sequential quadratic programming* (SQP) and *interior point method* (IP) (Rao, 2009). For constrained problem, the search direction in an SQP method is the solution of the quadratic programing (QP) problem

$$\begin{aligned} &\min_{\mathbf{p}} \frac{1}{2}\mathbf{p}^T \mathbf{W} \mathbf{p} + \bigtriangledown f^T(\mathbf{z}_k)\mathbf{p} \\ \text{s.t} \quad &\bigtriangledown g_i^T(\mathbf{z}_k)\mathbf{p}(\mathbf{z}_k) = \mathbf{0}, \quad (i \in \mathbb{G}) \\ &\bigtriangledown h_i^T(\mathbf{z}_k)\mathbf{p} \leq \mathbf{0}, \quad (i \in \mathbb{H}) \end{aligned} \tag{2.2.22}$$

where $\mathbb{G}$, $\mathbb{H}$ are the active, inactive constraint sets respectively, and $\mathbf{W}_k$ is an approximation of the Hessian of the Lagrangian , $\mathcal{L}$, where $\mathcal{L}$ is defined as

$$\mathcal{L}(\mathbf{z}, \lambda) = f(\mathbf{z}) - {\lambda_\mathbf{1}}^T \mathbf{g}(\mathbf{z}) - {\lambda_\mathbf{2}}^T \mathbf{h}(\mathbf{z}). \tag{2.2.23}$$

The common way of obtaining the matrix $\mathbf{W}_k$ is by using a *quasi-Newton* approximation in the form of an update method. One of the well-known update methods is the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) (Rao, 2009).

Heuristic Optimization Methods. There is a huge difference between a gradient and heuristic optimization method. A gradient method is a local method in the sense that, after convergence, a local optimal solution is obtained. A heuristic method, on the other hand, is a global technique. The principal idea of heuristic optimization is that it searches for an optimal solution in a stochastic way. One of the well-known heuristic methods is the *genetic algorithm* which is an evolutionary approach. A genetic algorithm imitates the evolutionary process on computer. The basis of all genetic algorithms are: encoding, fitness, selection, crossover and mutation. Another example of heuristic method is the *Simulated annealing* whereby every point in the search-space is likened to some physical process, and the fitness function represents the internal energy of the system. Minimizing the internal energy of the system is the goal of the process.

2.2.3 Systems of Nonlinear Algebraic Equations. Solving a system of nonlinear algebraic equations and *root finding* are identically the same. If all the algebraic equations can be written as equalities, then the problem to be solved is of the form

$$\mathbf{g}(\mathbf{z}) = \mathbf{0}. \tag{2.2.24}$$

Generally, methods for root finding can only handle functions with scalar variables (e.g bracketing and bisection, the secant method). Therefore, they are not so powerful. *Newton's method* is the best known

multi-dimensional root finding method. It involves guessing the initial vector $\mathbf{z}$. The iterations proceed with the formula

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \left[\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right]^{-1}_{\mathbf{z}_k} \mathbf{g}(\mathbf{z}_k). \tag{2.2.25}$$

Newton's method converges if the initial guess is in the neighbourhood of a root. However, most systems of non-linear equations have more than one solution. Due to this fact, the root obtained will in most cases be the one close to the initial guess which may not be the desired solution. Furthermore, from equation (2.2.25), it can be noted that Newton method will fail if the Jacobian of $\mathbf{f}$ $\left(\text{i.e.,} \frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right)$ is singular. To prevent the iteration from being undefined, $\frac{\partial \mathbf{g}}{\partial \mathbf{z}}$ is usually replaced with a quasi-Newton approximation (e.g a BFGS-type approximation to the Jacobian).

2.3 Methods for Solving Optimal Control Problems

As noted in the introduction, apart from very simple problems, optimal control problems must be solved numerically. In general, there are three basic approaches to solve continuous time optimal control problems. They are: (a) state space, (b) indirect, and (c) direct approaches. Indirect methods optimize in an infinite dimensional function space, while direct methods transform the problem to a finite-dimensional space first before the optimization takes place (Sager, 2006). In what follows, we describe each of these methods in detail.

2.3.1 State-space Approach. This approach uses the principle of optimality that states that each sub-arc of an optimal trajectory must be optimal. This is the basis of dynamic programming in discrete time. Analogous to it in the contnuous case is the *Hamilton-Jacobi-Bellman* (HJB) equation. It is a partial differential equation (PDE) in the state space. There are numerical methods to compute the approximate solution but the state space approach suffers from Bellman's "curse of dimensionality". Therefore it is used for problems with small state dimension (Sager, 2006).

2.3.2 Theorem (Optimality Condition)**.** *Let $F(x,u,t)$ and $f(x,u)$ be continuously differentiable functions of each of their arguments. Let $u^* \in \mathcal{U}[t_0,t_f]$ be an optimal control for the functional*

$$I(u) = \int_{t_0}^{t_f} F(x(t),u(t),t)dt, \tag{2.3.1}$$

subject to

$$\dot{x}(t) = f(x(t),u(t)), \quad t \in [t_0,t_f], x(t_0) = x_0.$$

Let x^* be the corresponding optimal state. If $t_m \in [t_0,t_f)$, then the restriction of u^* to $[t_m,t_f]$ is an optimal control for the functional

$$\tilde{I}(u) = \int_{t_0}^{t_f} F(x(t),u(t),t)dt, \tag{2.3.2}$$

subject to

$$\dot{x}(t) = f(x(t),u(t)), \quad t \in [t_m,t_f], x(t_m) = x^*(t_m).$$

Hamilton-Jacobi-Bellman (HJB) equation. "In place of determining the optimal sequence of decisions from the fixed state of the system, we wish to determine the optimal decision to be made at

any state of the system. Only if we know the latter, do we understand the intrinsic structure of the solution." These were the words of Richard Bellman, in his book "Dynamic Programming" (1957). The HJB equation is crucial in optimal control theory. The value of a cost function after solving the HJB equation is the minimum cost for a given dynamical system with the associated cost function.

2.3.3 Theorem (Hamilton-Jacobi-Bellman Equation)**.** *If the cost function* (2.1.1) *has continuous first derivative, i.e.,* $J(\cdot) \in C'$*, then the first-order non-linear partial differential equation obeyed by the optimal* $J(\cdot)$ *is given by*

$$-\frac{\partial \mathbf{J}}{\partial t}(\mathbf{x}, t) = \min_{u} \mathcal{L}(\mathbf{x}, \mathbf{u}, t; \mathbf{p}) + \bigtriangledown_x \mathbf{J}(x, t)^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t; \mathbf{p}) \tag{2.3.3}$$

where $t \in [t_0, t_f]$ *is any base period,* $\mathbf{x}$ *is any admissible trajectory, and* $\mathbf{p}$ *is a vector of constant parameters and* $\mathcal{L}$ *is the running pay-off. The HJB equation is a necessary condition for optimality if solved locally but when it is solved over the entire state space, it constitutes the necessary and sufficient condition for an optimal solution. The HJB method can be generalized to stochastic systems.*

2.3.4 Indirect Approach. The classical approach to solving optimal control problems is based on *Pontryagin's maximum principle* (PMP) (Sager, 2006). The approach is an elegant and compact way to characterize and compute solutions of optimal control problems. Its origin dates back to the calculus of variations and the classical work by Euler and Lagrange. However, its full generality was only developed in 1950s and 1960s, starting with the work of Pontryagin and co-workers (Diehl, 2011). PMP describes the necessary condition for optimality. The necessary conditions for optimality is used to transform the optimization problem to a multi-point boundary value problem that is solved with suitable numerical ODE solver, e.g., Runge-Kutta. An optimal solution typically consists of several arcs. On each arc, we have constraint-seeking or compromise-seeking controls (Sager, 2006). Using the optimality conditions in order to eliminate the controls from the problem and then numerically solving the BVP is called the *indirect approach* to optimal control (Diehl, 2011). It was widely used when the Sputnik and Apollo space missions were planned and executed. The method is still very popular in aerospace applications. The following are the main drawbacks of this approach according to Diehl (2011):

- It must be possible to eliminate the controls from the problem by algebraic manipulations. This is not always straight forward and may not even be possible in some cases.
- The optimal control might be a discontinuous function of x and λ (co-state variable), such that the BVP suffers from non-smooth differential equation.
- The differential equation might become highly nonlinear and unstable and not suitable for a forward simulation.

These problems can be addressed in some cases. An advantage of the indirect method is that it offers an exact characterization of the solution of the optimal control problem in continuous time.

2.3.5 Definition (Hamiltonian)**.** The *Hamiltonian* function H of the OC problem (2.1.1) is defined as

$$H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$$

with

$$H(t, \lambda_0, \lambda, x, u) = \lambda_0 \mathcal{L}(t, x, u) + \lambda(f(t, x, u). \tag{2.3.4}$$

where λ_0 is a constant and λ is the adjoint variable.

2.3.6 Theorem (Pontryagin maximum principle)**.** *Let* (x_*, u_*) *be a controlled trajectory defined over the interval* $[t_0, t_f]$ *with the control* u_* *piecewise continuous. If* (x_*, u_*) *is optimal, then there exists a co-vector* $\lambda : [t_0, t_f] \to (\mathbb{R}^n)^*$*, the so called adjoint variable, such that the following conditions are satisfied:*

I Nontriviality of the multipliers: $\lambda(t) \neq 0, \quad \forall t \in [t_0, t_f]$.

II The state equation: the solution to the differential equation

$$\dot{x}^*(t) = \bigtriangledown_\lambda H(x^*(t), u^*(t), \lambda(t)) \tag{2.3.5}$$

III Adjoint equation: the adjoint variable λ *is a solution to the time-varying linear differential equation*

$$\dot{\lambda}(t) = - \bigtriangledown_x H(x^*(t), u^*(t), \lambda(t)) \tag{2.3.6}$$

IV Minimum condition: everywhere in $[t_0, t_f]$ *we have that*

$$u^* = \arg\min_u H(x^*(t), \lambda(t), u(t)). \tag{2.3.7}$$

V Transversality condition: at the endpoint of the controlled trajectories,

$$\lambda(t_f) = \bigtriangledown \Phi(t_f).$$

The main advantage of indirect methods is the high accuracy of the obtained solution, as the infinite–dimensional problem has been solved (Sager, 2006).

2.3.7 Direct approach. Direct methods to continuous optimal control involves transformation of the problem from an infinite domain to a finite domain through parametrization of the control $u(t)$. The original problem is approximated by a finite dimensional NLP. This can then be solved by any of the numerical NLP solution methods. Due to this, the approach is often characterized as *"first discretize, then optimize"*. The direct approach links with all optimization methods developed in the continuous optimization. The successful direct methods parametrize the problem such that the resulting NLP has discrete time optimal control structure and thereby making the technique of NLP applicable. The following are some of the well known direct methods.

Direct Single Shooting. For a general Boundary Value Problem (BVP) for a second order ODE, the simple shooting method is stated as follows: Let

$$\begin{aligned} &x''(t) = f(t, x(t), x'(t)), \quad t \in [a, b] \\ &x(a) = \alpha, \quad x(b) = \beta \end{aligned} \tag{2.3.8}$$

be the BVP to be solved and let $x(t, s)$ denote solution to the IVP

$$x''(t) = f(t, x(t), x'(t)), \quad t \in [a, b] \tag{2.3.9}$$

$$x(a) = \alpha, \quad x'(a) = s \tag{2.3.10}$$

where s is a parameter that can be varied. The IVP (2.3.9) is solved with different values of s with, e.g., the RK4 method until the boundary condition on the right side $x(b) = \beta$ becomes satisfied. As we have noted above, the solution $x(t, s)$ of (2.3.9) depends on the parameter s. Let us define a function

$$F(s) := x(s, b) - \beta.$$

If the BVP (2.3.8) has a solution, then the function $F(s)$ has a root, which is just the value of the slope $x'(a)$ giving the solution $x(t)$ of the BVP. The zeros of $F(s)$ can be found with, for example, Newton's method which is, probably, the best known method for numerical roots finding of real-valued functions.

The idea of the method is to use the first terms of the Taylor series of a function $F(s)$ in the vicinity of the suspected root, i.e.,

$$F(s_n + h) = F(s_n) + F'(s_n)h + \mathcal{O}(h^2).$$

Where s_n is a n^{th} approximation of the root. If we now insert $h = s - s_n$, we obtain

$$F(s) = F(s_n) + F'(s_n)(s - s_n)$$

as the next approximation, s_{n+1} to the root. We choose the zero of this function, i.e.,

$$F(s_{n+1}) = F(s_n) + F'(s_n)(s_{n+1} - s_n) = 0 \Rightarrow s_{n+1} = s_n - \frac{F(s_n)}{F'(s_n)}. \tag{2.3.11}$$

The derivative $F'(s_n)$ can be calculated using the forward difference formula

$$F'(s_n) = \frac{F(s_n + \delta s) - F(s_n)}{\delta s},$$

where δs is small. It is worthy of note that this procedure can be unstable near a horizontal asymptote or local extremal.

All shooting methods use an embedded ODE or Dynamic Algebraic Equation (DAE) solver in order to eliminate the continuous time dynamic system. This is done by first parametrizing the control function $u(t)$ by polynomials, piecewise constant functions or piecewise polynomials. Let us denote the finite control parameters by the vector q, and the resulting control function by $u(t; q)$. The most popular parametrization is piecewise constant controls. In this approach, we choose a fixed grid $0 = t_0 < t_1 < t_2 < \ldots < t_N = t_f$, and N parameters $q_i \in \mathbb{R}^{n_u}, i = 0, 1, \ldots, N-1$ and then we set

$$u(t; q) = q_i \quad \text{if} \quad t \in [t_i, t_{i+1}].$$

Therefore, the dimension of the vector $q = (q_0, \ldots, q_{N-1})$ is Nn_u. We then regard the states $x(t)$ on $[t_0, t_f]$ as dependent variables that are obtained by a forward integration of the dynamic system, starting at x_0 and using the control $u(t; q)$. We denote the resulting trajectory as $x(t, q)$. In order to discretize inequality path constraints, we choose a grid, that is the same as the grid of discretization of the control, at which we check the inequalities. Hence, we transcribe the OCP (2.1.1) into the following NLP:

$$\min_{q \in \mathbb{R}^{Nn_u}} \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}, t, \mathbf{u}; \mathbf{p})dt + \Phi(\mathbf{x}(t_f, q), t_f; \mathbf{p}), \tag{2.3.12}$$

subject to

$$\begin{aligned} &\mathbf{C}_{\min} \leq \mathbf{C}(\mathbf{x}(t_i, q), \mathbf{u}(t_i, q), t_i; \mathbf{p}) \leq \mathbf{C}_{\max} \quad i = 0, \ldots, N-1, && \text{(path constraints)} \\ &\phi_{\min} \leq \phi(\mathbf{x}(t_0, q), \mathbf{x}(t_f, q); p) \leq \phi_{\max}. && \text{(terminal constraints)} \end{aligned}$$

The above problem can be solved by a dense NLP solver in a black box-fashion (Diehl, 2011).

Direct Multiple shooting. The direct multiple shooting method was originally developed by Bock and Plitt (Diehl, 2011). In this method, piecewise control discretization is performed on a grid, exactly as it was explained in direct single shooting. That is, we set

$$u(t) = q_i \quad \text{for} \quad t \in [t_i, t_{i+1}].$$

The difference in this case is that the ODE is solved on each sub-interval $[t_i, t_{i+1}]$, as opposed to solving over the whole interval $[t_0, t_f]$ in the case of single shooting, starting with artificial initial values s_i:

$$\begin{aligned}\dot{x}_i(t; s_i, q_i) &= f(x_i(t; s_i, q_i), q_i), \quad t \in [t_i, t_{i+1}],\\ x_i(t_i; s_i, q_i) &= s_i.\end{aligned}$$

Thus, we obtain the pieces of the trajectory as $x_i(t; s_i, q_i)$. In like manner, we numerically compute the integrals

$$l_i(s_i, q_i) := \int_{t_i}^{t_{i+1}} \mathcal{L}(\mathbf{x}_i(t_i; s_i, q_i), q_i)dt. \tag{2.3.13}$$

Lastly, we choose a grid at which to check the inequality path constraint. Hence the NLP to be solved is given by

$$\min_{s,q} \sum_{i=0}^{N-1} l_i(s_i, q_i) + \Phi(\mathbf{x}(t_f, s_n), t_f; \mathbf{p}), \tag{2.3.14}$$

subject to

$$\begin{aligned}
&x_0 - s_0 = 0, && \text{initial value}\\
&x_i(t_{i+1}; s_i, q_i) - s_{i+i} = 0, \quad i = 0, \dots, N-1, && \text{(continuity)}\\
&\mathbf{C}_{\min} \le \mathbf{C}(\mathbf{x}(t_i, q), \mathbf{u}(t_i, q), t_i; \mathbf{p}) \le \mathbf{C}_{\max} \quad i = 0, \dots, N-1, && \text{(path constraints)}\\
&\phi_{\min} \le \phi(\mathbf{x}(t_0, q), \mathbf{x}(t_f, q); p) \le \phi_{\max}. && \text{(terminal constraints)}
\end{aligned}$$

This NLP can be solved by a sparsity exploiting NLP solver.

Direct Collocation. The third type of direct methods are the direct transcription methods popularly known as *direct collocation*. In the method, the infinite OCP is discretized in both control and states on a fixed and relatively fine grid $\{t_k\}_{k=0}^N$. Recall that each collocation interval corresponds to an integrator step. Let us denote the states on the grid points by $s_k \approx x(t_k)$. We then choose a parametrization of the controls on the same grid, e.g., piecewise constant or piecewise polynomials, with control parameters q_k that yield on each interval a function $u_k(t; q)$.

On each collocation interval $[t_k, t_{k+1}]$, a set of m collocation points $t_k^{(1)}, \dots, t_k^{(m)}$ is chosen and the trajectory is approximated by a polynomial $p_k(t; a_k)$ with coefficient vector a_k. As equalities of the optimization problem, we now require that the collocation conditions (2.2.15) are met at the collocation points:

$$\begin{aligned}
s_k &= p_k(t_k; a_k),\\
f(p_k(t_k^{(1)}; a_k), u_k(t_k^{(1)}; q_k)) &= p_k'(t_k^{(1)}; a),\\
&\vdots\\
f(p_k(t_k^{(m)}; a_k), u_k(t_k^{(m)}; q_k)) &= p_k'(t_k^{(m)}; a).
\end{aligned}$$

This system can be summarized by the vector equation $c_k(s_k, a_k, q_k) = 0$ that has many components as the vector a_i. Furthermore, we also require continuity across interval boundaries. We also approximate the integrals $\int_{t_k}^{t_{k+1}} \mathcal{L}(x, u)dt$ on the collocation intervals by a quadrature formula using the same collocation points, which we denote by $l_k(s_k, a_k, q_k)$. Path constraints are enforced on a grid. Thus,

we obtain a large scale, but sparse NLP:

$$\min_{s,v,q} \sum_{k=0}^{N-1} l_k(s_k, a_k, q_k) + \Phi(\mathbf{x}(t_f, s_n), t_f; \mathbf{p}), \tag{2.3.15}$$

subject to (2.3.16)

$$\begin{aligned}
& s_0 - x_0 = 0, && \text{fixed initial condition} \\
& c_k(s_k, a_k, q_k) = 0, \quad k = 0, \ldots, N-1, && \text{(collocation condition)} \\
& p_k(t_{k+1}; a_k) - s_{k+1} = 0, \quad k = 0, \ldots, N-1, && \text{(continuity condition)} \\
& \mathbf{C}_{\min} \leq \mathbf{C}(\mathbf{x}(t_i, q), \mathbf{u}(t_i, q), t_i; \mathbf{p}) \leq \mathbf{C}_{\max} \quad i = 0, \ldots, N-1, && \text{(path constraints)} \\
& \phi_{\min} \leq \phi(\mathbf{x}(t_0, q), \mathbf{x}(t_f, q); p) \leq \phi_{\max}. && \text{(terminal constraints)}
\end{aligned}$$

The large sparse NLP above needs to be solved by structure exploiting solvers. A very interesting type of orthogonal collocation methods that is often called *pseudospectral optimal control method* uses only one collocation interval, but on this interval, it uses an extremely high order polynomial. State constraints are then typically enforced at all collocation points.

Sensitivity Computation in Shooting Methods. In all shooting methods, it is required to compute the derivatives of the result of ODE integration routine or of a DAE solver on a given interval. For simplicity, let us consider the autonomous ODE case $\dot{x} = f(x)$ on a time interval $[t_0, t_f]$. The case of control or other parameters on which the ODE depends and the time dependence can be covered by state augmentation. Therefore, we regard a starting point s and the evolution of the ODE

$$\dot{x} = f(x), \quad t \in [t_0, t_f], x(t_0) = s. \tag{2.3.17}$$

This gives a solution $x(t; s), \quad t \in [t_0, t_f]$, and we are most interested in the terminal value $x(t_f; s)$ and in the sensitivity matrix

$$G(t) = \frac{\partial x(t; s)}{\partial s}, t \in [t_0, t_f],$$

especially, its terminal value. This matrix $G(t_f) \in \mathbb{R}^{n_x \times n_x}$ can be computed in many different ways. Diehl (2011) stated five of them as follows:

1. External Numerical Differentiation (END).
2. Solution of the Variational Differential Equations.
3. Algorithmic Differentiation (AD) of the Integrator.
4. Internal Algorithmic Differentiation within the Integrator.
5. Internal Numerical Differentiation (IND).

The Penalty Function Approach. One of the ways by which constrained nonlinear programming problems are transformed into an unconstrained problem is by using the penalty function approach. There are three types of penalty function: barrier methods, partial penalty functions, and global penalty functions (Smith and Coit, 1995). No feasible solution is considered in the case of barrier penalty method. For partial penalty function, penalty is only applied near the feasible boundary. Global penalty is applied over the whole infeasible region. Generally, a penalty function is defined as follows. Given an optimization problem,

$$\min f(\mathbf{x}), \tag{2.3.18}$$

subject to

$$\mathbf{x} \in A,$$
$$\mathbf{x} \in B,$$

where $\mathbf{x}$ is a vector of decision variables. Assuming the constraints $\mathbf{x} \in A$ are easy to satisfy and the constraints $\mathbf{x} \in B$ are difficult to satisfy, the problem can be reformulated as

$$\min f(\mathbf{x}) + p(d(\mathbf{x}, B)), \tag{2.3.19}$$

subject to

$$\mathbf{x} \in A, \tag{2.3.20}$$

where $d(\mathbf{x}, B)$ is a metric function describing the distance of the solution of vector $\mathbf{x}$ from the region B, and $p(\cdot)$ is a monotonically non-decreasing penalty function such that $p(0) = 0$. Practically, the constraints $\mathbf{x} \in A, \mathbf{x} \in B$ are expressed as inequality and equality constraints. Although, penalty functions make NLP problems easier to deal with, this method has its well-known disadvantages of which some are illustrated in Becerra et al. (2004).

All the solution approaches to OCP discussed in this thesis are summarized in Figure 2.2

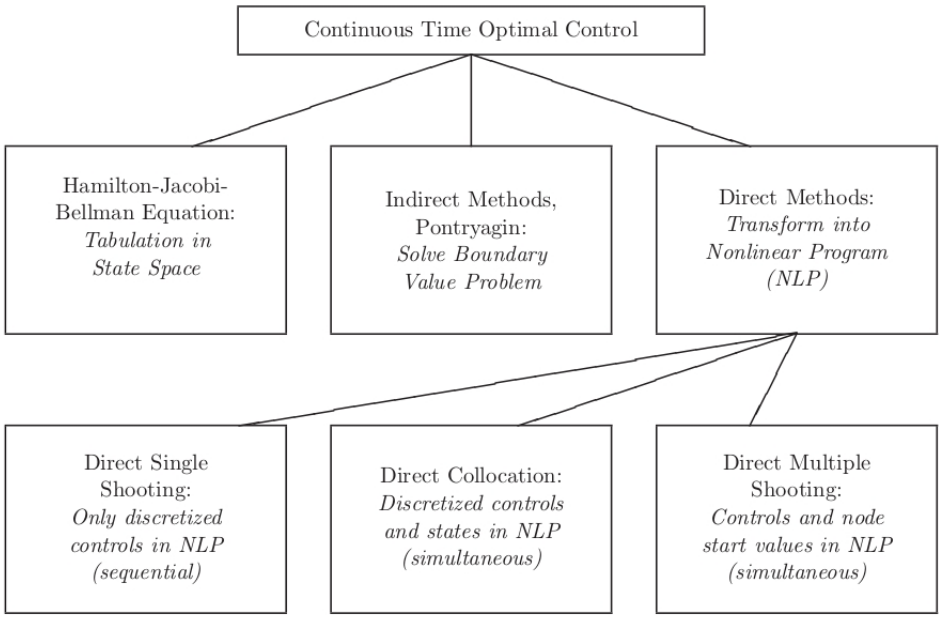


Figure 2.2: Optimal control family tree

3. Aircraft Conflict Avoidance

3.1 Introduction

Air Traffic Control (ATC) is the service provided by air traffic controllers at an airport. They direct aircraft on the ground and through the controlled airspace. The primary purpose of ATC is to prevent collisions. Aircraft sharing the same airspace are said to be potentially in conflict when they are too close to each other according to their predicted trajectories (Cafieri and Durand, 2012). Aircraft conflict avoidance is crucial in Air Traffic Management (ATM), and is even more challenging as air traffic is continuously increasing (Cellier et al., 2013). As traffic continues to increase, En-Route capacity, Europe in particular, becomes a serious problem. Aircraft conflict resolution and resolution monitoring are still done manually by controllers (Durand et al.). Civil aviation authorities around the world recruit more controllers as the traffic increases. It seems the available computing power is not used in this respect because the principle of control has not changed for the last 30 years (Durand et al., 1997). The minimum horizontal distance that must be kept between two aircraft is *5NM* while the vertical safety distance is *1NM* = 1000ft (Cellier et al., 2012), see Figure 3.1. Conflict avoidance typically consists of two processes namely: conflict detection and conflict resolution (Hu et al., 2000). In the conflict detection stage, the motions of aircraft are predicted based on their positions, heading and flight plans, and conflict situations are identified. This information is then used in the resolution stage to replan the trajectories of the aircraft involved in the conflict (Hu et al., 1999). Among the various ways of resolving aircraft conflict are change in trajectory (heading), change in flight level, and change in velocity.

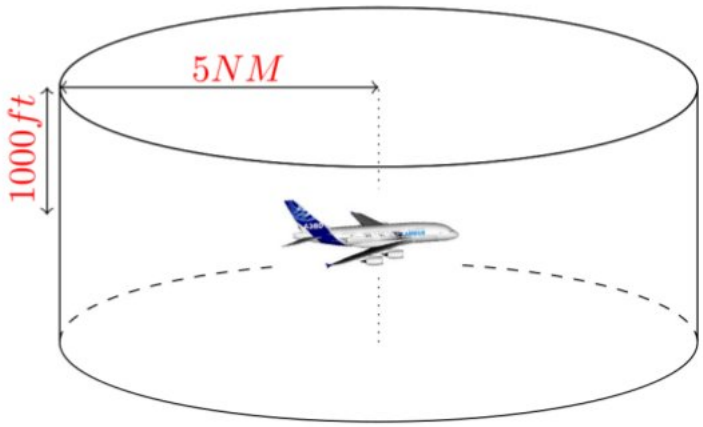


Figure 3.1: Minimum separation requirement

3.2 Literature Review

Not much work has been done in this area. Hu et al. (1999) presented a probabilistic approach to aircraft conflict prediction. Aircraft motion was modelled as a deterministic trajectory and Brownian motion perturbation. A survey was given in Kuchar and Yang (2000) on conflict detection and resolution. An approach to resolve aircraft conflict using genetic algorithm was presented in Durand et al.. The

European project ERASMUS (En-Route Air traffic Soft Management Ultimate System) considered speed regulation and suggested a small velocity change range to allow a small amount of control without informing the air traffic controller (Cellier et al., 2012). Based on velocity variation, new models and solution approaches have been developed (Cellier et al., 2013). Although the subliminal speed approach looks very promising, it can not handle a few cases, e.g., two aircraft flying along the same trajectory in opposite direction (Cafieri and Durand, 2012).

3.3 Mathematical Models

3.3.1 Background to the Model. The notations used in the continuous model that is aimed to be used to achieve separation base on speed change as described in Cellier et al. (2012) are as follows.

The acceleration u_i with respect to each aircraft i is the command on the system. x_i and v_i are position and velocity of aircraft i; with $i \in I = \{1, \ldots, n\}$ where n is the total number of aircraft involved. It is assumed that all the aircraft are at the same altitude and are travelling along straight trajectories (see Figure 3.2). For each aircraft i, velocity v_i, and acceleration u_i are bounded (i.e belonging to $\{\underline{v}_i, \overline{v}_i\}$ and $\{\underline{u}_i, \overline{u}_i\}$ respectively); t, t_0, t_f are instantaneous time, initial time and final time respectively. D is the minimum required horizontal separation distance between two aircraft and d_i is the direction heading of the i^{th} aircraft. The final time t_f of manoeuvre is identical for all aircraft.

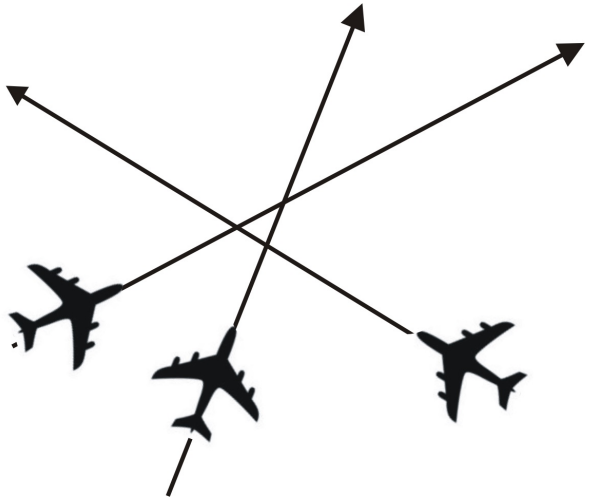


Figure 3.2: Aircraft flying along their predicted trajectories

3.3.2 Optimal Control Model. The mathematical model presented in Cellier et al. (2012) is given as

$$\min_u \sum_{i=1}^{n} \int_{t_0}^{t_f} u_i^2(t)dt,$$

subject to

$$\begin{aligned}
&\dot{v}_i(t) = u_i(t), && \forall t \in [t_0, t_f], \forall i \in I \\
&\dot{x}_i(t) = v_i(t)d_i, && \forall t \in [t_0, t_f], \forall i \in I \\
&\underline{u_i} \leq u_i(t) \leq \overline{u_i}, && \forall t \in [t_0, t_f], \forall i \in I \\
&\underline{v_i} \leq v_i(t) \leq \overline{v_i}, && \forall t \in [t_0, t_f], \forall i \in I \\
&x_i(t_0) = x_i^0, \quad v_i(t_0) = v_i^0, && \forall i \in I \\
&x_i(t_f) = x_i^{t_f}, \quad v_i(t_f) = v_i^{t_f}, && \forall i \in I \\
&D^2 - \|x_i(t) - x_j(t)\|^2 \leq 0 && \forall t \in [t_0, t_f], \forall i \neq j.
\end{aligned} \tag{3.3.1}$$

Both the direct and indirect methods was used in Cellier et al. (2012). The flight time was divided into pre-zone, zone and post-zone. Pre-zone is the period before conflict, zone is the conflict period and post-zone is the period when conflict has been resolved. It was assumed that a pre-processing is done to identify the conflict region and the model in (3.3.1) is used to resolve the conflict while another model is used in pre-zone and post-zone. Indirect method was used for the model used to get the trajectories during the pre-zone and post-zone while a direct shooting method (see Section 2.3.7) was used for the conflict zone. The discretization of the model is as follows. The control u_i was descretized to get the following descrete model:

$$\min_u \sum_{i=1}^{n} \sum_{k=0}^{K-1} (u_i(k))^2,$$

subject to

$$\begin{aligned}
&v_i(k+1) = Num_{v_i}(u_i(k)), && \forall k \in \{0, \ldots, K-1\}, \forall i \in \{1, \ldots, n\} \\
&x_i(k+1) = Num_{x_i}(v_i(k)), && \forall k \in \{0, \ldots, K-1\}, \forall i \in \{1, \ldots, n\} \\
&\underline{u_i} \leq u_i(k) \leq \overline{u_i}, && \forall k \in \{0, \ldots, K-1\}, \forall i \in \{1, \ldots, n\} \\
&\underline{v_i} \leq v_i(k) \leq \overline{v_i}, && \forall k \in \{0, \ldots, K-1\}, \forall i \in \{1, \ldots, n\} \\
&x_i(0) = x_i^0, \quad v_i(0) = v_i^0, && \forall i \in \{i, \ldots, n\} \\
&x_i(t_f) = x_i^{t_f}, \quad v_i(t_f) = v_i^{t_f}, && \forall i \in \{i, \ldots, n\} \\
&D^2 - \|x_i(k) - x_j(k)\|^2 \leq 0, \quad \forall k \in \{0, \ldots, K\}, \quad i, j \in \{i, \ldots, n\}^2, \quad i \neq j,
\end{aligned} \tag{3.3.2}$$

where $x_i(k), v_i(k)$ and $u_i(k), k = 0, \ldots, K-1$, are the discretized position velocity and acceleration of aircraft i; and Num is a numerical integrator, e.g., Euler and Runge-Kutta method; used to integrate the state equations over the time horizon i.e., for $t \in [t_0, t_f]$.

One of the difficulties in solving this control problem is the constraints on the state variables, especially the separation constraints. Another drawback of this approach is the decomposition of the flight time to zones, although this makes the computer implementation to run faster according to the results presented. To address these problems, we have reformulated this model.

3.3.3 Modified Mathematical model. Recall that the constraints on the velocity of the aircraft are given as

$$\begin{aligned}
&\underline{v_i} \leq v_i(t) \leq \overline{v_i},\\
&\Rightarrow \underline{v_i} \leq v_i(t), \quad v_i(t) \leq \overline{v_i},\\
&\Rightarrow \underline{v_i} - v_i(t) \leq 0, \quad v_i(t) - \overline{v_i} \leq 0,
\end{aligned}$$

since $\underline{v_i} = v_i(t_0) - 0.6v_i(t_0) = 0.4v_i(t_0)$ and $\overline{v_i} = v_i(t_0) + 0.3v_i(t_0) = 1.3v_i(t_0)$, we have $v_i(t) - 0.4v_i(t_0) \geq 0, \quad 1.3v_i(t_0) - v_i(t) \geq 0.$

The modified control model is given as

$$\min_u \sum_{i=1}^{n} \int_{t_0}^{t_f} u_i^2(t)dt + \alpha_1 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \int_{t_0}^{t_f} \exp(D^2 - (x_i(t) - x_j(t))^2)dt \right) +$$
$$\alpha_2 \left(\sum_{i=1}^{n} \int_{t_0}^{t_f} \exp(v_i(t) - 1.3v_i(t_0)) + \exp(0.4v_i(t_0) - v_i(t))dt \right),$$

subject to

$$\begin{aligned}
&\dot{v}_i(t) = u_i(t), && \forall t \in [t_0, t_f], \forall i \in I\\
&\dot{x}_i(t) = v_i(t)d_i, && \forall t \in [t_0, t_f], \forall i \in I\\
&x_i(t_0) = x_i^0, \quad v_i(t_0) = v_i^0, && \forall i \in I\\
&x_i(t_f) = free, \quad v_i(t_f) = free, && \forall i \in I
\end{aligned} \tag{3.3.3}$$

where α_1 and α_2 are penalty parameters to be chosen. The goal is to minimize a quadratic energy-dependent cost function depending on speed variation. This takes into account the contribution of each aircraft.

In our approach, we do not want conflicts to occur before we start to resolve them. We get the predicted trajectory at every time window and resolve the conflict from the beginning. Further, we cleverly handle the constraints by writing them as penalty function (see Section 2.3.7) in the objective function. Another modification is the relaxation on the over-specified conditions. The starting position and velocity is known but the final position and velocity are not known before hand thereby transforming the problem to a fixed time free endpoint problem. The restriction on the acceleration is ignored because it is automatically satisfied if the constraint on the velocity is satisfied.

3.3.4 Solution Method. The continuous model was transformed from an infinite dimensional space to a finite space by discretizing the control u_i over an equally spaced time grid i.e., $u_i(k), k \in \{t_0 = 0, \ldots, K = t_f\}$. An initial guess is made using the piecewise constant parametrization method. The discrete model is as given below.

$$\min_u \sum_{i=1}^{n}\sum_{k=0}^{K-1}(u_i(k))^2 + \alpha_1\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=0}^{K-1}\exp(D^2-(x_i(k)-x_j(k))^2)\right)$$
$$+\alpha_2\left(\sum_{i=1}^{n}\sum_{k=0}^{K-1}\exp(v_i(k)-1.3v_i(t_0))+\exp(0.4v_i(t_0)-v_i(k))\right), \tag{3.3.4}$$

subject to

$$\begin{aligned}
&v_i(k+1) = Num_{v_i}(u_i(k)), \quad \forall k \in \{0,\dots,K-1\}, \forall i \in \{1,\dots,n\}\\
&x_i(k+1) = Num_{x_i}(v_i(k)), \quad \forall k \in \{0,\dots,K-1\}, \forall i \in \{1,\dots,n\}\\
&x_i(t_0) = x_i^0, \quad v_i(t_0) = v_i^0, \quad x_i(t_f) = free, \quad v_i(t_f) = free, \qquad \forall i \in I
\end{aligned}$$

The initial u_i is used to solve the state equation to get the predicted trajectories. These are passed into an optimization routine to get optimal u_i. After finding the optimal control for the respective aircraft, the state equations are solved again to get the optimal trajectories.

3.4 Numerical Experimentation

For simplicity, we experiment with a case of two aircraft. For the two aircraft, the objective function to be minimized is

$$\min_u \sum_{k=0}^{K-1}\left((u_1(k))^2+(u_2(k))^2\right) + \alpha_1\left(\sum_{k=0}^{K-1}\exp(D^2-(x_1(k)-x_2(k))^2)\right)$$
$$+\alpha_2\left(\sum_{k=0}^{K-1}\exp(v_1(k)-1.3v_1(t_0))+\exp(v_2(k)-1.3v_2(t_0))\right.$$
$$\left.+\exp(0.4v_1(t_0)-v_1(k))+\exp(0.4v_2(t_0)-v_2(k))\right) \tag{3.4.1}$$

subject to

$$\begin{aligned}
&v_1(k+1) = Num_{v_1}(u_1(k)),\\
&x_1(k+1) = Num_{x_1}(v_1(k)),\\
&v_2(k+1) = Num_{v_2}(u_2(k)),\\
&x_2(k+1) = Num_{x_2}(v_2(k)),\\
&x_1(t_0) = x_1^0, \quad v_1(t_0) = v_1^0,\\
&x_1(t_f) = free, v_1(t_f) = free,\\
&x_2(t_0) = x_2^0, \quad v_2(t_0) = v_2^0,\\
&x_2(t_f) = free, v_2(t_f) = free.
\end{aligned} \tag{3.4.2}$$

4. Results and Discussion

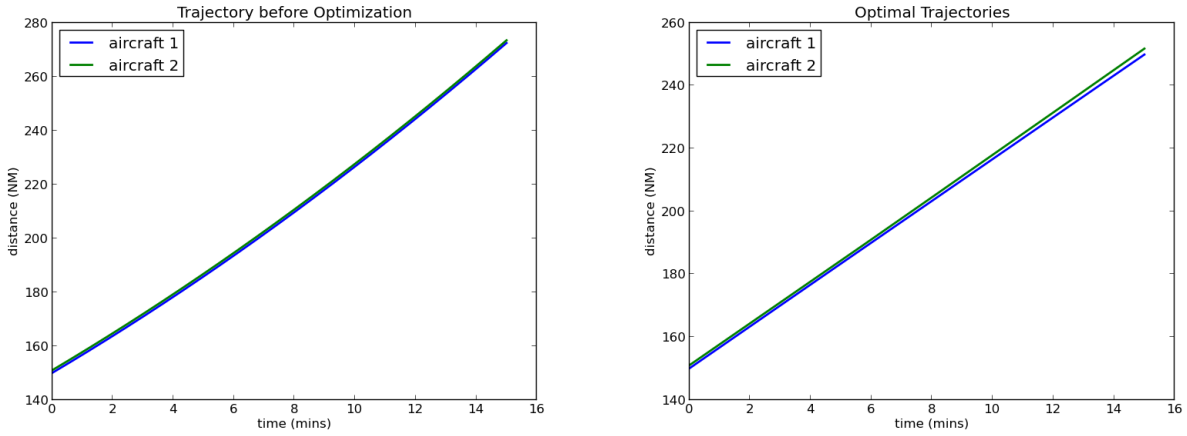
The experiment was performed for two cases with two aircraft. We have chosen carefully, the initial conditions of the two aircraft so that there will be conflict. The results are presented below.

For the first case, the initial data were chosen for the two aircraft as

$$d_1 = d_2 = 1, \quad h = 0.001, \quad initial \quad time = 0 \quad \text{min}, \quad final \quad time = 15 \quad \text{mins},$$
$$x_1(0) = 150NM, x_2(0) = 151NM, \quad v_1(0) = v_2(0) = 400NM/h, \quad u_1(k) = u_2(k) = 720NM/h^2$$
$$\forall k \in \{t_0 = 0, \ldots, t_f = 15\}$$

where h is the step size of the numerical integration. In this implementation, $\alpha_1 = \alpha_2 = 1$. The descretized model was solved with RK for system and *fmin* minimization routine in Scipy.optimize was used to find optimal u_1 and u_2 together with their corresponding trajectories.

The following results were obtained.

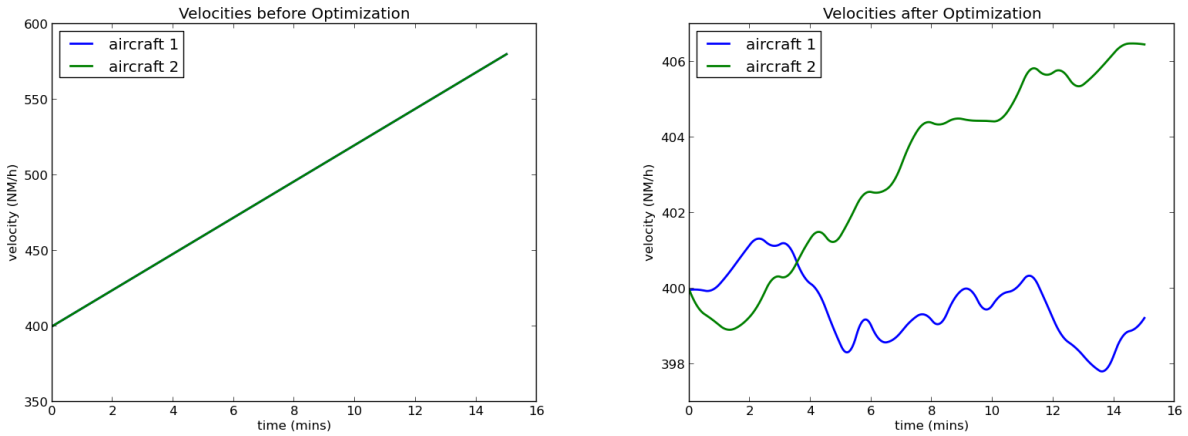


(a) Trajectories before optimization

(b) Trajectories after optimization

Figure 4.1: Trajectories of the aircraft

Figure 4.1 shows the predicted and optimal trajectory of the two aircraft. It can be observed from that the aircraft remain close throughout the predicted trajectories (Figure 4.1a) whereas Figure 4.1b shows how the aircraft separates out along the optimal trajectories.

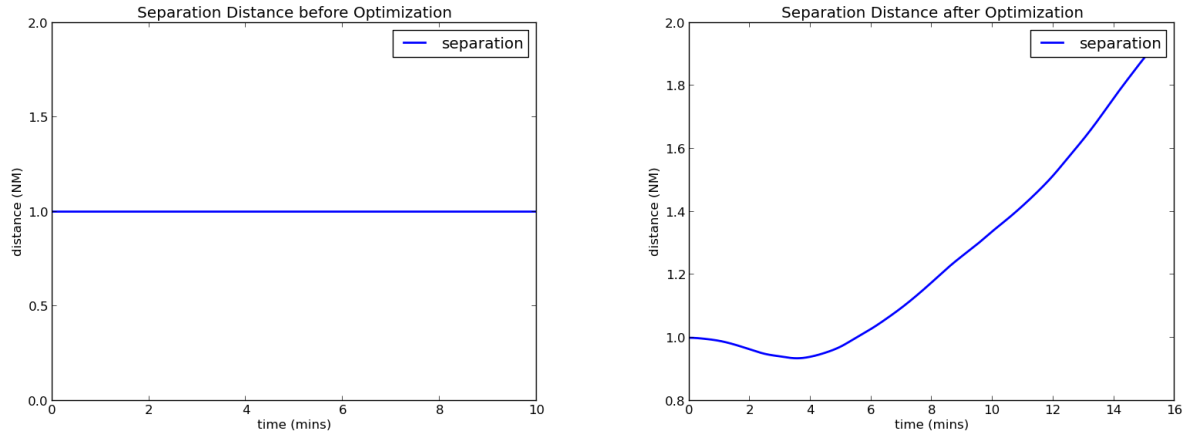


(a) Velocities before optimization

(b) Velocities after optimization

Figure 4.2: Velocities of the aircraft

With the initial (guessed) acceleration, the velocity of the aircraft in Figure 4.2a was increasing uniformly without bound. Figure 4.2b shows how the change in velocity has been normalised after the control has been optimized. One of the aircraft has to increase its velocity a little while the other slows down. These changes are very small and the effect is expected not to be noticed on the flight schedule.



(a) Separation before optimization

(b) Separation after optimization

Figure 4.3: Separation distance between the aircraft

The norm of separation distance through out the journey was constant $= 1$ as it can be seen in Figure 4.3a. This implies that the potential conflict that starts from the time $t = 0$ mins will continue till the end of the time interval if not resolved. Since the trajectory is straight, the option to resolve the conflict in this case is speed change. The curve of the separation distance shown in Figure 4.3b is monotonically increasing. This shows that the separation distance is increasing over time. The optimal acceleration of the two aircraft is shown in Figure 4.4.

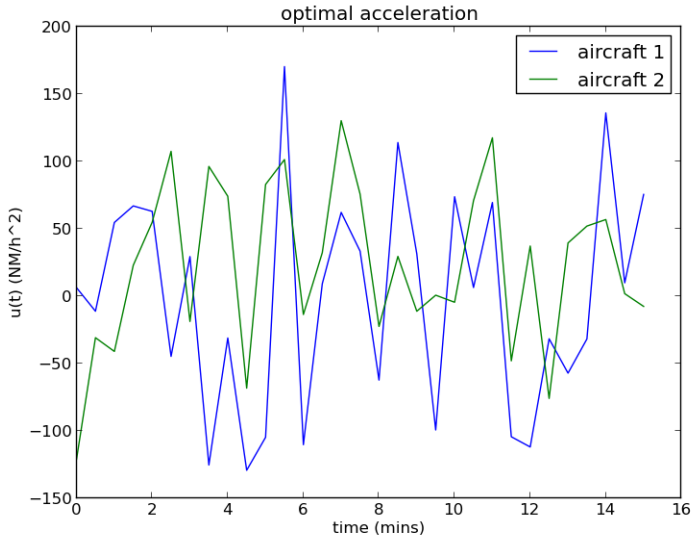


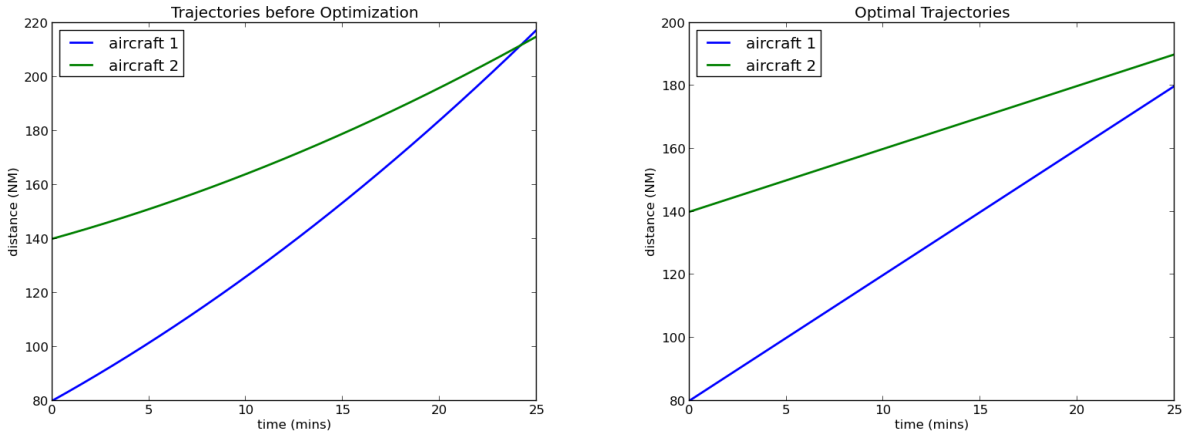
Figure 4.4: Optimal control u_1 and u_2

The execution time for this program was 25s.

For the second experiment, the initial conditions are

$d_1 = 0.6, \quad d_2 = 0.4, \quad h = 0.01, \quad initial \quad time = 0 \quad \text{min}, \quad final \quad time = 25 \quad \text{mins},$
$x_1(0) = 80NM, x_2(0) = 140NM, \quad v_1(0) = 400NM/h, \quad v_2(0) = 300NM/h, \quad u_1(k) = u_2(k) = 720NM/h^2$
$\forall k \in \{t_0 = 0, \ldots, t_f = 25\}, \quad \alpha_1 = 0.4, \quad \alpha_2 = 0.1$

The results are as follows.

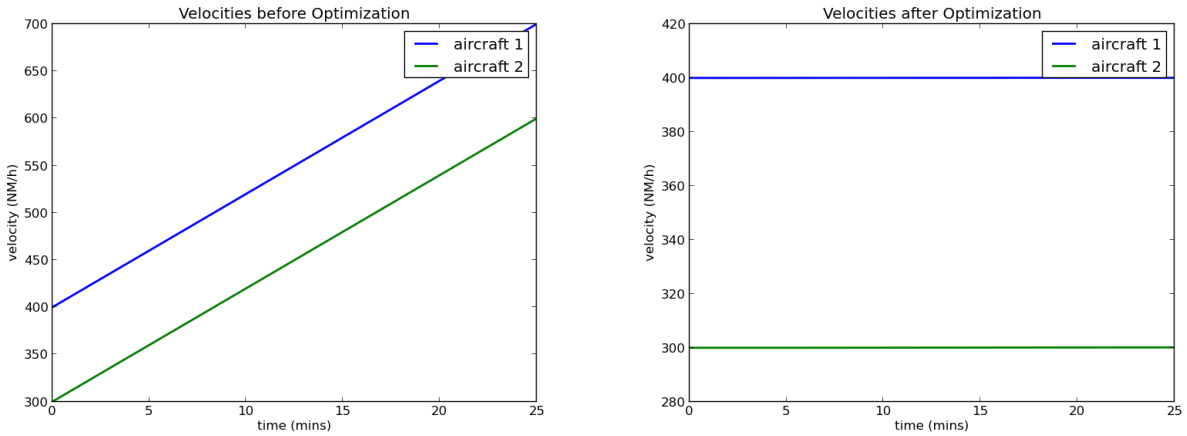


(a) Trajectories before optimization

(b) Trajectories after optimization

Figure 4.5: Trajectories of the aircraft

From Figure 4.5a, we observe that there is a conflict at about 24mins. This conflict is resolved after optimization and the optimal trajectory is shown in Figure 4.5b.

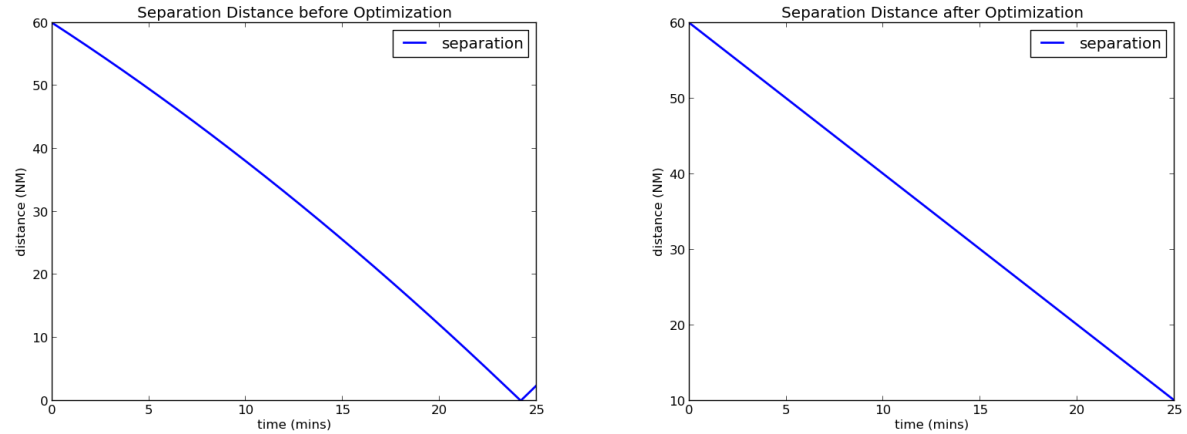


(a) Velocities before optimization

(b) Velocities after optimization

Figure 4.6: Velocities of the aircraft

Figure 4.6b shows an almost constant (optimal) velocity which resolves the conflict and respect the bounds on velocity and acceleration as opposed to the predicted velocity in Figure 4.6a which grows drastically.



(a) Separation before optimization

(b) Separation after optimization

Figure 4.7: Separtion distance between the aircraft

Although, the separation distances shown in Figure 4.7 are both monotonically decreasing curves but the speration distance goes to zero after 24min in Figure 4.7a which implies that there is a collision whereas the separation distance at the final time after optimization is 10NM (see Figure 4.7b).

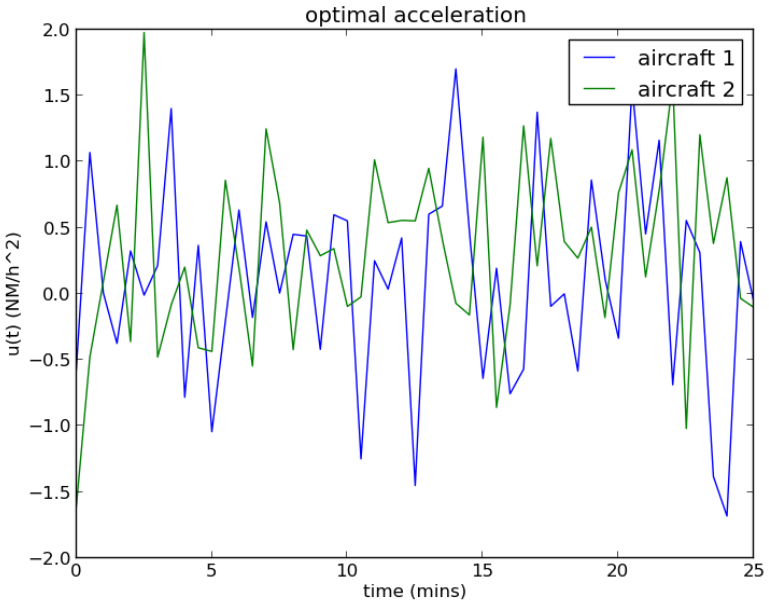


Figure 4.8: Optimal control

The program runs for 377.96s for this problem and the final value of the objective function is 2.218×10^1

Figure 4.4 and 4.8 show the optimal acceleration for the two scenarios considered. We can observe from these two figures that the change in acceleration required to resolve the conflict in the first case is much higher than the one required in the second case. This implies that the cost of resolving the conflict in the first case will be much higher than in the second case.

5. Conclusion

5.1 Concluding Remarks

Aircraft conflict avoidance is a very crucial issue in air traffic management. A solution approach to aircraft conflict resolution via optimal control was presented in this report. The optimal control model in Cellier et al. (2012) was reformulated and direct method of solving optimal control problem was applied to the resulting model. One of the contributions of this work is the clever way that was used to deal with the separation constraint that made the formulations in Cellier et al. (2012) very difficult to handle. The continuous model was transformed to NLP problem and the control u was discretized. Python programming language was used for the experimental results presented.

5.2 Future Direction

The problem solved here is fixed terminal time - free end point problem. A more realistic problem in this case is fixed end point problem whereby, the whole flight time will be considered. One main challenges to overcome is unequal flight time for all the aircraft that may be involved in conflict. Future work can also focus on looking at the problem in 2 dimension.

Acknowledgements

All gratitude, adoration and glorification are due to Allah (SWT) for seeing me thus far.

I appreciate the love, care and supports of my parents, I say, "may Allah care for you as you cared for me when I was young". I am also grateful to my entire family, friends and associates. Your solidarity and words of encouragement are my sources of inspiration.

Furthermore, I am indebted to the entire AIMS family for making my study a smooth one, you are wonderful people. I must also recognise the untiring efforts of my supervisor, Prof. Montaz Ali, and my tutor, Adriaan V. Vuuren. Thank you indeed.

I will like to conclude this piece by appreciating my fellow Nigerian students. You make an important part of my memories at AIMS. Above all, thank you all.

Dedication: To my wife

References

V. M. Becerra et al. Solving optimal control problems with state constraints using nonlinear programming and simulation tools. *IEEE Transactions on education*, 47(3):377–384, 2004.

J. T. Betts. *Practical Methods of Optimal Control using Non-linear Programming*. Society for Industrial and Applied Mathematics Philadelphia, 2001.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

S. Cafieri and N. Durand. Aircraft deconfliction with speed regulation: new models from mixed-integer optimization. *Journal of Global Optimization*, pages 1–17, 2012.

L. Cellier, S. Cafieri, and F. Messine. Hybridizing direct and indirect optimal control approaches for aircraft conflict avoidance. In *ADVCOMP 2012, The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 42–45, 2012.

L. Cellier, S. Cafieri, F. Messine, et al. Optimal control approaches for aircraft conflict avoidance using speed regulation: a numerical study. In *ISIATM 2013, 2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, 2013.

M. Diehl. *Numerical Optimal Control*. Optimization in Engineering Center (OPTEC), June 2011.

N. Durand, J.-M. Alliot, and J. Noailles. Automatic aircraft conflict resolution using genetic algorithms.

N. Durand, J.-M. Alliot, and O. Chansou. Optimal resolution of en route conflicts. In *Procceedings of the 1rst USA/Europe Seminar*, 1997.

L. C. Evans. An introduction to mathematical optimal control theory version 0.2. Department of Mathematics University of California, Berkeley.

J. Hu, J. Lygeros, M. Prandini, and S. Sastry. Aircraft conflict prediction and resolution using brownian motion. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pages 2438–2443. IEEE, 1999.

J. Hu, M. Prandini, and S. Sastry. Optimal maneuver for multiple aircraft conflict resolution: a braid point of view. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 4, pages 4164–4169. IEEE, 2000.

J. K. Kuchar and L. C. Yang. A review of conflict detection and resolution modeling methods. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):179–189, 2000.

D. Liberzon. *Calculus of Variations and Optimal Control Theory*. Princeton University Press, 2012.

A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

S. Sager. *Numerical methods for mixed-integer optimal control problems*. PhD thesis, Universistat Heidelberg, 2006.

A. Sasane. Mat305: Control theory (lecture note). London School of Economics, September 2004.

A. Smith and D. Coit. *Penalty Functions*. Department of Industrial Engineering, University of Pittsburgh, USA, September 1995.

R. T. Woodward. End points and transversality conditions (agec 637), October 2013.