

Angewandte Mathematik und Optimierung Schriftenreihe
Applied Mathematics and Optimization Series
AMOS # 24(2015)

Aminanmu Maolaaisha

Free-Flight Trajectory Optimization by Mixed
Integer Programming

Herausgegeben von der
Professur für Angewandte Mathematik
Professor Dr. rer. nat. Armin Fügenschuh

Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
Fachbereich Maschinenbau
Holstenhofweg 85
D-22043 Hamburg

Telefon: +49 (0)40 6541 3540
Fax: +49 (0)40 6541 3672

e-mail: appliedmath@hsu-hh.de
URL: <http://www.hsu-hh.de/am>

Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Print 2199-1928
Angewandte Mathematik und Optimierung Schriftenreihe (AMOS), ISSN-Internet 2199-1936

University of Nice-Sophia Antipolis
University of L'Aquila
University of Hamburg

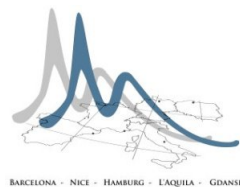
MASTER THESIS

Free-Flight Trajectory Optimization by Mixed Integer Programming

Author:
Aminanmu Maolaaisha

Supervisor:
Armin Fügenschuh

A thesis submitted to fulfillment of the requirements
for the degree of master in science



Declaration of Authorship

I, Aminanmu Maolaisha, declare that this thesis titled, Free-Flight Trajectory Optimization by Mixed Integer Programming, and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

The growing commercial air travel, fuel price and air pollution require to find an optimal solution for flight routing problem. In this thesis, we develop a trajectory optimization algorithm for Free-Flight, which will produce wind optimal trajectory in horizontal direction. Flying wind optimal trajectory will bring fuel saving.

We formulate the problem as a mixed integer program which concentrates on the time dependence and discrete aspects. We use commercial solvers, CPLEX, SCIP, to solve the model. Concerning the nonlinearities of the constraints, we approximate it by using piecewise linear functions such as special ordered sets Type2, lambda method, and delta method.

We also give a good mapping that can deal with the roundness of the earth so that the algorithm is more accurate and practical. At the end, we give practical result and simulation of the trajectory. From our model, the result shows that with weak wind we can save from 1.5% fuel to 3.5% , in strong wind we can save around 6% to 13% fuel.

We hope in the future, we combine our horizontal optimal trajectory with vertical optimization profile, get an optimal trajectory in 4D (2D in horizontal, 1D in vertical and 1D in time), which will be optimal in both in horizontal and vertical direction, consumes as less fuel as possible.

Acknowledgements

First of all, I want to express my thanks to my supervisor Prof. Dr. Armin Fügenschuh. He gave me the opportunity to work on such an interesting project involved real-world application. His intelligence, diligence and kindness is always big motivation for me. He supported me with helpful suggestions, creative ideas, necessary documents. We always have long hours friendly discussion on the topic which helped me most.

I want to thank Prof. Bruno Rubino and Prof. Francois Delarue for all the support during my master program, which enabled me to overcome all difficulties I had. I am grateful to Prof. Cedric Bernardin and Prof. Klaus Engel for having belief on me and the acceptance of being referee. I would like to thank our industry partners Lufthansa System for supporting my work by providing all necessary data and all relative knowledge about Aircraft. Many thanks to all my colleagues of the Optimization group as well as the group of Numerical Analysis, especially my college Eric who gave me his endless support and guide with his rich experience in operations research and computer science.

My special thanks goes to my family, My mother for always supporting me with my choice. My brothers and sister who are always keeping me feel beloved with their love. And finally, I am very thankful to my friend Abdigeni for his patience and his great belief on me...

Contents

Declaration of Authorship	i
Abstract	ii
Acknowledgements	iii
Contents	iv
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Introduction	1
1.2 Problem Statement and Methodology	1
1.3 Structure of the Thesis	2
2 Free Flight Routing Problem	3
2.1 Free-Flight	3
2.2 Flight Trajectory Optimization	3
2.3 Previous Approach to the Problem	4
3 Optimization With Mixed Integer Linear Programming	7
3.1 Discrete Optimization Problem	7
3.2 Mixed Integer Linear Programming	9
3.3 solution techniques	9
3.3.1 Brunch and bound	10
3.3.2 Cutting plane	12
3.3.3 Branch-and-Cut	13
4 Approximating the Non-linear Functions	14
4.1 1D Nonlinear Function Approximation by Piecewise Linear Functions	14
4.1.1 Linearization of Functions: SOS Polytopes (SOS Type 2)	16
4.1.2 Binary Linearization Model: Convex combination (lambda method)	18
4.1.3 Binary Linearization Model: Incremental (Delta method)	19
4.2 2-Dimension Nonlinear Functions Approximation	20
4.2.1 Linearization of 2D-Functions: SOS Polytopes (SOS Type 3)	21

4.2.2	Linearization of 2D-Functions: Lambda method	22
4.2.3	Linearization of 2D-Functions: Generalized Delta method	23
5	Model Formulation with MILP	27
5.1	Problem Description and Data Analysis	27
5.1.1	Weather Data	27
5.1.2	Fuel Consumption Data	31
5.2	Aircraft Motion Equation	34
5.3	Mathematical Model	36
5.3.1	MILP Formulation for Free-Flight	37
5.3.2	2D X-Y Plane Model	39
5.3.3	Trip 1: Implementation in SCIP with lambda Method	40
5.4	Linear Approximation of Second Order Cones	43
5.5	Speed Discretization	45
6	Adapted Model on Earth Surface with Real World Data	49
6.1	Mapping	49
6.2	3D Realistic Model with Real World Data	55
6.3	Trip 2 with Lambda Method	56
6.4	Trip 2 with Delta Method	58
6.5	Trip 3 with Delta method	64
7	Conclusion	69
7.1	Error Estimation for Model	69
7.2	Conclusion	70
7.3	Future Plan for Project	70

List of Figures

3.1	Bround-and-Bound tree	11
4.1	Approximation of non-linear function by piecewise linear functions	15
4.2	Brunching on SOS2	18
4.3	Lambda method for approximating nonlinear functions	18
4.4	Delta method for approximation non-linear functions	20
4.5	Approximation of nonlinear function by piecewise linear function	21
4.6	Brunching on SOS3	22
4.7	Triangulation and Special Ordering	24
4.8	Simplex	24
5.1	north and east wind component	28
5.2	wind data given by Luthansa Airline	29
5.3	Wind bars in North Atlantic Ocean at flight level 390	31
5.4	wind bars in Europe at flight level 390	31
5.5	Unit distance fuel consumption data	32
5.6	Fuel consumption surface for 27 speed grids and 15 weight grids	33
5.7	Fuel consumption as a function of speed for 3 different weight	34
5.8	Aircraft true speed+ wind effect	34
5.9	The wind field	41
5.10	Optimal speed for unit distance fuel consumption function for aircraft type A320	45
5.11	Optimal solution by using wind	47
5.12	Trajectory when we don't consider any influence of the wind(Assume there is no wind	48
6.1	Mapping the great circle connecting departure and arrival point to the equator	50
6.2	Mapping illustration	52
6.3	The projection of the area on equator	54
6.4	Projection of the area on x-y plane	54
6.5	Mapping	56
6.6	Optimal trajectory	58
6.7	Triangle ordering for Delta Method	59
6.8	Trajectory in stronger wind	62
6.9	Trajectory in stronger wind	63
6.10	Trajectory for trip 3 with weak wind with 3 speed, 4 weight grids	64
6.11	Trajectory for trip 2 with stronger wind	65
6.12	Trajectory for trip 2 with weak wind(27×8)	67

6.13 Trajectory for trip 2 with stronger wind 27×8	67
---	----

List of Tables

5.1	Units for variables in model 5.3	43
6.1	grid points	63
6.2	Trip 3 experiment result	66
7.1	Error estimation	69

Chapter 1

Introduction

1.1 Introduction

The rapid growth in commercial air travel, is putting immense pressure on the ATC (Air Traffic Control) system. Air traffic has doubled every 15 years in the past, and is expected to double again in the next 15 [1]. Consequently, for modernizing ATC to meet the demands for enhanced capacity, efficiency, and safety, the concept of "Free-Flight" (FF) was proposed by the RTCA (1995). Generally, as aircraft fly through the sky, they follow pre-planned routes, much like motorways on the ground that given by ATC which called air traffic network. This trajectory are more robust and can not handle with increasing air traffic. Since air traffic levels have doubled in the last decade, airspace design must be continuously rethought to provide the best and the shortest routes for the increasing number of flights.

1.2 Problem Statement and Methodology

Theoretically, a real Free-Flight can follow any route in the continuous three dimensional space instead of following a discretized network. In this thesis, for given departure and arrival airport, type of the aircraft, aircraft performance data and weather data, we want to develop an algorithm which can generate the optimal trajectory with minimum fuel cost for the trip. By following this trajectory, we are flying wind optimal route which leads to fuel and time saving. Our aim here is to find optimal trajectory considering horizontal direction assuming the altitude is fixed. For future work we are looking forward to add vertical optimization, and combine both horizontal and vertical profile. For model formulation, we choose to use Mixed Integer Programming. The advantage

of using mixed integer programming is that, if it can find, it will give us global optimal solution and if not it will tell us how far we are from optimal solution. Another reason using MIP is considering the practical issue, when we use our algorithm in practical application, we also need to consider restricted airspace. However, continuous approaches (differential equations) can not handle with restricted airspace. MIP method is the becoming the most natural choice to handle with most of the restriction in Free-Flight routing problem. Therefore, in this thesis, we worked on to develop an algorithm for Free-Flight routing problem, which can give the optimal trajectory with minimum fuel cost. And we achieved a good result.

1.3 Structure of the Thesis

In the first and second chapter, We introduce the problem we want to solve, the methods we use, and the background information. We also analyse the previous works done in this field up to now and drawbacks. Third chapter is about optimization problem with MILP technique and the MILP solving techniques such as branch and bound, branch and cut. Chapter four is about linear approximation methods we used in our algorithm. Chapter five and Chapter six, is the model formulation in simple 2D x-y plane and 3D real world trip (considering on Earth surface) with real wind data given by Lufthansa system. We also give the optimal trajectory, solving time for the model, presented how much fuel we save by using our method and how to choose the parameters for speed, wind and weight grids. Chapter 7 is a conclusion for our work, we presented the error estimation, the conclusion from our experiment as well as the future development of the project in order to improve the model and get it used in Lufthansa System.

Chapter 2

Free Flight Routing Problem

2.1 Free-Flight

Free flight is a new concept being developed to take the place of the current air traffic management methods through the use of more advanced technology. True free flight eliminates the need for Air Traffic Control (ATC) operators by giving the responsibility to the pilot in command. With the aid of computer systems and/or ATC, pilots will be able to make more flight path decisions independently. As in most complex systems, distributed yet cooperative decision making is believed to be more efficient than the centralized control characterized by the current mode of air traffic management. In Free-Flight case, trajectory can be changed arbitrarily without following ATN (Air Traffic Network), by this change we can efficiently make use of weather condition and save more fuel. Overcrowded air network, increasing fuel cost and competition in airline market require use of the Free-Flight while more accurate navigation equipment and future availability of automatic distributed air traffic control system make it possible. Therefore, Free-Flight is the future of aircraft routing and it will bring more cost saving, energy efficiency, and less crowded on existing airways.

2.2 Flight Trajectory Optimization

Flight trajectory optimization is to find a path that can connect departure airport to the destination airport with minimum cost. This cost include fuel consumption, time cost (delay or arrive earlier than planned time), over flight cost and restricted airspace cost. Developing optimal aircraft trajectory not only enhances air traffic flow but also helps the industry to cope with the increasing fuel costs. Among those costs, fuel consumption

is most costly. Aircraft never fill up their tanks like we do in our cars, superfluous fuel is weight and weight is burning fuel, this is what we call fuel burns fuel. So predicting the required fuel as accurate as possible is crucial for minimizing the cost.

Therefore, finding a path with optimal fuel consumption is the main task, and this is a 4D optimization problem that contains 1D profile in vertical, 2D in horizontal, and 1D in time. We try to work separately in horizontal and vertical direction then find a way to combine this two profile. In this thesis, we focused on horizontal trajectory optimization to make fully use of weather condition, to burn as less fuel as possible. Depending on how the possible solution space is defined, a flight routing problem can be categorized into discrete flight routing and continuous flight routing problem. Theoretically, a free flight can follow any routes in the continuous three dimensional space instead of following a discretised grid network. Practically, the current Free-Flight setting widely used in the real world follows the discrete flight space. The horizontal two dimensional flight space can be discretised by a certain precision, which is usually to be 1 degree and not finer than 0.5 degree. This precision setting is due to two factors. Firstly, the weather data is available in a grid of 1.25° ; Secondly, the current flight navigation system has a limited precision of minimum half degree [35]. As the modern navigation system and weather data become more and more accurate, and the automatic air control systems become more and more intelligent, it makes free flight in a continuous space easier and easier to realise. So there is two way to consider this problem, continuous way and discrete way with very fine grids (1 to 1.25°). In our thesis, we consider to find an algorithm for optima fuel consumption trajectory in continuous case making fully use of the real wind data given by Lufthansa System, using Mixed Integer Programming. Moreover, we are looking forward in future to combine the vertical profile so that the trajectory is optimal in both vertical and horizontal direction with less fuel consumption and time consumption.

2.3 Previous Approach to the Problem

The earliest free flight trajectory optimization by MILP approach [2] simulated the fuel cost function as a second order function of speed. We know that fuel consumption depends on speed, but speed is not the only factor that determines the fuel consumption. Even the result is quite good but taking the fuel consumption as a second order function of speed is not enough, We know that, the heavier the aircraft weight, the higher aircraft fly, the more fuel it burns even for the same speed. So the fuel consumption depends on more factors than just speed. We got the fuel consumption data (the amount of fuel consumed to travel one unit distance) from Lufthansa System on specific grids, we

can simulate the fuel consumption function in a more accurate way. This will be more practical and more accurate. Another drawback is that the model only made on x-y plane which neglected the roundness of the earth.

The recent study on same topic is given by NASA research center and University of California [3]. The study develops a practical trajectory optimization algorithm for aircraft that approximately minimizes cost of time and fuel burn by combining the method for computing minimum-time routes in winds on multiple horizontal planes and aircraft fuel burn model for generating fuel-optimal vertical profiles. The result of the study is that flying wind optimal trajectories with at a single altitude reduces average fuel burn by 1% – 3%. Wind optimal trajectories reduce fuel burn and travel time relative to the flight plan route by up to 3% for domestic cargo flight. The horizontal trajectory is optimized by determining the heading angle that minimize the travel time in the presence of winds. There are two drawbacks in this research work: first, they assume the speed is constant, by which we already lost some probability to save the fuel since the fuel consumption depends on speed. Second is that in horizontal case, just consider the trajectory which consume less time is not enough to minimize the cost. As we know the fuel consumption depends on the aircraft speed and weight, faster the fly does not mean less fuel to burn, some time there will be a trajectory that consume more time than other but it is the most fuel saving.

There are also some other approaches used on Flight Routing Optimization problem by using direct application of Dijkstra' algorithm for the calculation of the cost-optimal paths on graphs with time dependent arc traversal costs. But there are drawbacks using this method on free flight. First we are still using network and it is not continuous any more, in this sense, it is still not a real Free-Flight unless we take the grid very fine until to the limit of the navigation system. The other drawback is that the airway networks as well as the Free-Flight graph become very problematic because of the time dependent wind which has huge impact on the arc cost. Dealing with those time dependence problem make the computation very time consuming. Another most common approach is dynamic programming, however, it does not provide a particular algorithm, the algorithm may vary for each case according to their own aim. Differential equation models also used for trajectory optimization by using aircraft motion equation. The problem is that when this method used, usually it is hard to deal with restricted airspace, and also in this case, some analytical functions are used to approximate the fuel consumption, and those functions always have big gap between real fuel consumption.

Most importantly, most of the study on flight trajectory optimization neglecting the roundness of earth and simply working on Cartesian coordinate, this is not very practical and accurate. Comparing to all this, in this thesis, we took into account the roundness of the earth and found a good projection that can project the earth to plane by preserving the distance. Another advantages is the data we use in our model, it is real world data on earth given in 3D (latitude, longitude and altitude) coordinates, in every 1.25 degree in latitude and longitude on whole earth and 13 flight level ranging from 85 (8500 feet) to 530. This makes the model more accurate and practical. Most importantly we do not use any analytic function to simulate the fuel consumption (the amount of fuel consumption to travel one unit distance) function, which depends on aircraft weight, aircraft true speed and altitude. We have the exact discrete fuel consumption data value on grids for specific weight and speed, then we use linear interpolation in two dimensions to approximate the fuel consumption for other speeds and weights. This is the main improvement in Free-Flight routing problem formulation, which can lead to more practical and accurate result.

Chapter 3

Optimization With Mixed Integer Linear Programming

In this chapter we give a brief introduction to the mathematical modelling method we use in the thesis to solve the Free-Flight routing problem. First we give simply introduction to Discrete Optimization, MILP programming and general solution technique used in MILP problem.

3.1 Discrete Optimization Problem

Definition 1. A minimization problem Π is given by a set of instances τ . Each instance $I \in \tau$ specifies

- a set F of feasible solutions for I .
- a cost function $c : F \rightarrow R$. Given an instance $I = (F, c) \in \tau$, the goal is to find a feasible solution $s \in F$ such that $c(s)$ is minimum. We call such a solution optimal solution of I .

Discrete optimization concentrate on optimization problems Π , where for every instance $I = (F, c)$, the set F of feasible solutions is discrete, i.e, F is a countable set.

More generally we can wrote in a form as following :

$$P : \text{ minimize}_{x \in X} f(x) \tag{3.1}$$

$$\text{subject to } x \in \Omega \tag{3.2}$$

Where X is the variable space (decision space), $f : X \rightarrow R \cup \{\pm\infty\}$ is called the objective function, and the set $\Omega \subset X$ is called the constraint region. We have different type of categorise as follows:

(1) Variable Type:

- Continuous variables: $X = R^n$
- Discrete variables: $X = Z^n$
- Binary variables: $X = \{0, 1\}$
- Mixed variables: $X = R^r \times Z^s \times \{0, 1\}^t$

(2) Constraint Type:

- Unconstrained: $\Omega = X$
- Constrained: $\Omega \neq X$

(3) Problem Type:

- Convex Programming: f is a convex function and Ω is a convex set.

Definition 2. The set $\Omega \in R^n$ is said to be convex if for every $x, y \in \Omega$ one has $[x, y] \in \Omega$ where $[x, y]$ denotes the line segment connecting x and y :

$$[x, y] = \{\lambda x + (1 - \lambda)y, 0 \leq \lambda \leq 1\} \quad (3.3)$$

Definition 3. The function $f : X \rightarrow R \cup \{\pm\infty\}$ is said to be convex if the set $\{(x, \mu) : f(x) \leq \mu\}$ is a convex set in R^{n+1} . In particular:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \quad (3.4)$$

for all $0 \leq \lambda \leq 1$ and $f(x), f(y)$ not both of them are infinite.

- Linear Programming

The minimization (or maximization) of a linear functional subject to a finite number of linear inequality :

$$f(x) = c^T x \quad c \in R^n \quad (3.5)$$

$$\Omega = \{x | Ax \leq b\} \quad (3.6)$$

Linear Programming is a special case of convex programming. In this case the constraint region Ω is called a polyhedral convex set. Polyhedral have a very special geometric structure.

- Quadratic programming: The objective function is a quadratic functions over a convex polyhedron:

$$f = \frac{1}{2}x^T Qx + b^T x + \alpha \quad (3.7)$$

- Nonlinear Programming: There objective function or the constraints are not linear.

3.2 Mixed Integer Linear Programming

We want to formulate the Free-Flight routing optimization with methods of discrete optimization. Especially in form of a Mixed Integer Linear Program. Mixed integer linear programming is a subset of the broader field of mathematical programming, it is an optimization method that combines continuous and discrete variables. A mixed integer linear programming problem (MILP, MIP) is of the form:

Let $A \in R^{m \times n}$, $B \in R^{m \times p}$, $b \in R^m$, $c \in R^n$, $d \in R^p$ $p \in 0, \dots, n$

$$\min c^T x + d^T y \quad (3.8)$$

$$Ax + By \leq b \quad (3.9)$$

$$x \in Z^n \quad (3.10)$$

$$y \in R^p \quad (3.11)$$

This is a linear programming, since all the constrain and objective functions are linear. It is called mixed integer programming because we have integer variable x and continuous variables y . If we find $(x, y) \in Z^n \times R^p$ such that it satisfies all constraints, we say it is a feasible solution. Between all feasible solution, if (\hat{x}, \hat{y}) can have the minimum value when put back to the objective function, it called global optimal solution.

3.3 solution techniques

There are alternative ways to model optimization problems as MIPs. Usually we end up with some models that even with small instances it takes too long time to reach the feasible solution, this is not practical. We always look for a model formulation that

can produce an efficient algorithm which is fast enough to be practically useful. Some models maybe smaller in terms of number of constraints and variables required, but may be more difficult to solve than larger models. In order to improve the efficiency of our model, we need to understand how MIP solvers work. There are mostly used MIP solvers such as CPLEX, SCIP, GUROBI. The solvers use a combination of branch-and-bound and cutting-plane techniques. These are the exact methods, this means that, if an optimal solution for MILP exists, they find it, otherwise, they prove that such a solution does not exist.

3.3.1 Brunch and bound

Branch-and -Bound algorithms are well known in discrete mathematics. The main idea is to divide the main problem to finite subproblems, and solve the subproblems, this is called branching. Branching usually done by relaxation procedure. There are many types of relaxation and this is the most common used relaxation called linear programming relaxation. The first step is solving continuous relaxation of MILP. As above we see the form of MILP, $x \in Z^n$, so the relaxation MILP is obtained by relaxing the integrality constraints on the x variables, then the problem becomes a LP problem

$$\min c^T x + d^T y \quad (3.12)$$

$$Ax + B \leq b \quad (3.13)$$

$$x \in R^n \quad (3.14)$$

$$y \in R^p \quad (3.15)$$

Then give a fractional value \hat{x}_i from the solution of LP. Afterwords the problem is divided into two subproblems, where the constraint $x_i \leq \hat{x}_i$ is added to the first subproblem and the the constraint $x_i \geq \lfloor \hat{x}_i \rfloor + 1$ is added to the second subproblem. Each of those new constraints represents a branching decision, because the partition of the problem in subproblems is represented with a tree structure, the BB tree. Each subproblem is represented as a node of the BB tree and has the following form:

$$\min c^T x + d^T y \quad (3.16)$$

$$Ax + By \leq b \quad (3.17)$$

$$x \in R^n \quad (3.18)$$

$$y \in R^p \quad (3.19)$$

$$x \leq l^k \quad (3.20)$$

$$x \geq u^k \quad (3.21)$$

where l^k and u^k are vectors defined so as to mathematically represent the branching decisions taken so far in the previous levels of the Branch-and-Bound tree (BB tree), the process is iterated for each node until the solution of the continuous relaxation of the subproblem is integer feasible or the continuous relaxation is infeasible or the lower bound of the value of subproblem is not smaller than the current best solution encountered so far. The following graph (3.1) illustrates the tree representation of this search process [34]:

Example 3.3.1. For example, if we want to solve the following MILP problem:

$$\text{Minimize } 4x_1 + 6x_2 \quad (3.22)$$

$$\text{subject to } 2x_1 + 2x_2 \leq 5 \quad (3.23)$$

$$x_1 - x_2 \geq 1 \quad (3.24)$$

$$x_1, x_2 \leq 0, \{x_1, x_2\} \in Z \quad (3.25)$$

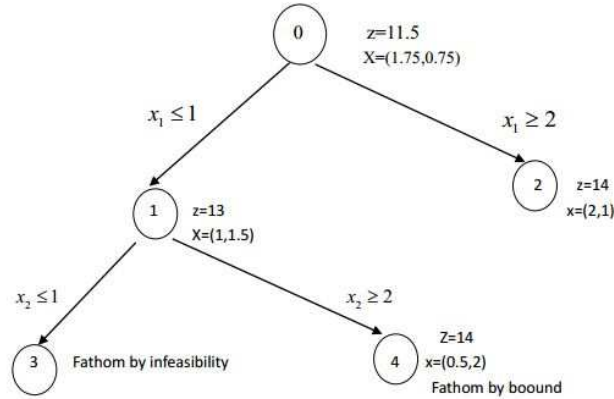


FIGURE 3.1: Branch-and-Bound tree

As we see from figure (3.1), the region 3 is empty, because for this branch, there is no solution can satisfy the constraint (3.24). Therefore, there is no integer solution for this branching and we can exclude this possibility out from solution space. Region 4 we still have to search for the integer solution by branching, but we can see that the objective function's value for relaxed problem is 14. We know that there is no integer solution which can give value to objective function less than 14 in this direction. However, we know there is integer solution which can give value to objective function 14 in region 2, so we also exclude region 4 from our solution space because, there is solution exists but it is not the optimal solution. We interested the most optimal solution but not on

every solution. Thus, region 4 is excluded because of the bounding, it called fathomed by bounding.

This produce can be done iteratively and so we get the well known brunch-and-bound tree. Summing up the idea described above we reach the following algorithm [4]:

Algorithm 1 BRUNCH-AND-BOUND

1. Let L be the list of the unsolved problems. Initialize L with the MILP which has to be solved. Set $U := +\infty$ as upper bound.
 2. Choose an unsolved problem X_j from the list L and delete it from L .
 3. Compute the lower bound b_{x_i} by solving the linear relaxation. Let \hat{x}_{X_j} be the optimal solution, so $b_{X_j} := c^T \hat{x}_{X_j}$.
 4. If $\hat{x}_{X_j} \in Z$, the problem X_j solved and we found solution of X_j ; if $U > b_{X_j}$, set $U := b_{X_j}$ and delete all subproblems X_i with $b_{X_i} \leq U$ from the list.
 5. If $\hat{x}_{X_j} \notin Z$, split problem X_j into subproblems and add them to the list L .
 6. Go to the step 2, until the list is empty.
-

3.3.2 Cutting plane

Cutting plane is a way for relaxation. As in the Branch-and Bound method, the LP relaxation is solved. Given the fractional LP solution (\hat{x}, \hat{y}) , a problem whose aim is finding a valid linear inequality that cuts off the (\hat{x}, \hat{y}) (not satisfied by (\hat{x}, \hat{y})). An inequality is valid for MILP problem if it is satisfied by any integer feasible solution of the problem. Once a valid inequality is founded it is added to the problem, which makes the LP relaxation tighter and the iterative addition of cuts might lead to an integer solution. In other words: if the solution \hat{x} satisfies $x \in Z^p \times R^n$ then the solution is found, if not then, there exists a valid inequality $\alpha^T x \leq \beta$ for the polyhedron P_{mip} that cuts of the \hat{x} . Therefore the valid inequality $\alpha^T x \leq \beta$ called a cutting plane. There are many different types of cuts have been studied, such as, geometry cuts, rounding cuts, lift-and-project cuts, split cuts, clique cuts. Usually different types of cuts are combined [4]. The complete algorithm for cutting plane is as follows [5]:

Algorithm 2 CUTTING PLANE

1. let $k := 0$ and LP^0 the linear programming relaxation of the mixed integer program
 2. Solve LP^k . Let \hat{x}^k is the optimal solution.
 3. if \hat{x}^k is in $Z^p \times R^{n-p}$ then stop, \hat{x}^k is the optimal solution of the mixed integer program.
 4. Otherwise, find a linear inequality, that is satisfied by all the feasible mixed integer points, but not by \hat{x}^k .
 5. Add this inequality to the LP^k to obtain LP^{k+1} .
 6. Increase k by one and go to step 2.
-

3.3.3 Branch-and-Cut

Cutting plane algorithm and branch-and-bound algorithm are usually combined in order to fasten the solution time of the mixed integer programs which called branch-and-cut algorithm. The idea is integrating the two methods, merging the advantages of both techniques. like in BB, at the root of node the LP relaxation is solved. If the solution is not integer feasible, a separation problem is solved and, in the case are found, they are added to the problem, otherwise a branching decision is performed. The algorithm is given as following [5]:

Algorithm 3 BRANCH-AND-CUT

1. Let L be the list of the unsolved problems. Initialise L with the mixed integer programming which has to be solved. Set $U := +\infty$ as upper bound.
 2. Choose an unsolved problem X_j from the list L and delete it from L .
 3. Compute the lower bound b_{X_j} by solving the linear relaxation. Let $x_{\hat{X}_j}$ be the optimal solution, so $b_{X_j} := c^T x_{\hat{X}_j}$.
 4. If $x_{\hat{X}_j} \in Z$, the problem X_j solved and we found solution of X_j ; if $U > b_{X_j}$, set $U := b_{X_j}$ and delete all subproblems X_i with $b_{X_i} \geq U$ from the list.
 5. If $x_{\hat{X}_j} \notin Z$, look for the cutting plans and add them to the linear relaxation.
 6. Go to the step 3. until no more violated inequalities can be found or violated inequalities have too little impact in improving the lower bound.
 7. Split problem X_i into subproblems and add them to the list L .
 8. Go to the step 2, until the list is empty.
-

Chapter 4

Approximating the Non-linear Functions

Since we want to formulate the problem as MIP, we know that most of the MIP solvers only work for linear constraint or work faster for the linear constraint than nonlinear ones such as CPLEX, SCIP (CPLEX can not handle with either nonlinear objective functions or constraints). In this chapter we will introduce how we approximate the nonlinear functions by piecewise linear functions or how to interpolate a function which is just given on discrete grid points as a black box, which is the case in our model for fuel consumption and wind data. There are three different approaches now used in different fields for linearising non-linear functions. The first one is called Special Ordered Sets of Types 2 also known as SOS Type 2. The second one is so called Convex Combination or Lambda Method. The last one is known as Incremental or Delta method.

4.1 1D Nonlinear Function Approximation by Piecewise Linear Functions

First, we present the piecewise linear approximation of a one dimensional non-linear functions.

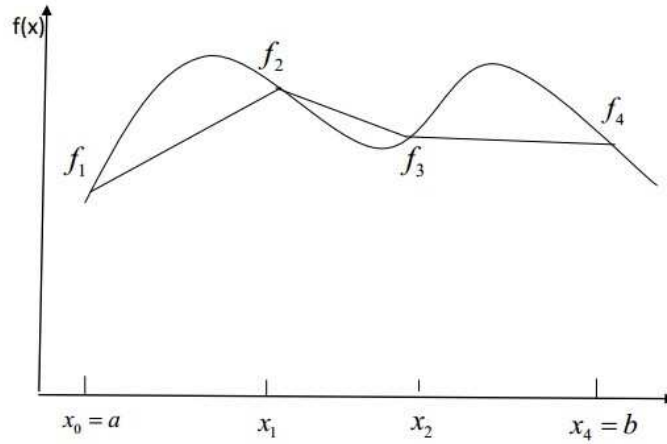


FIGURE 4.1: Approximation of non-linear function by piecewise linear functions

Approximating non-linear functions by piecewise linear functions:

Let's consider a non-linear function $f : D \rightarrow R$ where $D = [a, b]$. First we decompose the interval D to small subintervals $[x_i, x_{i+1}]$ where $x_i \in \{a = x_1 \leq x_2 \leq \dots x_k = b\}$, and we get the corresponding value of function f at grid points as $f(x_1), f(x_2), \dots, f(x_k)$. We want to approximate the non linear function f by the piecewise linear functions as shown in figure (4.1). In order to do that, we introduce a variable σ_i for each grid point x_i and take the convex combination as follows:

$$x = \sum_{i=1}^k x_i \sigma_i \quad (4.1)$$

$$\sum_{i=1}^k \sigma_i = 1 \quad (4.2)$$

$$\sigma_i \geq 0 \text{ for all } i = 1, 2, \dots, K \quad (4.3)$$

$$f(x) \approx \sum_{i=1}^k f(x_i) \sigma_i \quad (4.4)$$

The above constraints are not enough to guarantee that function f is linearised by means of line segment that shown in the figure (4.1). We need to add some other constraints, such as, at most two σ_i are positive(since we only chose those nearest points to approximate the value of $f(x)$, in 1D case, we have intervals, so we chose at most two neighbouring grids to approximate the function value) and if two are positive they must adjacent. This condition is called Special Ordered Set of Type2 condition, briefly SOS Type2 condition. We have different ways to handle the SOS conditions, that is why we

have SOS Type Polytopes, lambda method, and Delta method.

Now we define some basic concepts we used for linearization in order to understand the techniques to tighten the formulation of Mixed Integer Linear Programming. When we linearize our functions, we first decompose our domain into disjoint subdomains as we see above, in one dimension functions, it is interval, in two dimensional functions it can be triangle, rectangle or even some other shape. We can chose a appropriate way which will accelerate our algorithm, but every subdomain should be a polytop.

Definition 4. A subset $P \in R^n$ is called polyhedron if there exist a matrix $A \in R^{m \times n}$ and a vector $b \in R^m$ such that:

$$p = \{x \in R^n | Ax \leq b\} \quad (4.5)$$

Definition 5. A polyhedron is called Polytop if it is bounded, i.e, there exists a number $B \in R, B > 0$, with

$$p \subseteq \{x \in R^n | \|x\| \leq B\} \quad (4.6)$$

4.1.1 Linearization of Functions: SOS Polytopes (SOS Type 2)

Special ordered sets were introduced by Beale and Tomlin[6]. There are two types, sets of Type 1, are the set of variables of which not more than one member may be nonzero in the final solution. Sets of type 2 (SOS2) is a set of consecutive variables in which no more than two adjacent members may be non-zero in a feasible solution. All such sets are mutually exclusive of each other, the members are not subject to any other discrete conditions and each set is grouped together consecutively in the data. SOS Type2 were introduced to make it easier to find the global optimum solutions to problems containing piecewise linear approximations to a non-linear function of a single argument (as in classical Separable Programming). SOS Type 2 sets satisfy the SOS Type 2 condition implicitly during the branch-and-bound phase ([6],[7]).

The MILP formulations such that :

$$x = \sum_{i=1}^k x_i \sigma_i \quad (4.7)$$

$$\sum_{i=1}^k \sigma_i = 1 \quad (4.8)$$

$$\sigma_i \leq 0 \text{ for all } i = 1, 2, \dots, K \quad (4.9)$$

$$f(x) \approx \sum_{i=1}^k f(x_i) \sigma_i \quad (4.10)$$

$$\sigma_1, \sigma_2, \dots, \sigma_k \text{ are SOS2} \quad (4.11)$$

Remember that a set of consecutive variables $\lambda_1, \lambda_2, \dots, \lambda_n$ is SOS2, if at most two variables can be non zero and if so they must be adjacent. This condition is implicitly done by means of branching scheme, without introducing any additional (binary variables). Branching strategies for classical SOS 2 were developed in 1979 and can be found in [7]. Here we give a brief introduction about how it was done.

Branching for SOS conditions:

In the case of SOS, branching on sets of variables are more efficient than to branch on individual variables. Let us denote the set $s = \lambda_1, \lambda_2, \dots, \lambda_n$ is SOS2. If there are two positive variables λ_i, λ_j with $|i - j| > 1$, we chose one of them and divide the set into two parts, if we chose λ_j then we cut the problem into two subproblem, then we add the constrain $\sum_{i=1}^j \lambda_i = 1$ to the first subproblem and $\sum_{i=j}^n \lambda_i = 1$ to the second subproblem. For example, if we use the SOS2 in one dimensional function approximation, taking segment as a simplex, we have branching as following figure (4.2):

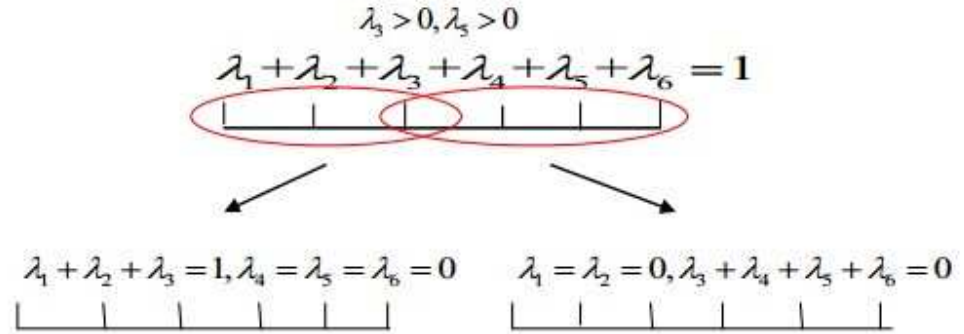


FIGURE 4.2: Branching on SOS2

4.1.2 Binary Linearization Model: Convex combination (lambda method)

In lambda method, the SOS Type 2 condition is modelled explicitly by introducing binary variables. Here we use the same notation as before, first again decompose the domain into small intervals by introducing grid points x_1, x_2, x_3, \dots and we have corresponding variables $\sigma_1, \sigma_2, \dots, \sigma_i, \dots, \sigma_k$, we introduce a binary variable λ_i for each segment $[x_i, x_{i+1}]$ $i = 1, 2, 3, \dots, k-1$ such that $\lambda_i = 1$ if the segment i is chosen, otherwise $\lambda_i = 0$. See the figure (4.3):

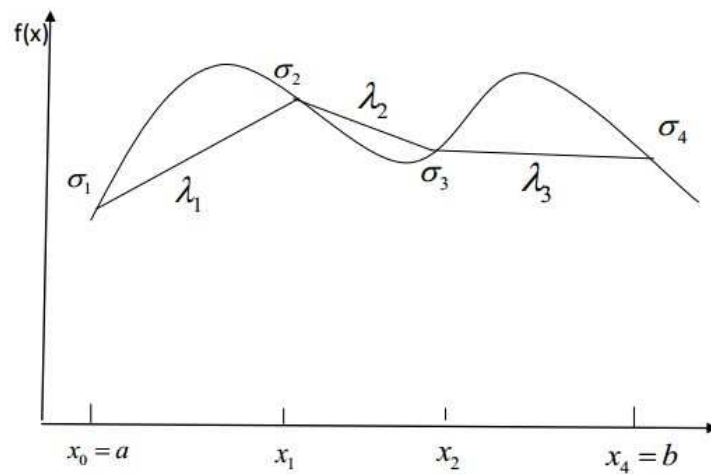


FIGURE 4.3: Lambda method for approximating nonlinear functions

Here we can formulate the approximation as a MIP problem:

$$\sum_{i=1}^{k-1} \lambda_i = 1 \quad (4.12)$$

$$\sigma_1 \leq \lambda_1 \quad (4.13)$$

$$\lambda_i \leq \sigma_{i-1} + \sigma_i \text{ for all } i = 2, 3, \dots, K-2 \quad (4.14)$$

$$\sigma_k \leq \lambda_{k-1} \quad (4.15)$$

$$\sum_{i=1}^k \sigma_i = 1 \quad (4.16)$$

$$\sigma_i \geq 0 \text{ for all } i = 1, 2, \dots, K \quad (4.17)$$

$$\sigma_i \in [0, 1] \quad (4.18)$$

$$x = \sum_{i=1}^k x_i \sigma_i \quad (4.19)$$

$$f(x) \approx \sum_{i=1}^k f(x_i) \sigma_i \quad (4.20)$$

The first constrain guarantee that only one segment is chosen, second, third and fourth constrain is actually the explicit form of SOS Type2 condition making sure that only those σ_i who is adjacent to the segment that chosen can be positive, all other σ_i are zero.

4.1.3 Binary Linearization Model: Incremental (Delta method)

With the same notation: see figure (4.4)

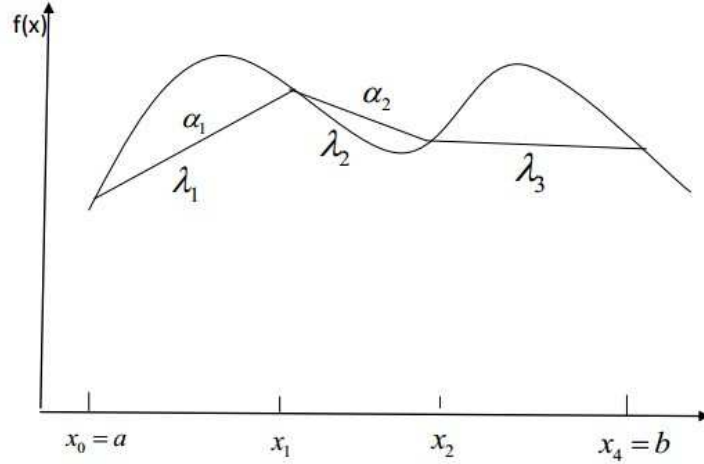


FIGURE 4.4: Delta method for approximation non-linear functions

we have decomposition on domain $D = [a, b]$, by grid points $x_1, x_2, x_3 \dots x_i \dots x_k$, and a continuous variable α_i $i = 1, 2 \dots k - 1$ for each segment $[x_i, x_{i+1}]$ which is having values on $[0, 1]$ and binary variables λ_i for $i = 1, 2 \dots k - 2$ corresponding to the segment $1, 2 \dots k - 2$. Therefore we have the MIP formulation for this approximation:

$$\alpha_{i+1} \leq \lambda_i \quad \text{for } i = 1, 2 \dots k - 2 \quad (4.21)$$

$$\lambda_i \leq \alpha_i \quad \text{for } i = 1, 2 \dots k - 2 \quad (4.22)$$

$$0 \leq \alpha_i \leq 1 \quad \text{for all } i = 1, 2 \dots K - 2 \quad (4.23)$$

$$x = a + \sum_{i=1}^{k-1} (x_{i+1} - x_i) \alpha_i \quad (4.24)$$

$$f(x) \approx f(a) + \sum_{i=1}^{k-1} (f(x_{i+1}) - f(x_i)) \alpha_i \quad (4.25)$$

The first and second constraints are called filling condition that assures that if an interval is chosen, then all intervals to its left must also be used completely.

4.2 2-Dimension Nonlinear Functions Approximation

Let us assume we have function $f: D \rightarrow R$ where $D \in R^2$. Here same as we did before for one dimension functions, we use decomposition of the domain D to finite set of

simplices and function is interpolated within each simplex. In our model we use uniform triangulation of domain. As shown in figure(4.5):

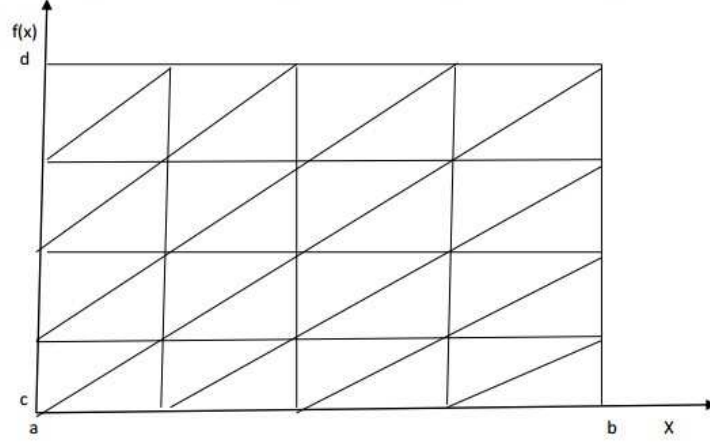


FIGURE 4.5: Approximation of nonlinear function by piecewise linear function

Now we linearise the function within a rectangle domain as follows. We have $D = [a, b] \times [c, d]$ we trivialize this domain by subdividing $[a, b]$ to n small intervals and $[c, d]$ to m small intervals, then we divide the each sub rectangle to two triangle as shown in the graph, therefore we have set of grid points (vertices of triangle) $A = \{v_1, v_2, v_3, \dots, v_{(m+1)(n+1)}\}$ and set of simplices $B = \{N_1, N_2, N_3, \dots, N_{2mn}\}$ with $2mn$ triangles. Same as in one dimensional case, here it need to fulfil the SOS condition stating that those variables σ_i on each vertexes, at most three of them can be positive and these three positive variables must belong to one triangle. This called SOS Type3 condition.

4.2.1 Linearization of 2D-Functions: SOS Polytopes (SOS Type 3)

For higher dimension functions (2D), the brunching concepts for SOS3 is a little complicated. We take uniform triangulation of the domain $D = [a, b] \times [c, d]$. (see the graph for 2D delta and Lambda method triangulation), Let $a = a_1 < a_2 < a_3 \dots < a_n = b, c = c_1 < c_2 < c_3 \dots < c_m = d$ is uniform subdivision for the domain D . We obtain mn grid points (a_i, b_j) , for $s \in \{2, 3 \dots m-1\}$ we separate the grids on the domain into to subsets such as $L = \{(i, j) \in \{1, 2 \dots n\} \times \{1, 2 \dots s\}\}$ and $R = \{(i, j) \in \{1, 2 \dots n\} \times \{s+1 \dots m\}\}$.

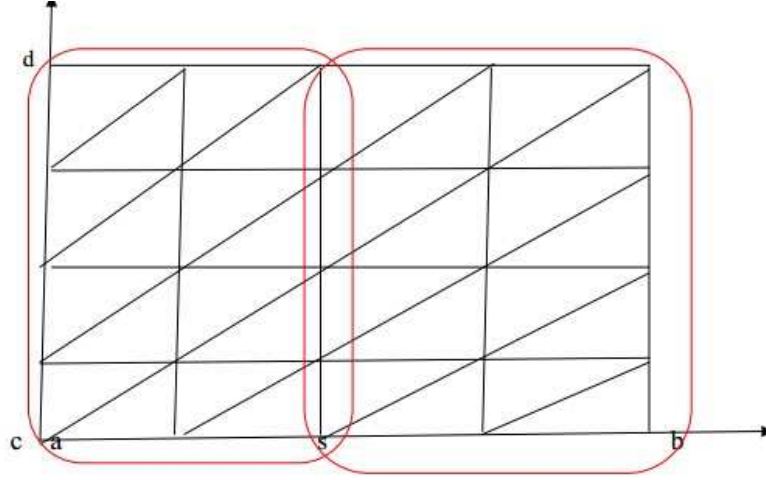


FIGURE 4.6: Branching on SOS3

This branching called vertical branching, see figure (4.6). Now our original problem convert to two subproblems with one more constrain. For the first subproblem, we add the equation,

$$\sum_{i \in L} \lambda_i = 1 \quad (4.26)$$

For the second subproblem, we add,

$$\sum_{i \in R} \lambda_i = 1 \quad (4.27)$$

We also can take horizontal branching, it is done in the same way with vertical branching, by doing it iteratively, at the end, we can observe that, if there is no further vertical or horizontal branchings, the positive $\lambda_{i,j}$ variables are restricted to exactly one rectangle, then we just need to branch on the two triangles which constitutes this last rectangle. For higher dimension functions, we need to branch in N direction, this is done by hyperplane branching[8]. In our case, we may use it for wind interpolation (2D now, but in future we would like to introduce the time dependence of wind, then will be 3D), fuel range function (2D in our model, if we consider horizontal direction and assume the optimal altitude is known) and also for approximation quadratic function of speed by piecewise linear functions.

4.2.2 Linearization of 2D-Functions: Lambda method

Lambda method as we seen before uses additional binary variables to explicitly model the SOS conditions. We introduce the binary variables λ_j for each simplex $N_j \in B$, σ_i

for each vertex $i \in A$, lets denote $S = \{1, 2, \dots, mn\}$ is the set of indices of the binary variables (indices of triangles), Then the MILP formulation for the approximation is following:

$$\sum_{j \in S} \lambda_j = 1 \quad (4.28)$$

$$\sum_{i \in A} \sigma_i = 1 \quad (4.29)$$

$$\sigma_i \leq \sum_{j: i \in N^j} \lambda_j \quad \text{for all } i \in A \quad (4.30)$$

$$0 \leq \sigma_i \leq 1 \quad \text{for all } i \in A \quad (4.31)$$

$$\lambda_j \in \{0, 1\} \quad \text{for all } j \in S \quad (4.32)$$

$$x = \sum_{i=1}^k x_i \sigma_i \quad (4.33)$$

$$f(x) \approx \sum_{i=1}^k f(x_i) \sigma_i \quad (4.34)$$

The first constrain guarantee that only one simplex is chosen and second constrain modes SOS Type3 condition.

4.2.3 Linearization of 2D-Functions: Generalized Delta method

In one dimensional case, we have filling condition that easily fulfilled and ordering is automatically done since the simplices consist of sequence of segment. But in higher dimension case it is quite complicated since it require the ordering of simplices. There are some kind ordering methods are talked in literatures [9]. The ordering is done by making assumption for simplices which is called ordering assumption:

Assume that the simplices $B = \{N_1, N_2, N_3, \dots, N_{2mn}\}$ are ordered in a way such that: $N_j \cap N_{j+1} \neq \emptyset$, and for each simplex N_j we can label it's vertices (v_j^0, v_j^1, v_j^2) such that $v_{j-1}^2 = v_j^0$ (type 3, triangulation) holds for all $j = 1, 2, \dots, mn$. See figure (4.7), (4.8).

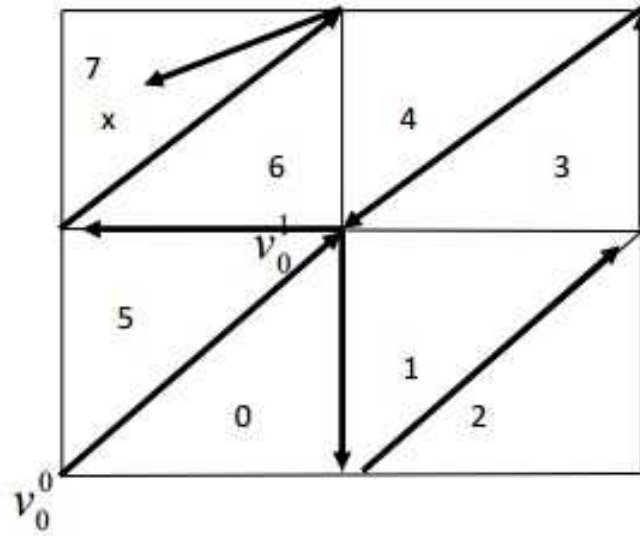


FIGURE 4.7: Triangulation and Special Ordering

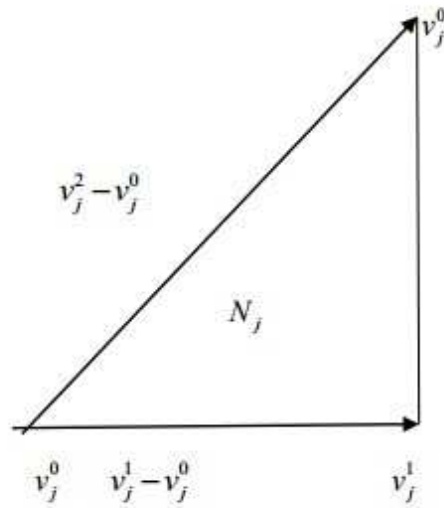


FIGURE 4.8: Simplex

As shown in the figure(4.8), if we chose one simplex $N_j \in B$ and assume it fulfils ordering assumption, it is obvious that every point $x \in N_j$ can be written as:

$$x = v_j^0 + \sum_{i=1}^3 (v_j^i - v_j^0) \sigma_j^i \quad (4.35)$$

where $\sum_{i=1}^2 \sigma_j^i \leq 1$ and $\sigma_j^i \geq 0$

The point v_j^i can be written as follows:

$$v_j^0 = v_{j-1}^0 + (v_{j-1}^2 - v_{j-1}^0) \quad (4.36)$$

This is because we assume that the simplices satisfies the ordering assumption which is states that $v_{j-1}^2 = v_j^0$. By iterating this procedure we came up with the following equation:

$$v_j^0 = v_1^0 + \sum_{i=1}^{j-1} (v_i^2 - v_i^0) = v_1^0 + \sum_{i=1}^{j-1} (v_{i+1}^0 - v_i^0) \quad (4.37)$$

This showing that any point in simplex N_j can be presented by v_1^0 plus vectors from v_j^0 to v_{j+1}^0 , $j = 1, 2, \dots, j-1$ and plus a conical combination of the $v_j^i - v_j^0$. By all this analysis we reach the MILP formulation for of the approximation as follows:

$$x = v_1^0 + \sum_{i=1}^{2mn} \sum_{j=1}^2 (v_i^j - v_i^0) \sigma_i^j \quad (4.38)$$

$$\sum_{i=1}^2 \sigma_i^j \leq 1 \text{ for all } i = 1, 2 \dots mn \quad (4.39)$$

$$\sigma_i^j \geq 0 \text{ for all } j = 1, 2 : i = 1, 2 \dots mn \quad (4.40)$$

$$f(x) \approx f(v_1^0) + \sum_{i=1}^{2mn} \sum_{j=1}^2 (f(v_i^j) - f(v_i^0)) \sigma_i^j \quad (4.41)$$

Remember that we had filling condition for Delta method in one dimension stating that if an segment is chosen then all segments to its left should be completely used. In two dimension case we have triangles instead of segments, and those triangles should also satisfy filling condition such that if any triangle are chosen then all those triangles ordered before it should be completely used. In other word, if any variables σ_i^j is positive then all variables $\sigma_{i-1}^2 = 1$ for all $i = 1, 2 \dots mn$ In order to model this condition we again introduce extra binary variables λ_j associate with simplex N^i for $i = 1, 2 \dots mn-1$. Then we have the following constrains:

$$\sum_{j=1}^2 \sigma_{+i1}^j \leq \lambda_i \text{ for all } i = 1, 2, \dots, mn - 1 \quad (4.42)$$

$$\lambda_i \leq \sigma_i^2 \text{ for all } i = 1, 2, \dots, d - 1. \quad (4.43)$$

Above two constrain if σ_{i+1}^i positive this force λ_i is one and then σ_i also one, using iteratively then it make sure that if N_i simplex is chosen, then all simplices ordered before is used completely.

Chapter 5

Model Formulation with MILP

5.1 Problem Description and Data Analysis

As we defined in chapter 1, we want to find an optimal route for Free-Flight, which is not based on a network. Under the allowance of the navigation technique and passenger's comfort, the direction can be changed at any time during the trip to make fully use of the wind. Therefore, given the coordinate (latitude, longitude) of two points on the earth as departure and arrival airport, we use the weather data to find an algorithm which can produce the optimal route on horizontal direction with minimum cost. For cost function, we mainly consider fuel cost.

A typical aircraft trajectory consists of an initial climb, a steady-state cruise, and a final descent. Here the aircraft performance is optimized for the cruise phase only. The fuel and time saving for initial climb and descent are small compared to the cruise. We need additional traffic constraints to model these stages, which make the model more complicated but bring almost no contribution for fuel saving. Therefore, we escape the climb and descent stage, just consider the cruise stage.

5.1.1 Weather Data

The weather data from Lufthansa System is given with (latitude,longitude, altitude) coordinate at each 1.25 degrees in horizontal direction and on 13 flight levels from flight level 50 (5000 feet) to flight level 830 in vertical direction. Wind is given in north and east component, the figure (5.1) shows the direction of two component. The U stands for the east wind component , V stands for north wind component.

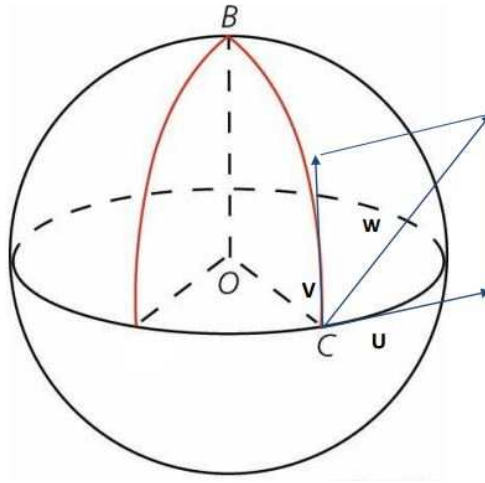


FIGURE 5.1: north and east wind component

Here is a part of original data form by Lufthansa System shown in figure (5.2). The weather data is measured at flight level 10 (1000 feet), on 8th of October, 2013 at time 00:00. The U stands for the east wind component , V stands for north wind component, and T stands for temperature. N9000 represent altitude 90 degree (north pole), E0000 and E00115 represent longitude 0 degree(east hemisphere), longitude 1 degree 15 minutes (east hemisphere) respectively. The figure (5.2) shows the part of original weather data given by Lufthansa System at north pole from longitude 0 to longitude 31.25 degree.

```

Observation date: 08-Oct-2013
Observation time: 00:00:00
Prognosis date: 08-Oct-2013
Prognosis time: 12:00:00
Level: Height: 16221
          hPa: 100
          FL: 530
          Level_id: 10

Available Parameter: U,V,T,R
#####
Coordinates      U      V      T      R
N9000
E00000          2.3     -3.1    -52.24    ---
E00115          2.23    -3.14    -52.24    ---
E00230          2.16    -3.19    -52.24    ---
E00345          2.09    -3.24    -52.24    ---
E00500          2.02    -3.28    -52.24    ---
E00615          1.94    -3.33    -52.24    ---
E00730          1.87    -3.37    -52.24    ---
E00845          1.8     -3.41    -52.24    ---
E01000          1.72    -3.45    -52.24    ---
E01115          1.65    -3.48    -52.24    ---
E01230          1.57    -3.52    -52.24    ---
E01345          1.49    -3.55    -52.24    ---
E01500          1.41    -3.58    -52.24    ---
E01615          1.34    -3.61    -52.24    ---
E01730          1.26    -3.64    -52.24    ---
E01845          1.18    -3.67    -52.24    ---
E02000          1.1     -3.69    -52.24    ---
E02115          1.02    -3.72    -52.24    ---
E02230          0.93    -3.74    -52.24    ---
E02345          0.85    -3.76    -52.24    ---
E02500          0.77    -3.78    -52.24    ---
E02615          0.69    -3.79    -52.24    ---
E02730          0.6     -3.81    -52.24    ---
E02845          0.52    -3.82    -52.24    ---
E03000          0.44    -3.83    -52.24    ---
E03115          0.35    -3.84    -52.24    ---

```

FIGURE 5.2: wind data given by Luthansa Airline

We see from figure (5.2), at north pole, we have different magnitude of wind at both east and north component even north pole is only one point. This is because of the direction. The wind is the same but, the direction is different at different longitude on the north pole. Here we give the formula to calculate the wind at north pole for each different direction.

(1) if the wind at altitude 90 degree (north)longitude 0 degree (E0,N90) is given as following:

north wind component(V) at (N900,E00): $V_1 = -3.1$

east wind component(U) at (N900,E00): $U_1 = 2.3$

at any other point A on north pole with longitude θ (degree) , the wind north and east wind component (V_2, U_2) is given by:

$$V_2 = \sin(\arctan(\frac{V_1}{U_1}) - \theta) \sqrt{(V_1^2 + U_1^2)} \quad (5.1)$$

$$U_2 = \cos(\arctan(\frac{V_1}{U_1}) - \theta) \sqrt{(V_1^2 + U_1^2)} \quad (5.2)$$

For example at E03115:

$$\theta = 31.25 \quad (5.3)$$

$$v_2 = \sin(-36.57 + 31.25) \cdot 3.86010 = -3.84 \quad (5.4)$$

$$u_2 = \cos(-36.57 + 31.25) \cdot 3.86010 = 0.35 \quad (5.5)$$

The direction of the north and east component wind is given by the following rule

For north component:

$$- \text{ if } \theta > 90 + \arctan(\frac{V_1}{U_1}) \quad (5.6)$$

$$+ \text{ if } \theta < 90 + \arctan(\frac{V_1}{U_1}) \quad (5.7)$$

For east component:

$$- \text{ if } 36.57 < \theta < 180 + \arctan(\frac{V_1}{U_1}) \quad (5.8)$$

$$+ \text{ if } 0 < \theta < \arctan(\frac{V_1}{U_1}) \text{ or } \theta > 180 + \arctan(\frac{V_1}{U_1}) \quad (5.9)$$

So for the point (N000,E0315) the north wind component is -3.84, and east component is 0.35.

The impact of wind is strong on aircraft over north Atlantic ocean if we fly from Europe to the United states, the following figures (5.3), (5.4) shows the wind data visualization at north Atlantic ocean and Europe:

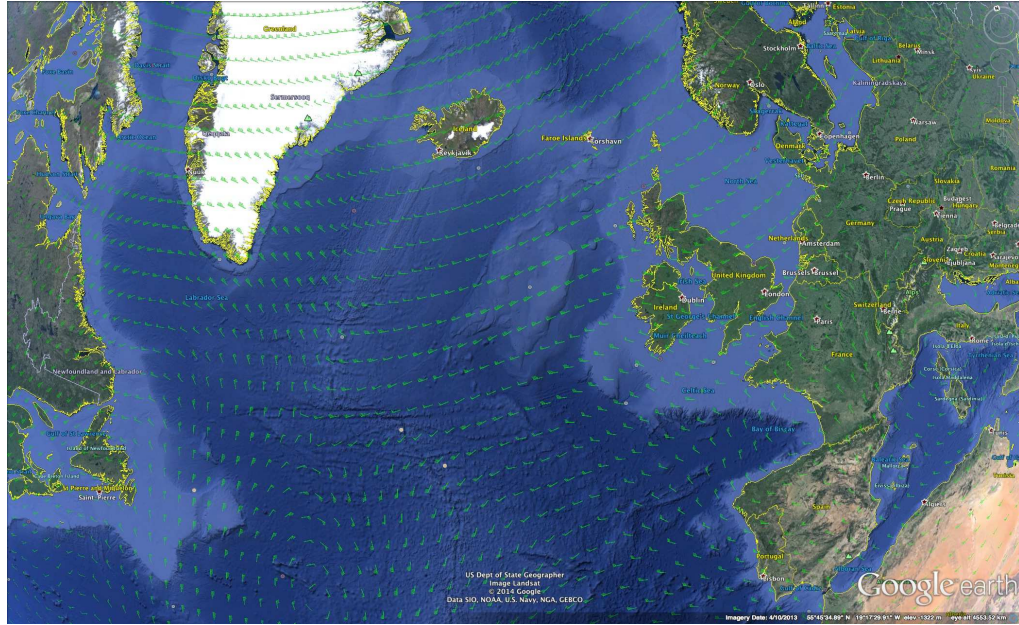


FIGURE 5.3: Wind bars in North Atlantic Ocean at flight level 390

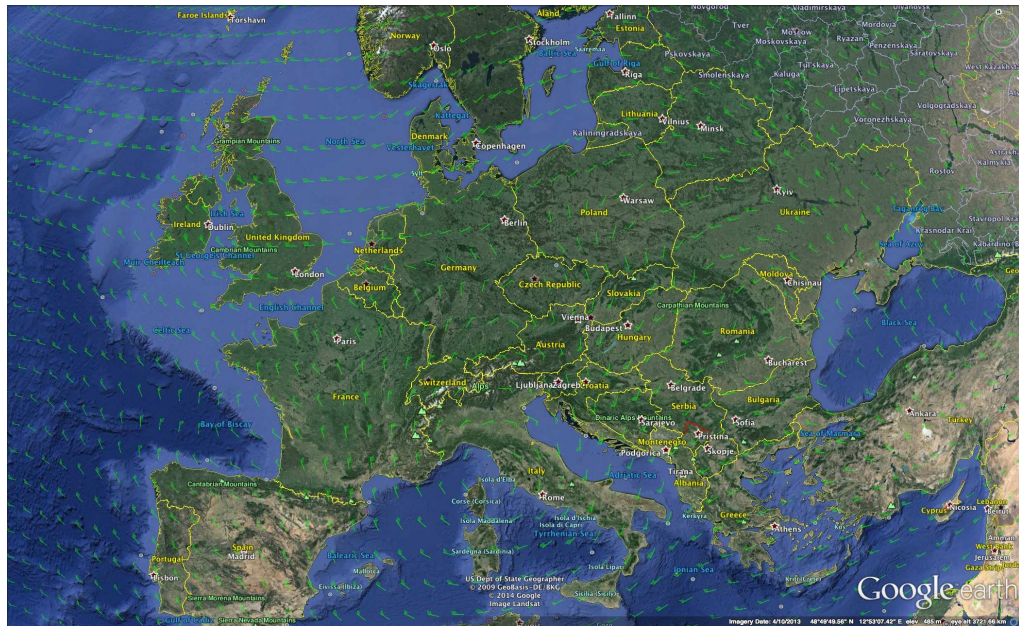


FIGURE 5.4: wind bars in Europe at flight level 390

5.1.2 Fuel Consumption Data

The fuel consumption (the total amount of fuel consumed to travel one unit distance (here is one nautical mile)) data is given as a function of aircraft true speed, weight,

and altitude. Here is the original fuel consumption data on flight level 360, on 27 speed grids and 15 weight:

#	speed	index, weight	index, unit	distance	fuel consumption
0.4	56500.0	7.394808844191378			
0.4	58000.0	7.5820759724012445			
0.4	59500.0	7.770611547128759			
0.4	61000.0	7.958615200955033			
0.4	62500.0	8.144648965629582			
0.4	64000.0	8.329862557267806			
0.4	65500.0	8.517887563884157			
0.4	67000.0	8.702462796971544			
0.4	68500.0	8.89125989152663			
0.4	70000.0	9.074410163339383			
0.4	71500.0	9.262689885142645			
0.4	73000.0	9.444654325651682			
0.4	74500.0	9.63298333493883			
0.4	76000.0	9.954210631096954			
0.4	77500.0	10.32204789430223			
0.44	56500.0	6.565557087518876			
0.44	58000.0	6.716818914562063			
0.44	59500.0	6.911327666044647			
0.44	61000.0	7.037297677691766			
0.44	62500.0	7.233796296296296			
0.44	64000.0	7.3675679658144855			
0.44	65500.0	7.557436517533253			
0.44	67000.0	7.700600646850454			
0.44	68500.0	7.883326763894364			
0.44	70000.0	8.028904054596548			
0.44	71500.0	8.210854749979472			
0.44	73000.0	8.353521009105338			
0.44	74500.0	8.540438978563499			
0.44	76000.0	8.673026886383347			
0.44	77500.0	8.87154009936125			

FIGURE 5.5: Unit distance fuel consumption data

Fuel consumption changes as altitude, aircraft true speed and weight changes (figure (5.5)). We want to see how the aircraft true speed and weight influence the fuel consumption function (in our work, we assume the altitude is fixed, later we try to combine vertical and horizontal optimization). Here is the fuel consumption surface for A320 type aircraft in figure (5.6), at altitude level 360, as a function of weight and speed:

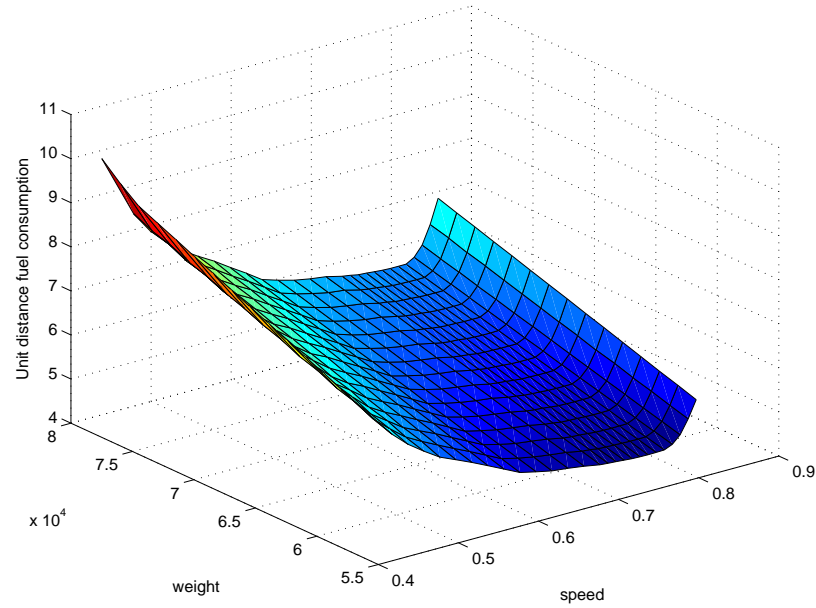


FIGURE 5.6: Fuel consumption surface for 27 speed grids and 15 weight grids

As we can see, it is not a regular function, the function value is given on grids, and we are not able to simulate it by some analytical functions. This fuel consumption function is like a black box, only have values for some specific discrete points. Given aircraft true speed, weight, altitude on grids, returns a corresponding value which stands for the amount of the fuel consumed for travelling one unit of distance. But we don't know the what is the value of the function for other weight, speed and altitude. From figure (5.6),(5.7), it can be seen that there is an optimal speed for this fuel consumption function, whatever the weight of the aircraft, the fuel consumption always minimum at this speed (when there is no wind).

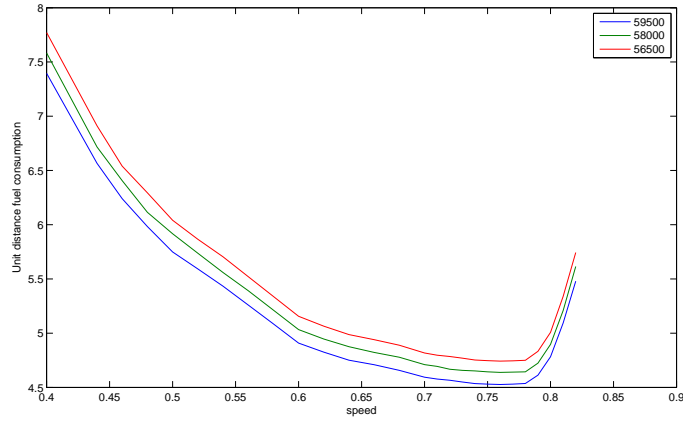


FIGURE 5.7: Fuel consumption as a function of speed for 3 different weight

We simulate the fuel consumption function on flight level 360 for 3 different weights and it was shown in figure (5.7) such that flying at speed 0.76 Mach number, always consume less fuel for one unit length no matter how the aircraft weight change (when there is no wind influence).

5.2 Aircraft Motion Equation

We can formulate aircraft motion in different coordinate system in different way. As figure (5.8) shows, the true distance aircraft travelled is the combination effect of aircraft true velocity vector (represented by V in figure) and wind velocity vector (w) effect.

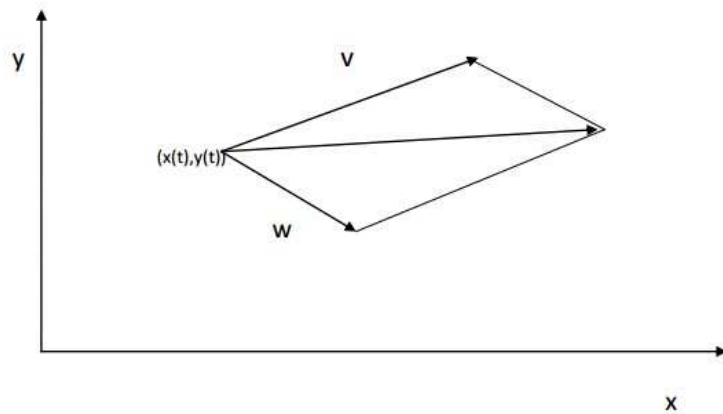


FIGURE 5.8: Aircraft true speed+ wind effect

If we simply consider the aircraft motion equation on 2D X-Y plane:

$$\dot{x}(t) = w_x(t, x(t), y(t)) + V \cos \gamma(t) \quad (5.10)$$

$$\dot{y}(t) = w_y(t, x(t), y(t)) + V \sin \gamma(t) \quad (5.11)$$

Where $\gamma(t)$ is the flight angle, the angle from positive x axis to the aircraft true velocity direction. (from x direction)

On Earth's surface:

$$\dot{\phi} = \frac{u(\phi, \theta, \alpha) + V \cos \gamma}{(R + a) \cos \theta} \quad (5.12)$$

$$\dot{\theta} = \frac{v(\phi, \theta, \alpha) + V \sin \gamma}{R + a} \quad (5.13)$$

Where ϕ is longitude, θ is latitude, γ is heading angle, and R is Earth's radius. The east component of the wind velocity is $u(\phi, \theta, \alpha)$ and the north-component of the wind is $v(\phi, \theta, \alpha)$.

Before formulating the problem into mathematical model, we need to think about which coordinate system would be more convenient, accurate and near to the practical usage. Working on 2D (Cartesian coordinate) would make the computation more efficient and fast, but the problem is that by working on 2D, we are neglecting the roundness of the earth. This kind of model only work for short trip near the equator, for long trip it is quite inaccurate. 3D coordinate (spherical coordinate) would be ideal for accuracy, however, in this case, we have wind data at each 1.25 degree longitude and 1.25 degree latitude grids, with east and north component. When mapped to 3D, the model becomes quite complicated (will see later) that involve trigonometric functions which is hard for MIP solvers to solve, at least there is no efficient MIP solvers for this kind of problem right now. At the end, after analysing all formulation possibility, accuracy, and solvers efficiency, we came up with the idea that, through pre-processing, we escape all complicated calculation for MIP, make the model more efficient and fast. This means we first map the solution space from earth surface to on X-Y plane in such a way that it is more accurate and practical. Change the coordinate from latitude, longitude to Cartesian coordinate, then use MIP solver to solve the model.

Therefore, first we make a model that works efficiently on 2D x-y plane. Then we adapt our model to the earth surface.

5.3 Mathematical Model

In this section we introduce the mathematical formulation of the problem with mixed integer programming. Formulating a real-world problem as a MIP sometimes is quite challenging, since we need to find a way to present the correctly and as close to practical issue as possible. At the same time we need to guarantee the solvability of the problem using various techniques. Thus, we keep updating the model during formulation, especially for those non-linear equations in the model, we need to find a way that can be presented in a linear way, which can speed up the algorithm.

Flight routing optimization with MIP is done in two steps. First, the problem has to be presented completely as a mixed integer programming. Second, is to solve the model using a linear programming based branch and cut approach.

First let us give a mathematical model formulation for Free-Flight routing problem. For model formulation, we need the following variables and parameters:

Variables:

- In order to describe the position of the aircraft over time, we introduce the continuous variable $(x(t), y(t))$ describing the location of aircraft as a function of time.
- Velocity of aircraft (V_x, V_y) (aircraft true speed) is presented in x,y component as a continuous function of t. Once the velocity become zero, the flight is considered terminated.
- Weight of the aircraft $W(t)$ can be written as a function of time t.
- fuel consumption function represented as $f(V, W, a)$ a function of aircraft true speed, weight, and altitude.
- Binary variables λ which having value on 0,1 introduced for descritization.

Parameters:

- The wind field $(w^x(x_t, y_t), w^y(x_t, y_t))$ is given at $(x(t), y(t))$ as a function of t.
- Fuel consumption $f(a_t, V_t, W_t)$ is a function of altitude, aircraft true speed, and weight.

5.3.1 MILP Formulation for Free-Flight

In order to formulate the aircraft motion as MIP mode, we discretize the time T which is the scheduled time for trip as $0 = t_0 < t_1 < t_2 < t_3 < \dots < t_N = T$.

Variables:

- x_i : x coordinate of the aircraft position at time t_i
- y_i : y coordinate of the aircraft position at time t_i
- w_i^x : wind velocity x -component at (x_i, y_i)
- w_i^y : wind velocity y -component at (x_i, y_i)
- V_i^x : aircraft speed x -component at time t_i
- V_i^y : aircraft speed y -component at time t_i
- W_i : aircraft weight at time t_i
- f_i : the amount of fuel consumed on segment i

- l_i : the distance travelled by aircraft within time Δt_i

- Δt_i : the time spend on segment i which is $\Delta t = t_{i+1} - t_i$
(denote segment i by: $[(x_i, y_i), (x_{i+1}, y_{i+1})]$)

- $Fr(V_i, W_i)$: fuel consumption for one unit length when aircraft speed is V_i and weight is W_i .

parameters:

- (x_0, y_0) : starting point
- (x_N, y_N) : destination point
- EW : dry weight of aircraft(empty weight of aircraft+load weight+extra fuel for safety concern)
- T : duration of the trip.

We want to minimize the cost. In our model the cost function is mainly considered the fuel consumption for the whole trip. Few hours before starting the trip, some amount of fuel is loaded on the plane which is estimated would be enough for finishing the whole trip. Here we want to minimize this fuel amount loaded to the aircraft at the starting of trip, so that we can finish the trip with as less fuel as possible and also guarantee that we will not have the risk of out of fuel in mid flight.

Model 5.1:

Objective function :

$$\min (W_0 - W_N) \quad (5.14)$$

Constraints:

$$V_i^2 = (V_i^x)^2 + (V_i^y)^2 \quad \text{for all } i = 1, 2, \dots, N \quad (5.15)$$

$$x_{i+1} = x_i + \Delta t_i \cdot (V_i^x + w_i^x) \quad (5.16)$$

$$y_{i+1} = y_i + \Delta t_i \cdot (V_i^y + w_i^y) \quad (5.17)$$

$$w_i^x = w^x(x_i, y_i) \quad (5.18)$$

$$w_i^y = w^y(x_i, y_i) \quad (5.19)$$

$$l_i = V_i \Delta t_i \quad (5.20)$$

$$f_i = l_i \cdot Fr_i(V_i, W_i) \quad (5.21)$$

$$W_N = EW \quad (5.22)$$

$$\Delta t_i = t_{i+1} - t_i \quad (5.23)$$

$$(x_0, y_0) = \text{departure location} \quad (5.24)$$

$$(x_N, y_N) = \text{arrival location} \quad (5.25)$$

Before directly going to the model considering the roundness of earth, and work on real world data, we tried to test our model on usual X-Y plane (2D Cartesian coordinate), with artificial wind. Later we adapt the model to real world case by considering the roundness of the earth, and work on real world case using the weather data and fuel consumption data from Lufthansa System.

5.3.2 2D X-Y Plane Model

As we can see in model 5.1, there are non-linearities in the model such as constraints (5.26), and (5.21). In order to linearise (5.26), we can think of using fixed time step or fixed segment length.

In our model we choose to use fixed time step $\Delta t = T/N = t_{i+1} - t_i$, $i = 1, 2, 3 \dots N$ for each segment. We introduce relaxation factors on the arriving point allowing that we can arrive in a small region include the arrival point with a small deviation. This relaxation is introduced in order make sure the feasible solution exists. As we only consider cruise stage, if we have small deviation at the end of the cruise stage, we still have chance to arrive the exact point. So if the deviation is small, we still can make sure to arrive the exact airport within scheduled time. And by adding this relaxation to the objective function we can keep it as small as possible. We also introduced a small weight in the objective function for relaxation term. By doing this, we emphasise that we want to minimize the fuel, minimizing deviation is less important than minimize the fuel consumption because we usually give a small range of magnitude for relaxation term when we introduce it to model.

Model 5.2

- The objective function:

$$\min (W_0 - W_N + 0.01 \cdot (\sigma_x + \sigma_y)) \quad (5.26)$$

This is the magnitude of fuel consumed during whole trip.

- Constraints:

$$V_i^2 = (V_i^x)^2 + (V_i^y)^2 \quad (5.27)$$

$$x_{i+1} = x_i + \Delta t \cdot (V_i^x + w_i^x) \quad (5.28)$$

$$y_{i+1} = y_i + \Delta t \cdot (V_i^y + w_i^y) \quad (5.29)$$

$$w_i^x = w^x(x_i, y_i) \quad (5.30)$$

$$w_i^y = w^y(x_i, y_i) \quad (5.31)$$

$$l_i = V_i \cdot \Delta t \text{ for all } i = 1, 2 \dots N - 1 \quad (5.32)$$

$$f_i = l_i \cdot Fr_i(V_i, W_i) \text{ for all } i = 1, 2 \dots N - 1 \quad (5.33)$$

$$W_N = EW \quad (5.34)$$

$$x_0 = D_x \quad (5.35)$$

$$x_N = A_x + \sigma_x \quad (5.36)$$

$$y_0 = D_y; \quad (5.37)$$

$$y_N = A_y + \sigma_y \quad (5.38)$$

$$B_x \geq \sigma_x \geq 0 \quad (5.39)$$

$$B_y \geq \sigma_y \geq 0 \quad (5.40)$$

Where (D_x, D_y) is the coordinate of departure airport, (A_x, A_y) is arrival airport location. B_x , and B_y is the bound for relaxation factor.

5.3.3 Trip 1: Implementation in SCIP with lambda Method

We give a trip starting from (0,0) to $(0.9 \times 2000, 0.9 \times 2000)$ (nautical miles) and allow small deviation. Whole trip takes 5 hours and we fix each time step as 0.5 hour. We introduce the wind artificially (the wind generating method is used same method in [2]), which is given by x and y component and magnitude ranging from 0.03016 to 0.087 (Mach number) in x direction and -0.07221 (Mach number) to 0.043 (Mach number) in y direction. We chose the aircraft type A320. This is a small type of aircraft which can fly continuously at most 7 hours, and allowed speed range is from 0.4 Mach number to 0.82 Mach number.

Since it is free flight, the trajectory is continuous, means it is not based on grid, each time can reach any point in the solution space under its capability. Therefore, we need continuous wind and fuel data on the solution space. However, the fuel data is given on *altitude* \times *aircraft* *true speed* \times *weight* grid while wind given on a grid *altitude* \times *latitude* \times *longitude*, so we need to do the wind and fuel interpolation by triangulation such that we can get a wind fuel consumption value at any point on the solution space.

Wind data interpolation by By lambda method:

the wind is introduced at grid points $\{0, 200, 400, 600, \dots, 2000\}$ in both x and y direction. Here we use 2D lambda method to approximate the wind data. Let us denote $A = \{v_1, v_2, \dots, v_n\}$ the set of the grid points (vertices of triangles) and A_1 the index set of the points. $B = \{N_1, N_2, \dots, N_m\}$ set of triangles, $T_1 = \{t_1, t_2, \dots, t_N\}$ the time grids, in our case, it is a uniform grid. We define a variable $\alpha(i, j)$ where $(i, j) \in A_1$ taking value on $[0, 1]$ for each grid points(vertices of the triangle) and a binary variable λ_i where

$i \in B1 = \{1, 2, 3, \dots, m\}$ for each triangle. Then we have the following approximation:

$$\sum_{i \in B1} \lambda_i^n = 1, \text{ for all } n \in T \quad (5.41)$$

$$\sum_{(i,j) \in A1} \alpha_{(i,j)}^n = 1 \text{ for all } n \in T \quad (5.42)$$

$$x(n) = \sum_{(i,j) \in A1} x_i \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.43)$$

$$y(n) = \sum_{(i,j) \in A1} y_i \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.44)$$

$$w^x(n) = \sum_{(i,j) \in A1} w^x(i,j) \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.45)$$

$$w^y(n) = \sum_{(i,j) \in A1} w^y(i,j) \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.46)$$

$$\lambda_i^n \leq \sum_{\{l \in B1 | (i,j) \in A1\}} a_{(i,j)}^n \text{ for all } (i,j) \in A1 \text{ } n \in T \quad (5.47)$$

$$0 \leq a_{(i,j)}^n \leq 1 \text{ for all } (i,j) \in A1 \text{ } n \in T \quad (5.48)$$

The wind field:

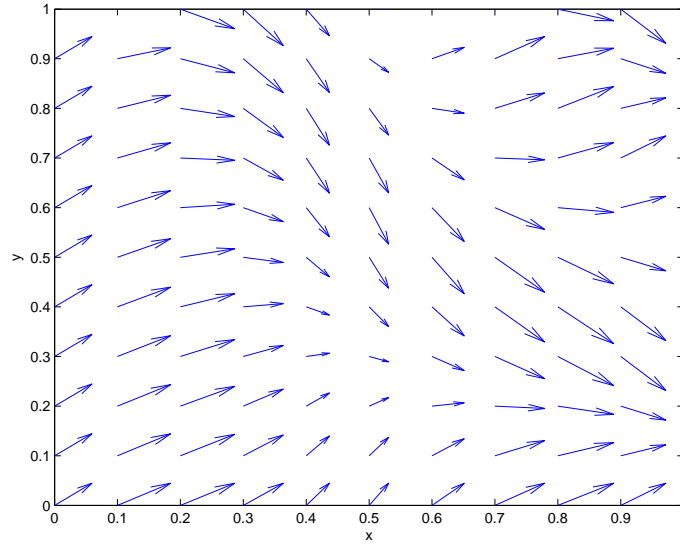


FIGURE 5.9: The wind field

Fuel data interpolation by Lambda method:

Since in our work, we don't work on vertical optimization, but we work on fixed altitude and consider only horizontal direction. Therefore, the altitude is fixed and fuel consumption considered as a function of aircraft true speed and weight. We do triangulation on the speed and weight domain, the grid points on this domain is the speed and weight grid points, $A = \{v_1, v_2 \dots v_n\}$ set of the points on the grid, A_1 is the set of indexes for points (vertices). $B = \{N_1, N_2, \dots N_m\}$ set of triangles, B_1 is the index set for triangles ($B_1 = \{1, 2, \dots, m\}$), $T_1 = \{t_1, t_2 \dots t_N\}$ is time grids, we define a variable $\alpha(i, j)$ where $(i, j) \in A_1$ taking value on $[0, 1]$ for each grids (vertices of the triangle) and a binary variable λ_i where $i \in B_1 = 1, 2, 3 \dots m$ for each triangle.

$$\sum_{i \in B_1} \lambda_i^n = 1, \text{ for all } n \in T \quad (5.49)$$

$$\sum_{(i,j) \in A_1} \alpha_{(i,j)}^n = 1 \text{ for all } n \in T \quad (5.50)$$

$$V(n) = \sum_{(i,j) \in A_1} V_i \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.51)$$

$$W(n) = \sum_{(i,j) \in A_1} W_i \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.52)$$

$$Fr(n) = \sum_{(i,j) \in A_1} Fr(i, j) \times \alpha_{(i,j)}^n \text{ for all } n \in T \quad (5.53)$$

$$\lambda_i^n \leq \sum_{s \in \{l \in B_1 | (i,j) \in N^l\}} \alpha_{(i,j)}^n \text{ for all } (i, j) \in B_1, n \in T \quad (5.54)$$

$$0 \leq \alpha_{(i,j)}^n \leq 1 \text{ for all } (i, j) \in B_1 n \in T \quad (5.55)$$

Model 5.3:

- Objective function:

$$\min (W_0 - W_N + 0.01 \cdot (\sigma_x + \sigma_y)) \quad (5.56)$$

- Subject to constraints:

$$W_N = 56613.75 \quad (5.57)$$

$$x_0 = 0 \quad (5.58)$$

$$x_N = 0.9 + \sigma x \quad (5.59)$$

$$y_0 = 0 \quad (5.60)$$

$$y_N = 0.9 + \sigma y \quad (5.61)$$

$$0.01 \leq \sigma_x \leq 0 \quad (5.62)$$

$$0.01 \leq \sigma_y \leq 0 \quad (5.63)$$

constraint (5.32) to constraint (5.38)

constraint (5.46) to constraint (5.53)

constraint (5.54) to constraint (5.60)

The units we used in our model for this trip is given by the following table (5.1):

Units for variables in model 5.3	
wind speed	Mach number
aircraft true speed	Mach number
segment length	nautical mile
x,y coordinate	nautical mile
aircraft weigh	kg
fuel consumption function(speed, weight, altitude)	kg (Mach number, kg, flight level)

TABLE 5.1: Units for variables in model 5.3

We run the model 5.3 in SCIP but it is too slow to solve, after running 2 days still 80% gap to finish, so we stopped. The reason slowing down the solving procedure is that, we have a quadratic constraint (5.32) for aircraft speed.

5.4 Linear Approximation of Second Order Cones

As we seen in model 5.3, the equation (5.32):

$$v(t)^2 = v_x(t)^2 + v_y(t)^2 \quad (5.64)$$

bring non-linearities to the model which slows down the solving procedure. Therefore, we would like to find a linear approximation for this constraint, which can speed up the solving procedure. Moreover, with linear constraints, we are able to use more efficient solvers like CPLEX to speed up the solving procedure. Here is the Conic quadratic programming introduced in [23]

We tried to relax the constraint by following step:

$$\sqrt[2]{v_x^2 + v_y^2} \leq v \quad (5.65)$$

This is a quadratic cone.

Definition 6. A quadratic cone (second order cone, icecream cone) is the set described by :

$$\sqrt{x_1^2 + x_2^2 + K + x_{n-1}^2} \leq x_n \quad (5.66)$$

Pure SOCPs (second order conic programming) can be solved by linear methods, There is one method given in [26]

$$\sqrt[2]{x_1^2 + x_2^2} \leq x_3 \Leftrightarrow \quad (5.67)$$

$$\alpha_0 = x_1 \quad (5.68)$$

$$\beta_0 = x_2 \quad (5.69)$$

$$\alpha_{i+1} = \cos\left(\frac{\pi}{2^i}\right)\alpha_i + \sin\left(\frac{\pi}{2^i}\right)\beta_i \quad (5.70)$$

$$\beta_{i+1} \geq |\sin\left(\frac{\pi}{2^i}\right)\alpha_i + \cos\left(\frac{\pi}{2^i}\right)\beta_i| \quad (5.71)$$

The error is :

$$\varepsilon = \cos\left(\frac{\pi}{2^n}\right)^{-1} - 1. \quad (5.72)$$

n controls the approximation error, $i = 1, 2, \dots, n$

However, there is one point we need to notice is that, by relaxing the equation (5.31) to (5.72), we are having the risk that model may not give the most optimal solution. As we introduced in figure (5.7), for fuel consumption function, there is an optimal speed 0.76 (this is just for aircraft type A320, aircraft performance data is different for different type of aircraft) Mach number. We can see from the graph (5.10), when aircraft true speed is bigger than 0.76, if the speed is more small, the problem is more optimal so the relaxation equation (5.58) will push the value of v as small as possible, push it more

close to $v_x^2 + v_y^2$, the error will be small. But if the true aircraft speed (value of $v_x^2 + v_y^2$) is smaller than 0.76, we always get 0.76 because of the relaxation.

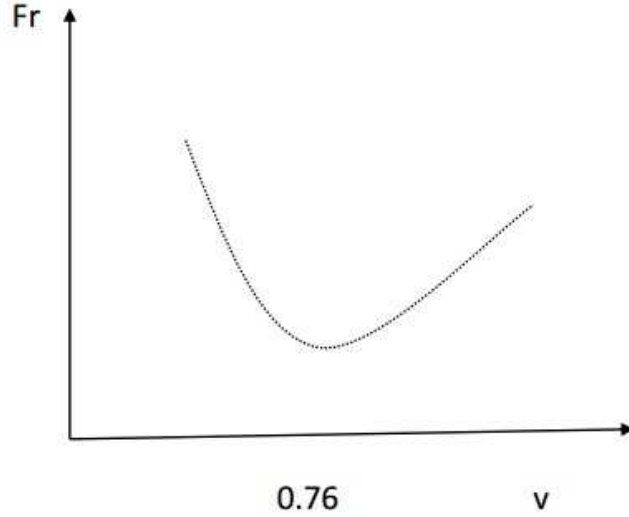


FIGURE 5.10: Optimal speed for unit distance fuel consumption function for aircraft type A320

However, even using conic linearization, we run the model in SCIP and result is still very slow. We waited for 3 days but the program still didn't finished. Apparently, we still need to speed up. In this case CPLEX still can't be used because of the nonlinearities of constraint (5.35). There is a way to linearise the equation(5.35) and also reduce the dimension of the interpolation of fuel consumption function. We chose discrete speed instead of continuous speed, which means we restrict the aircraft true speed to a set of discrete speed $\{v_1, v_2, v_3, \dots, v_p\}$, so each segment, the aircraft speed is one of the value from this set, and also in order to leave more flexibility to the model, we don't use conic linearization, from now on we test our model on SCIP.

5.5 Speed Discretization

As we analyse above, we discretize the possible speed for aircraft, this means, instead of giving an option for speed on continuous set $[0.4, 0.82]$ (this is the possible speed range for aircraft type A320), we give a discrete speed set: $\{v_i | 0.4 = v_0 \leq v_1 \leq v_2 \dots v_q = 0.82\}$. By doing this procedure, we can linearise the equation (5.38) (here we use the starting speed of each segment to calculate the length of segment), in order to do this, we introduce binary variables β_i for each speed step v_i having value on $\{0, 1\}$, $\beta_i = 1$ if v_i is

chosen, and $\beta_i = 0$ otherwise. By doing this step we linearize the equation (5.38) and obtain the following constraints:

$$f_i = \left(\sum_{j=1}^q V_j \beta_{(i,j)} \right) \Delta t \cdot \sum_{j=1}^q (Fr(V_j, W_i) \cdot \beta_{(i,j)}) \quad \text{for } i = 0, 1, \dots, N \quad (5.73)$$

$$l_i = V_i \Delta t \quad \text{for } i = 0, 1, \dots, N \quad (5.74)$$

$$V_i = \sum_{j=1}^q v_j \beta_{(i,j)} \quad \text{for } i = 0, 1, \dots, N \quad (5.75)$$

Where $\beta_{(i,j)}$ is binary variable, $V_i \in \{v_j, j = 1, 2, \dots, q | \min V = v_0 < v_1 < \dots < v_q = \max V\}$

By discretizing the speed, we are not only able to linearize the equation (5.38) but we also changed the fuel interpolation from two dimension to one dimension.

Fuel interpolation:

Through discretizing the speed, we change the fuel interpolation into one dimensional non-linear approximation, let us denote the weight grids by W_i where $i \in \Lambda$, Y is the indexes set of $k - 1$ segments that define the set of intervals, we have binary variables λ_j $j \in Y$ for each segment, corresponding variables $\alpha_i, i \in \Lambda$

$$W_{i+1} = W_i - f_i \quad (5.76)$$

$$\sum_{j \in \Lambda} \lambda_j^n = 1, \quad \text{for all } n \in T \quad (5.77)$$

$$\sum_{j \in Y} \alpha_j^n = 1 \quad \text{for all } n \in T \quad (5.78)$$

$$W(n) = \sum_{j \in A} W_j \times \alpha_j^n \quad \text{for all } n \in T \quad (5.79)$$

$$Fr(i, n) = \sum_{j \in A} Fr(i, j) \alpha_j^n \beta_i^n \quad \text{for all } i = 1, 2, \dots, p, n \in T \quad (5.80)$$

$$\lambda_1^n \leq \alpha_1^n \quad \text{for all } n \in T \quad (5.81)$$

$$\lambda_j^n \leq \alpha_{j-1} + \alpha_j^n \quad \text{for all } j \in \Lambda - 1, k \quad \text{for all } n \in T \quad (5.82)$$

$$\lambda_k^n \leq \alpha_{k-1}^n \quad \text{for all } n \in T \quad (5.83)$$

After speed discretization, our model 5.3 changed from continuous speed model to discrete speed model 5.4.

Model 5.4:

- Objective function:

$$\min (W_0 - W_N + 0.01 \cdot (\sigma_x + \sigma_y)) \quad (5.84)$$

- Subject to constraints: constraint (5.62) to constraint (5.67)
 constraint (5.32) to constraint (5.39)
 constraint (5.46) to constraint (5.53)
 constraint (5.79) to constraint (5.87)

We run the model 5.4 with SCIP, for 7 speed possible steps(0.76, 0.77....0.82), 15 weight grids (56500.0 ,58000.0, 59500.0, 61000.0, 62500.0, 64000.0, 65500.0, 67000.0, 68500.0, 70000.0, 71500.0, 73000.0, 74500.0, 76000.0, 77500.0). Let $\Delta t = 0.5$, $T=5$ (hour)
 The result is shown in the following table and the trajectory in given in figure (5.11).

The result for Trip1 with lambda method, with discrete speed	
Optimal fuel consumption is: 12508.2414606226 kg	Solving time is 586.01 seconds

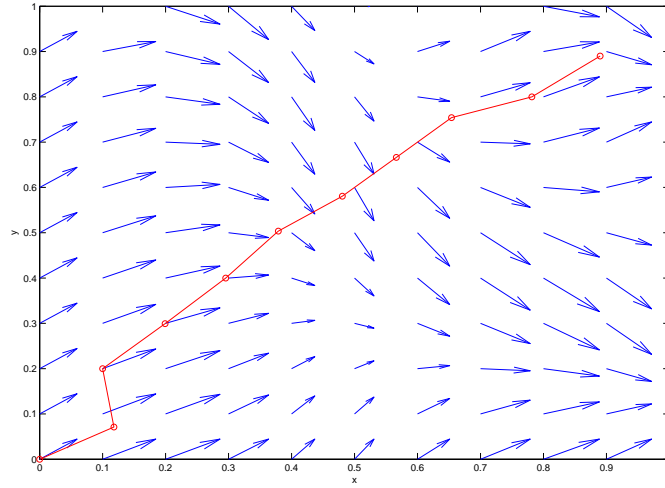


FIGURE 5.11: Optimal solution by using wind

To test how much fuel we saved by flying on wind optimal path, we tried to fly in shortest distance direction whatever the wind direction, solving time 43.29 seconds, optimal fuel consumption: $12691.30414 - 0.001 \times (0.0003801 + 0.00003801)$, $\sigma_x = 0.00003801$, $\sigma_y = 0.00003801$. In this case we save 183.10414 kg fuel (1.45%). The fuel saving is not very

obvious for this trip because we can see that, most of the track wind is still on shortest path direction. So wind influence almost same in optimal trajectory and shortest path.

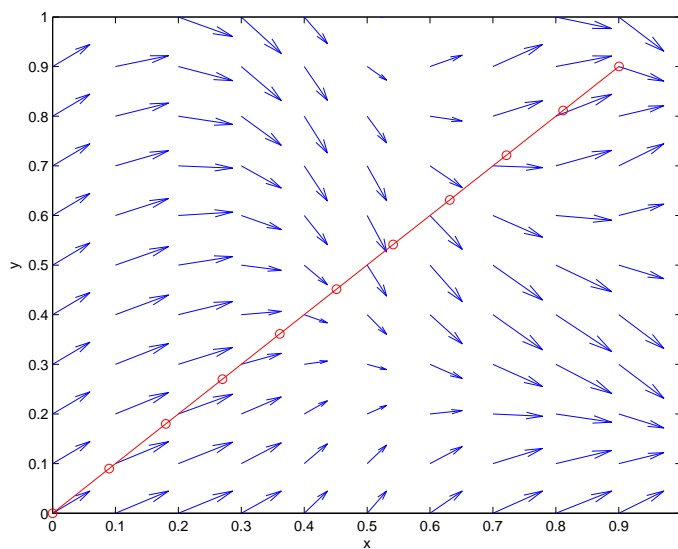


FIGURE 5.12: Trajectory when we don't consider any influence of the wind(Assume there is no wind)

Chapter 6

Adapted Model on Earth Surface with Real World Data

6.1 Mapping

Choosing a good coordinate system is quite important for our algorithm, As we analysed in chapter 5, both simple Cartesian coordinate or spherical coordinate is not proper for our model when consider the practical issue, data structure as well as solvers restriction. Therefore, we decide to map the solution space from earth surface to x-y plane in a way that can preserve distance, convert all the coordinate of points from (longitude, latitude) to the x-y Cartesian coordinate by preserving the distance, then work on x-y plane. This require a big amount of preprocessing work. But by this procedure, we can make our model more accurate, practical and fast, most importantly, we can make use current MIP solvers to solve the model. Therefore the mapping is quite important, and directly influence the model accuracy. We give the mapping procedure in three steps:

- First step, we want to find a mapping that can map any great circle line on the earth to the line that lies on equator, this is a rotation map. By this mapping we changed the great circle that connecting the departure point and arrival point to the line that lies on the equator. The procedure is shown in figure (6.1)

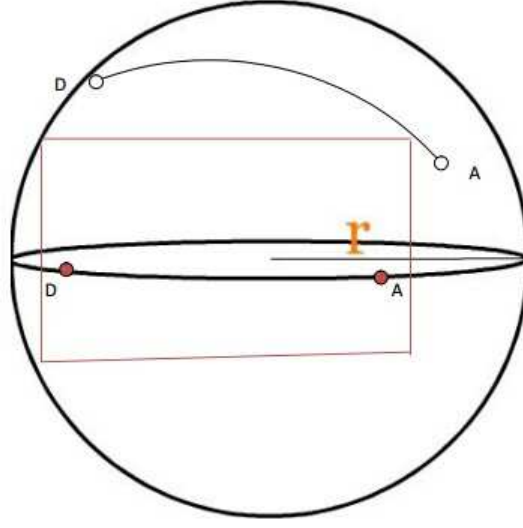


FIGURE 6.1: Mapping the great circle connecting departure and arrival point to the equator

This mapping is done by combining three basic rotating matrix. A basic rotation(called elemental rotation) is a rotation about one of the axes of a Coordinate system. The following three basic rotation matrices $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ rotate vectors by an angle θ about the x,y or z axis, in three dimensions, using the right hand rule,

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The coordinate system is right-handed, and the angle θ is positive. R_y , for instance, would fix the Y-axis, rotate toward the X-axis a vector aligned with Z-axis.

Now we want to rotate the the earth so that the great circle line DA coincide with equator, we assume that this rotation done by $R_x(\theta_1)$, $R_y(\theta_2)$, $R_z(\theta_3)$, let rotating matrix $M = R_x(\theta_1)R_y(\theta_2)R_z(\theta_3)$, this is still a rotating matrix by Euler's rotation theorem (In three dimensional space, the composition of two rotations is also a

rotation.). The procedure how to find the matrix M is given as following:

The coordinate of starting point and arrival point given by $(\lambda_D, \phi_D), (\lambda_A, \phi_A)$ where λ is latitude and ϕ is longitude of the point, First we find the spherical coordinate of D and A:

$$x = R \cos(\lambda) \cos(\phi) \quad (6.1)$$

$$y = R \cos(\lambda) \sin(\phi) \quad (6.2)$$

$$z = R \sin(\lambda) \quad (6.3)$$

we got the spherical coordinates $D(x_1, y_1, z_1)$, $A(x_2, y_2, z_2)$ by above computation. After applying rotation matrix M, we got longitude of \hat{A} (the point A after applying rotation) is zero (E00), and longitude of \hat{D} can be calculated by $\phi_{\hat{A}} = \frac{d_{gcd}(D, A)}{R}$, where d_{gcd} means the great circle distance between two points. Therefore, we have the spherical coordinates (which lies on equator) for this two points on the line after mapping as following:

$$\hat{D} = \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{A} = \begin{bmatrix} R \cos(\phi_{\hat{A}}) \\ R \sin(\phi_{\hat{A}}) \\ 0 \end{bmatrix}$$

We assume that we rotate θ_1 degree along x-axis, θ_2 degree along y-axis, θ_3 degree along z-axis. We have :

$$M = \begin{bmatrix} \cos \theta_2 \cos \theta_3 & -\sin \theta_3 \cos \theta_2 & \sin \theta_2 \\ \sin \theta_1 \sin \theta_2 \cos \theta_3 + \sin \theta_3 \cos \theta_1 & -\sin \theta_1 \sin \theta_2 \sin \theta_3 + \cos \theta_1 \cos \theta_3 & -\sin \theta_1 \cos \theta_2 \\ -\cos \theta_3 \cos \theta_1 \sin \theta_2 + \sin \theta_3 \sin \theta_1 & \cos \theta_1 \sin \theta_2 \sin \theta_3 + \sin \theta_1 \cos \theta_3 & \cos \theta_1 \cos \theta_2 \end{bmatrix}$$

Now, we can apply this rotation matrix to this two points, and find three rotation angle by solving the following nonlinear system.

$$MD = \hat{D} \quad (6.4)$$

$$MA = \hat{A} \quad (6.5)$$

Solving this non-linear system with coordinate of D and A seems quite complicated, here is more direct way to find the rotation matrix.

Alternative way:

We do three rotation, first rotate along z-axis, then rotate along y-axis, then rotate

along z axis. This procedure is shown in figure (6.2).

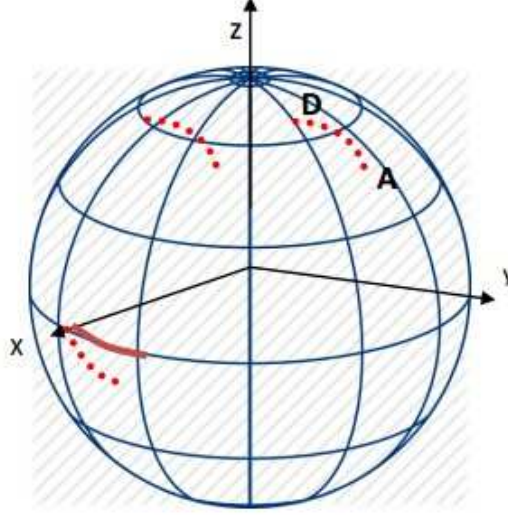


FIGURE 6.2: Mapping illustration

1. First rotate along z axis such that longitude of D map to zero:

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R \cos \lambda_D \cos \phi_D \\ R \cos \lambda_D \sin \phi_D \\ R \sin \lambda_D \end{bmatrix} = \begin{bmatrix} R \cos \lambda_D \\ 0 \\ R \sin \lambda_D \end{bmatrix}$$

So we have that $\theta = \phi_D$

2. Second step is rotate along y-axis such that point D will come to equator:

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} R \cos \lambda_D \\ 0 \\ R \sin \lambda_D \end{bmatrix} = \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$$

From this easily we got that $\theta = \lambda_D$

3. Third step is rotating along x axis(means that x is fixed, this will grantee that point D will remain fixed after this rotation) such that point A will arrive on equator. For to do this, we know the coordinate of point D after applying first two rotation, and we are expecting after this third rotation the Z coordinate of D is zero. There for we have the following equation to find rotation angle θ_x for this rotation. denote \hat{A} is the point A after went through above two rotation

$$\begin{bmatrix} \cos \theta_D & 0 & \sin \theta_D \\ 0 & 1 & 0 \\ -\sin \theta_D & 0 & \cos \theta_D \end{bmatrix} \begin{bmatrix} \cos \lambda_D & -\sin \lambda_D & 0 \\ \sin \lambda_D & \cos \lambda_D & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R \cos \lambda_A \cos \phi_A \\ R \cos \lambda_A \sin \phi_A \\ R \sin \lambda_A \end{bmatrix} = \begin{bmatrix} x_{\hat{A}} \\ y_{\hat{A}} \\ z_{\hat{A}} \end{bmatrix}$$

Apply the R_x to \hat{A}

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{bmatrix} \begin{bmatrix} x_{\hat{A}} \\ y_{\hat{A}} \\ z_{\hat{A}} \end{bmatrix} = \begin{bmatrix} x_{\hat{A}} \\ y_{\hat{A}} \\ z_{\hat{A}} \end{bmatrix}$$

$$y_{\hat{A}} \sin \theta_x + z_{\hat{A}} \cos \theta_x = 0 \quad (6.6)$$

$$\theta_x = \arctan\left(\frac{-z_{\hat{A}}}{y_{\hat{A}}}\right) \quad (6.7)$$

So from above system, by solving last equation we can get the angle for third rotation θ_x . Finally the rotation matrix M is given by:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{bmatrix} \begin{bmatrix} \cos \theta_D & 0 & \sin \theta_D \\ 0 & 1 & 0 \\ -\sin \theta_D & 0 & \cos \theta_D \end{bmatrix} \begin{bmatrix} \cos \lambda_D & -\sin \lambda_D & 0 \\ \sin \lambda_D & \cos \lambda_D & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The second step is to choose a solution space on the earth surface according to our starting point and arrival point. We fix a square area on the earth surface around the great circle line connecting starting point and arrival point, which included all the possible trajectory. This is shown in figure (6.3).

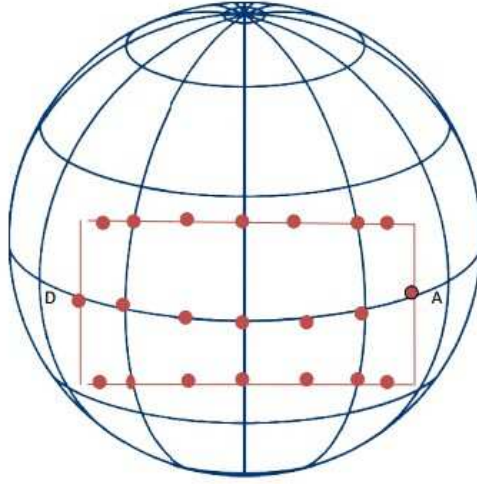


FIGURE 6.3: The projection of the area on equator

- Last step is, we map this area onto the x-y plane by a mapping that can preserve distance between any two point. The procedure is shown in figure (6.4).

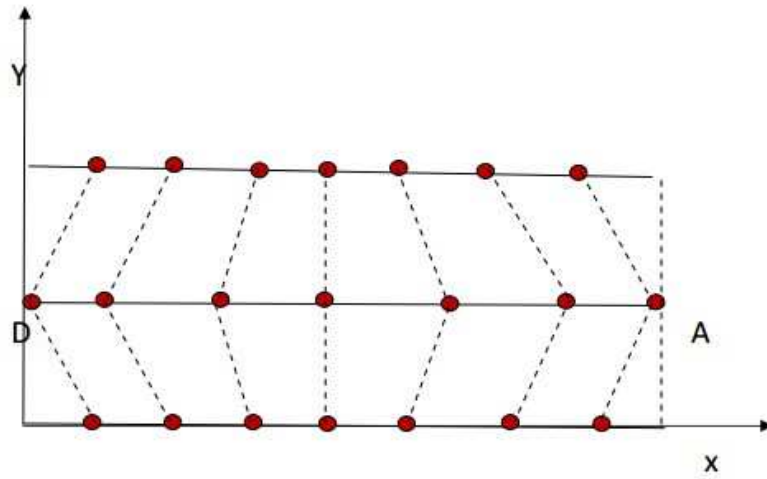


FIGURE 6.4: Projection of the area on x-y plane

Therefore, after all this preprocessing, we can map solution space and wind grids on the earth to the x-y plane preserving distance, then we can simply formulate our problem in the projected X-Y plane.

6.2 3D Realistic Model with Real World Data

Assume the starting location $D(\phi_1, \theta_1)$ and arrival location is $A(\phi_2, \theta_2)$. We rotate the earth to let the great circle line connecting D and A come to equator. After rotation, let us denote the coordinate of departure point D by $\hat{D}(0, \hat{\theta}_1)$ and the arrival point A $\hat{A}(0, \hat{\theta}_2)$. We chose the area which include all the possible solution space $[\hat{\theta}_1, \hat{\theta}_2 + \theta] \times [-20, 20]$ (this means the area from latitude -20 degree to 20 degree, longitude from $\hat{\theta}_1$ to $\hat{\theta}_2 + \theta$, we put θ as a small extension considering there will be have deviation), we project the wind grid points by the method we presented in section 6.1 from the earth surface to the x-y plane by preserving the distance by following calculation:

$$\hat{D} = (0, 0) \quad (6.8)$$

$$\hat{A} = (0, d_{gcd}(0, \hat{\theta}_1, 0, \hat{\theta}_2 + \theta)) \quad (6.9)$$

$$y_i = \hat{\phi}_i \times 60 \quad (6.10)$$

$$x_i = d_{gcd}(0, \hat{\theta}_1, 0, \hat{\theta}_2) - \frac{1}{2}d_{gcd}(\hat{\phi}_i, \hat{\theta}_i, \hat{\phi}_i, \hat{\theta}_2 + \theta) + 60.108 \times \cos(\hat{\phi}_i) \times (\hat{\theta}_i - \hat{\theta}_1) \quad (6.11)$$

Where great circle distance d_{gcd} is given by :

$$d_{gcd} = r\Delta\sigma \quad (6.12)$$

$$\Delta\sigma = \arctan\left(\frac{\sqrt{(\cos\phi_2 \sin\Delta\theta)^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos\Delta\theta)^2}}{\sin\phi_1 \sin\phi_2 - \cos\phi_1 \cos\phi_2 \cos\Delta\theta}\right) \quad (6.13)$$

After this projection, actually the equator will arrive at x axis, but for computation convenient, we would like to work on positive y, so we just translate whole mapping by $y + d_{gcd}(-20, 0, 20, 0)$, then whole area will be above the x axis. The mapping illustrated in figure(6.5).

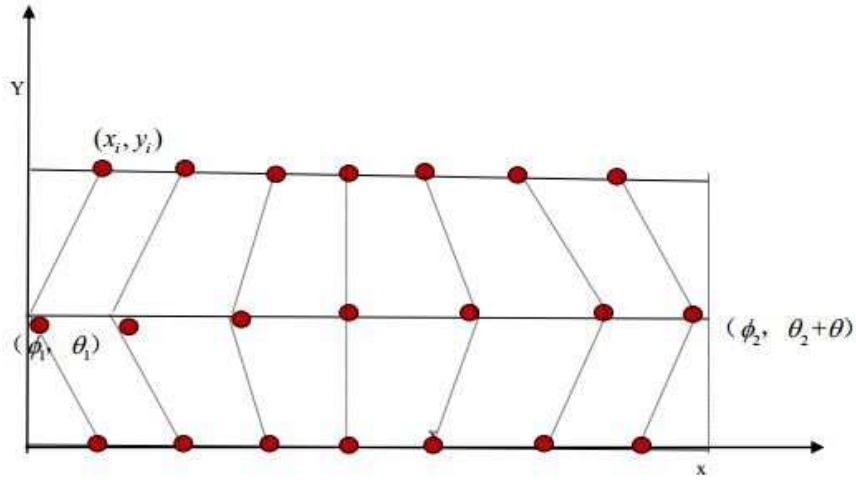


FIGURE 6.5: Mapping

After all this preprocessing, we have the invertible mapping and wind data on x-y plane. Therefore, from now on, we can use our 2D model formulation which we presented in chapter 5.

6.3 Trip 2 with Lambda Method

We give a trip from from (N00,E45) flying to (N00, E87.5), where, N00 represent for north latitude 0 degree and E45 represent for longitude 45 (east) degree. The whole trip in great circle distance is 2554.6 nautical miles, We schedule 5 hour for whole trip (remember that we only considering cruise stage). We chose the solution space as: $[-17.5, 17.5] \times [45, 90]$

The Model is given as following, the notation is same as before:

Model 6.1

- The objective function:

$$\min (W_0 - W_N + 0.01 \times (\sigma_x + \sigma_y)) \quad (6.14)$$

- Constraints:

$$V_i^2 = (V_i^x)^2 + (V_i^y)^2 \quad (6.15)$$

$$x_{i+1} = x_i + \Delta t \cdot (V_i^x + w_i^x) \quad (6.16)$$

$$y_{i+1} = y_i + \Delta t \cdot (V_i^y + w_i^y) \quad (6.17)$$

$$w_i^x = w^x(x_i, y_i) \quad (6.18)$$

$$w_i^y = w^y(x_i, y_i) \quad (6.19)$$

$$l_i = V_i \cdot \Delta t \text{ for all } i = 1, 2 \dots N - 1 \quad (6.20)$$

$$f_i = l_i \times Fr_i(V_i, W_i) \text{ for all } i = 1, 2 \dots N - 1 \quad (6.21)$$

$$W_N = 56613.75 \quad (6.22)$$

$$x_0 = 0 \quad (6.23)$$

$$x_N = 2554.6 + \sigma_x \quad (6.24)$$

$$y_0 = 0 \quad (6.25)$$

$$y_N = 0 + \sigma_y \quad (6.26)$$

$$10 \geq \sigma_x \geq 0 \quad (6.27)$$

$$10 \geq \sigma_y \geq 0 \quad (6.28)$$

First, we solve the Model 6.1 using 2D lambda method for wind interpolation (constraint (6.18),(6.19)) and 1D lambda method for fuel interpolation (constraint (6.21)). The interpolation method is the same as we did in chapter 5 for trip 1. Then we get a new model 6.2 as following : Model 6.2

- The objective function:

$$\min (W_0 - W_N + 0.01 \times (\sigma_x + \sigma_y)) \quad (6.29)$$

- subject to constraints :

constraint (6.15) to constraint (6.28)

constraint (5.46) to constraint (5.53)

constraint (5.81) to constraint (5.87)

We run model 6.2 in SCIP with 3 speed steps (0.76,0.81,0.82) and 4 weight grids (speed level means the possible speed steps for aircraft, weight grids means we have 4 weight grids used to interpolate the fuel consumption, it does not mean we only have 4 weight steps for aircraft). Wind grids taken at each 2.5 degree, we have 19×15 wind grid points and 504 wind triangles. The result is:

The result for Trip2 with lambda method, with discrete speed	
Optimal fuel consumption is: 13047.7027 kg	Solving time is 94343.12 seconds

The trajectory is given in figure(6.6).

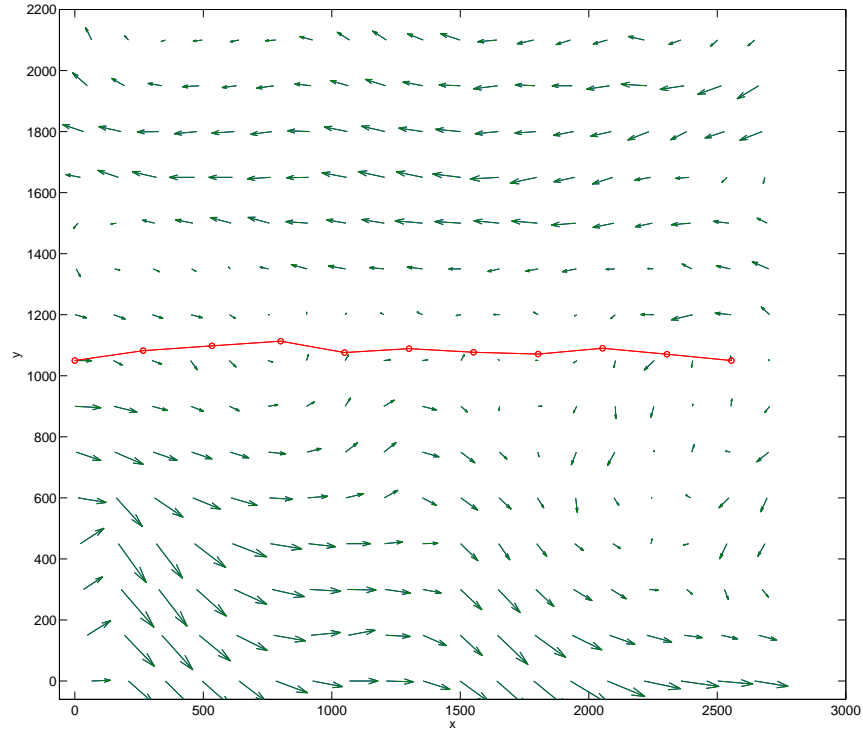


FIGURE 6.6: Optimal trajectory

6.4 Trip 2 with Delta Method

As we see above, the algorithm take too long time (more than 26 hours) to solve the model, which is bad for practical use, and we need more good algorithm such that it fits for practical application. Considering the practical usage of free flight, we need an algorithm works more fast. There is only few ways to speed up the programm, either change the solver or change the algorithm, for the former, we do not have much choice because of the computational complexity of the problem. As we see there is second order constrain, which CPLEX can not solve, one way to linearise it is quadratic cone. However, as we proved before it only work for the speed that is bigger than the optimal speed, this have a limitation for the algorithm. It seems, up to now, we only can use SCIP to solve the model. Therefore, in order to speed up, we need to modify the model, as

we see in chapter 4, for non-linear function approximation, we have 3 different methods, [23] shows that the lambda method is computationally inferior to the delta method as its linear programming relaxation always produce worse bound than the relaxation of the delta method. And it is proven that both delta method and lambda method produce same value.

Here we use Delta method for Model6.1 in order to speed up. In this case, which is different from lambda method is that we need to order our triangles. There are a plenty of different ways to order the triangles, we use the following ordering in our model which shown in figure (6.7):

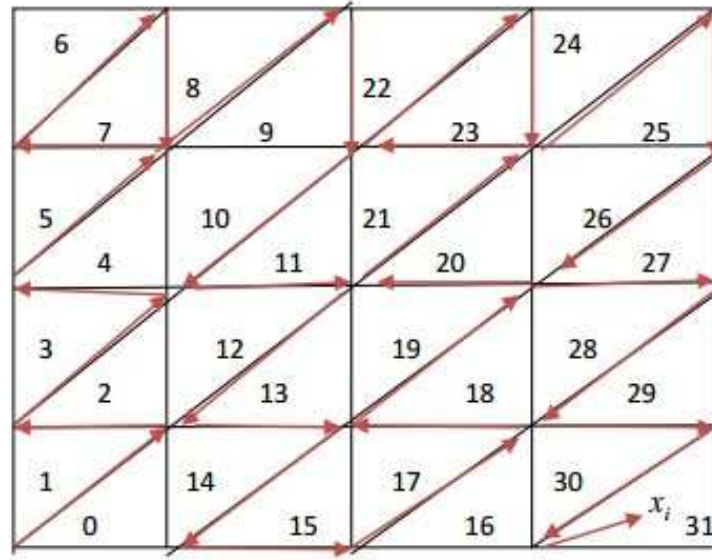


FIGURE 6.7: Triangle ordering for Delta Method

The wind interpolation by delta method:

Let A be the indices set for grids, B is the $4 \times n$ matrix such that $B(1, :)$ is the index of the triangles, and other three columns are the index of three vertices of corresponding triangles, we introduce binary variables w_i associated with triangle N_i $i = 1, 2, \dots, q$, a variable δ_i^j corresponding to j th vertex of triangle N_i , Then:

$$\sum_{j=1}^3 \delta_{i,j}^n \leq 1 \text{ for all } i = 1, 2, \dots, q, n \in T \quad (6.30)$$

$$\delta_{i,j}^n \geq 0 \text{ for all } i = 1, 2, \dots, q-1, n \in T \quad (6.31)$$

$$\sum_{j=1}^3 \delta_{i+1,j}^n \leq w_i^n \text{ for all } i = 1, 2, \dots, q-1, n \in T \quad (6.32)$$

$$w_i^n \leq \delta_{i,3}^n, n \in T \quad (6.33)$$

$$x^n = x_{1,1} + \sum_{i=1}^q \sum_{j=1}^3 (x_{i,j} - x_{i,1}) \delta_{i,j}^n, n \in T \quad (6.34)$$

$$y^n = y_{1,1} + \sum_{i=1}^q \sum_{j=1}^3 (y_{i,j} - y_{i,1}) \delta_{i,j}^n, n \in T \quad (6.35)$$

$$w_x^n = w_x(x_{1,1}, x_{1,1}) + \sum_{i=1}^q \sum_{j=1}^3 (w_x(x_{i,j}, y_{i,j}) - w_x(x_{i,1}, y_{i,1})) \delta_{i,j}^n, n \in T \quad (6.36)$$

$$w_y^n = w_y(x_{1,1}, y_{1,1}) + \sum_{i=1}^q \sum_{j=1}^3 (w_y(x_{i,j}, y_{i,j}) - w_y(x_{i,1}, y_{i,1})) \delta_{i,j}^n, n \in T \quad (6.37)$$

Unit distance fuel consumption function interpolation:

For fuel consumption function, since we discretize the speed, now we just need to do delta method approximation on one dimension. Here we have segments instead of triangles, and for segment, since it is on one line, the ordering is done automatically by binary variables w_i , use the same notes as above (here we use segment instead of triangles). $V = \{v_i, i = 1, 2, \dots, p\}$ is the set of discrete speeds.

$$\delta_{i+1}^n \leq w_i^n \text{ for all } i = 1, 2, \dots, q-1, n \in T \quad (6.38)$$

$$w_i^n \leq \delta_i^n \text{ for all } i = 1, 2, \dots, q-1, n \in T \quad (6.39)$$

$$0 \leq \delta_i^n \leq 1 \text{ for all } i = 1, 2, \dots, q-1, n \in T \quad (6.40)$$

$$W^n = W_1 + \sum_{i=1}^{q-1} (W_{i+1} - W_i) \delta_i^n, n \in T \quad (6.41)$$

$$Fr_l^n = Fr(V_l, W_1) + \sum_{i=1}^{q-1} (Fr(V_l, W_{i+1}) - Fr(V_l, W_i)) \delta_i^n \text{ for all } l = 1, 2, \dots, p, n \in T \quad (6.42)$$

here $m = 1, 2, \dots, k-1$ m is the index of segment,

i is the index of time, j is the index of speed grids.

For trip 2 applying model 6.1 with fuel and wind interpolation by delta method we have new model 6.3:

Model 6.3

- The objective function:

$$\min (W_0 - W_N + 0.01 \times (\sigma_x + \sigma_y)) \quad (6.43)$$

- subject to constraints :
 constraint (6.15) to constraint (6.28)
 constraint (6.30) to constraint (6.37)
 constraint (6.38) to constraint (6.42)

We implement the model 6.3 in SCIP, run with 3 speed levels, 4 weight grids and wind grids given at each 2.5 degree, we have 19×15 wind grid points and 504 wind triangles. The result is:

The result for Trip2 with delta method, with discrete speed	
Optimal fuel consumption is: 13047.70 kg	Solving time is: 2139.33 seconds

If we don't consider the wind and keep flying on the shortest distance, fuel consumption is 13230, we save 183kg fuel (1.4 percent).

Here the wind around our trip is quite small, average is 1.5m/s. So actually the trip is not benefiting much from wind much. If we try to enlarge the wind by 10 times (Assume we have wind around 15m/s), we got fuel consumption 12764.1246kg which is much smaller than the case if we don't fly wind optimal direction. In this case we save 3.5% fuel. Trajectory is given in the figure (6.8)

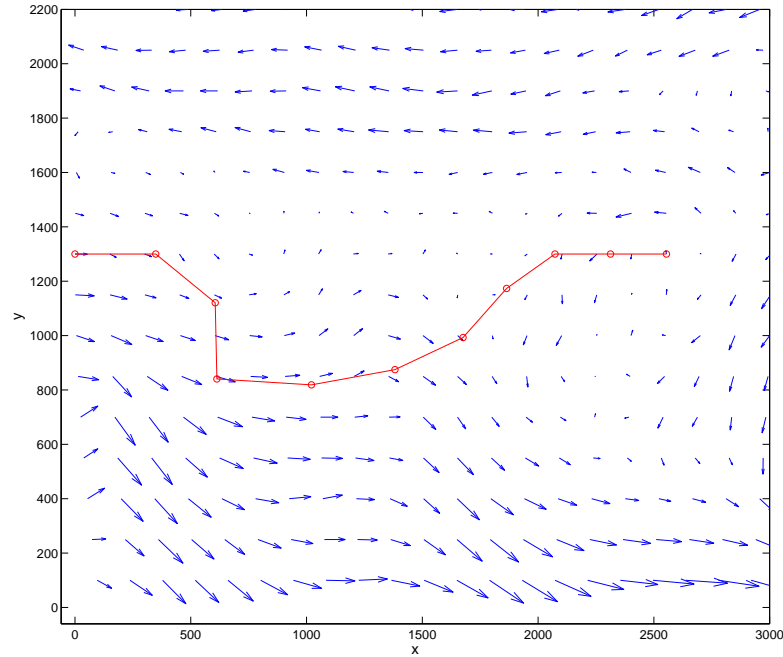


FIGURE 6.8: Trajectory in stronger wind

As we show above, we implemented our model by using various method of linearization, using various grid and run with SCIP. It turns out that delta method is more efficient than lambda method while returning the same value for objective function. The method used for linearization is not the only factor that influence solving speed of our algorithm, it also depends on how big our instance. For example, how fine the grid points for wind, how many weight and speed steps we are considering when doing the fuel and wind interpolation, the finer the grid, the bigger the model become (in terms of number of variables and instances) the more time it takes. The only reason we take more finer wind grid is considering if there will have drastic change in wind in very small area, which we may ignore if we use very coarse grids. And introducing more speed allowance for aircraft also may bring fuel saving. Therefore it is very important to chose reasonable grid levels which can bring the accuracy and efficiency for our model. We did several experiments for trip 2 with different methods, and different grid points. The result is shown in the following table (figure(6.9)):

possible speed steps and weigh grids used in trip 2 experiment		
3 speed step, 4 weight grids	0.76, 0.81, 0.82	56500, 64000, 71500, 77500
7 speed step, 15 weight grids	0.76, 0.77,..., 0.82	56500, 58000,..., 77500
27 speed step, 15 weight grids	0.4, 0.44,...,0.82	56500, 58000,...,77500

TABLE 6.1: grid points

With wind (grid point) degree wind triangles speed step weight step		Without wind 3 speeds 4 weights	With wind(19/15) 2.5 degrees 504 triangles 3 speed steps 4 weight steps	With wind(19/15) 2.5 degree 504 triangles 7 speed step 15 weight steps	Wind with(11/9) 5 degrees 160 triangles 3 speed steps 4 weight steps	With wind(6/5) 10 degrees 40 triangles 3 speed steps 4 weight steps	With wind(6/5) 10 degrees 40 triangles 7 speed steps 15 weight steps	With wind(6/5) 10 degrees 40 triangles 27 speed steps 15 weight steps
Lambda method	Fuel consumption (kg)	13620.24	13047.70	12858.81	13047.26	13047.70	12820.70	12820.70
	Solving time (seconds)	0.84	94343.12	2234321.45	3326.20	329.85	3060.22	2954.86
Delta method	Fuel consumption (kg)	13620.76	13047.70	12858.81	13038.47	13047.26	12820.70	12820.70
	Solving time (seconds)	1.09	2139.33	5371.88	562.56	32.58	80.89	173.95

FIGURE 6.9: Trajectory in stronger wind

Here is the speed levels and weight grids we used in the experiment.

As we see from figure (6.9), the wind field dose not change drastically in 10 degrees. This is also tested by above result. It is shown that, the result don't benefit by taking more fine wind grid. It does not make difference between taking grid points at each 2.5 degrees and at each 5 degrees, even at 10 degrees, but only slows down the algorithm. However, we can't exclude some extreme case that even within 2.5 degrees distance, the wind can be change dramatically. For this case we can try to take the average of starting wind and ending wind for each segment(in above case we simply took the wind at the starting point for each segment). As for speed, and weight, we have seen from figure(6.9) that the more possible speed steps, and more finer weight grids we take, the more fuel we save. Comparing the experiment result for 7/15 (7 possible speed step, 15 weight grids) and 27/15 (27 possible speed step, 15 weight grids), we have seen from figure (6.9), there is no difference in the objective functions value. It seems slowing down does not benefit a lot for fuel consumption, because the 27 possible speed is obtained by adding more smaller speed to 7 possible speed. And most obviously, delta method, is much faster than lambda method. Based on above testing result, From later on, we run all experiment with delta method, and wind grids at each 5 degrees.

6.5 Trip 3 with Delta method

We try model 6.3 with another trip

Flying from (N-5,E-125) to (N0,E-75), whole trip great circle distance is 3007.65812 nautical miles, scheduled time for whole trip is 5.5 hours, time step is half hour. Using delta interpolation for both wind and fuel function, taking wind grids 11×9 (5 degrees), 160 triangles, with 3 possible speed steps (0.76,0.81,0.81) and 4 weight steps. We run the model in SCIP, the result is given as following:

The result for Trip 3 with delta method in weak wind	
Optimal fuel consumption is: 15931.502kg	Solving time is 547.94 seconds

The trajectory is as shown in figure (6.10)

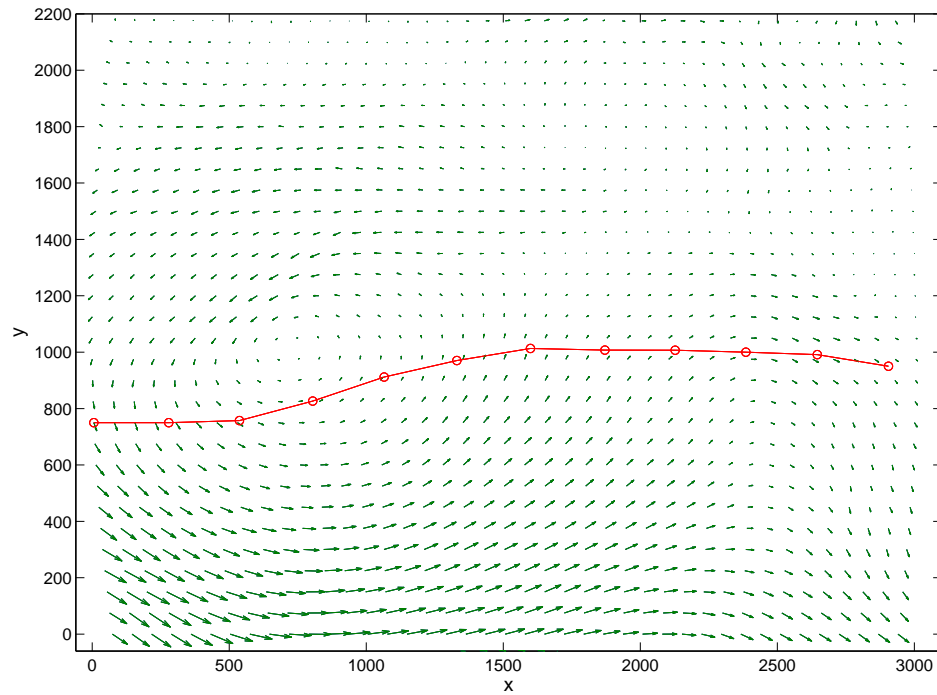


FIGURE 6.10: Trajectory for trip 3 with weak wind with 3 speed, 4 weight grids

As same before, we have very weak wind (average 2 m/s) around the shortest distance direction, so the trajectory still have less benefit from wind, So we try the model in a stronger wind (enlarge the wind by 5 times), and the result is:

The result for Trip 3 with delta method, in stronger wind	
Optimal fuel consumption is: 14176.5212kg	Solving time is: 639.85 seconds

The trajectory is given in figure (6.11)

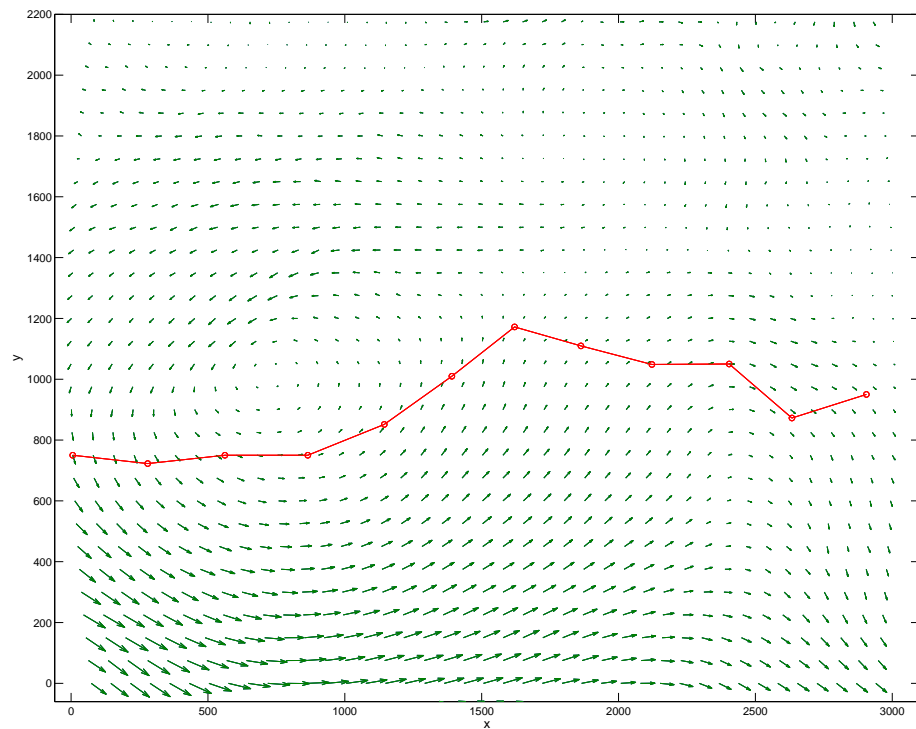


FIGURE 6.11: Trajectory for trip 2 with stronger wind

If we run the model without wind, flying on shortest distance, the fuel consumption is 16824.13 kg. By estimating the wind contribution on the shortest distance, at least we are saving 2% fuel by flying in optimal trajectory in weak wind and 6% by flying in stronger wind.

We run the model 6.3 with trip 3 with different number of speed levels and weight grids, The result is shown in the following table.

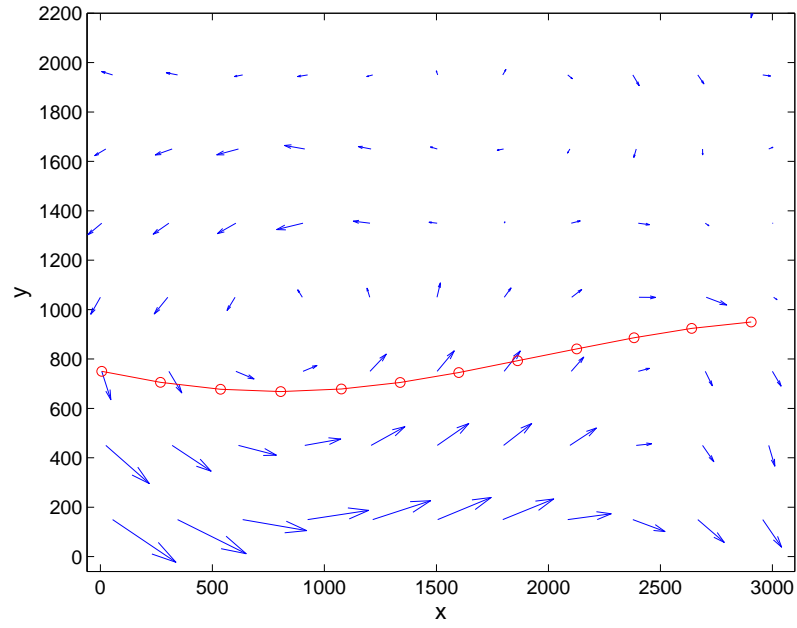
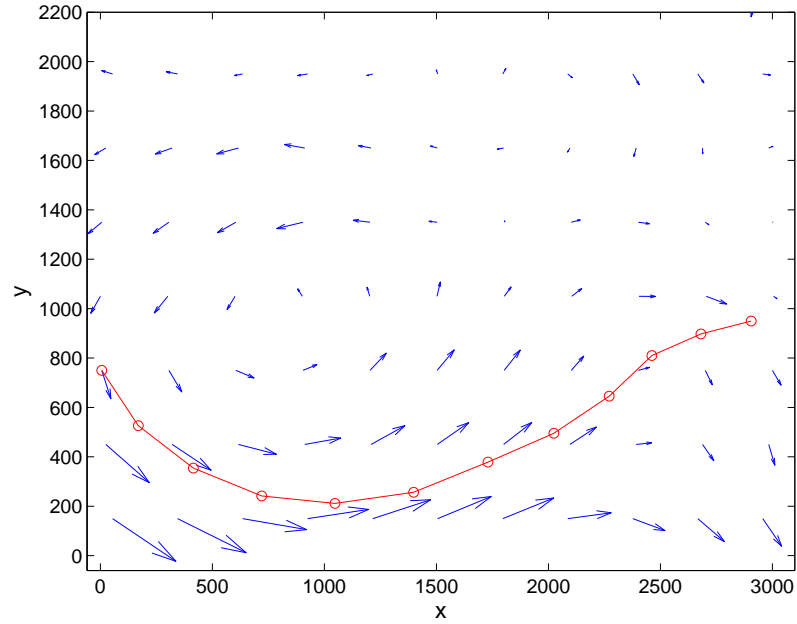
Trip 3 with delta method with wind grids at each 5 degrees, 160 triangles,		
(1) 3 speed steps, 4 weight grids with weak wind	15931.5023 kg	547.94 seconds
(2) 7 speed steps, 15 weight grids with weak wind	14953.982 kg	1245.21 seconds
(3) 27 speed steps, 8 weight grids with weak wind	14410.1621 kg	28090.12 seconds
(4) 3 speed steps, 4 weight grids with stronger wind	14176.52 kg	397.62 seconds
(5) 7 speed steps, 15 weight grids with stronger wind	14176.96 kg	479.24 seconds
(6) 27 speed steps, 8 weight grids with strong wind	11891.6313 kg	27127.5 seconds

TABLE 6.2: Trip 3 experiment result

As we seen above that, using more possible speed steps help us to save more fuel. For previous trip, we see that only speeding up is befitting. However, for this longer trip, giving only possible speed which is bigger than optimal speed is causing big deviation. Therefore we need speed which allow the aircraft slow down. This will prevent deviation and save more fuel than the case that only allow speeding up. For above table, the speed and weight grids are given as following:

possible speed steps and weigh grids used in trip 3 experiment		
3 speed step, 4 weight grids	0.76,0.81,0.82	56500, 64000,71500,77500
7 speed step, 15 weight grids	0.76,0.77,...0.82	56500,58000,59500,...77500
27 speed step, 8 weight grids	0.4,0.44,0.46,0.50,0.52...0.81,0.82	56500,59500,...77500

From by above two table, this trip actually benefit a lot from slowing down. Therefore, this is again a strong proof that we can't just take speeding up from optimal speed, and we can't use conic linearization. If we consider use conic, we end up with only speed above 0.76 and have never reached this optimal fuel consumption. So from all the experiments, we can come to the conclusion that, in all case we considered up to now, taking wind grids at each 5 degrees is most reasonable choice from both accuracy and solving time point. For speed steps and weight grids, 27 speed steps(or less but also including speed that allow us slow down) and weight grids would be best choice for our model. Here is the trajectory for trip 2 using the grids 27×8 (speed, weight) with weak wind (figure (6.12)) and strong wind (figure (6.13)):

FIGURE 6.12: Trajectory for trip 2 with weak wind 27×8)FIGURE 6.13: Trajectory for trip 2 with stronger wind 27×8

We also tried the Trip 3 with model 6.3 taking time step more smaller $\Delta t = 0.25$ The result is:

Trip 3 with $\Delta t = 0.25$		
(1) 3 speed step, 5 weight step with weak wind	14028.950851 kg	31436.74 seconds
(2) 7 speed step, 15 weight step with weak wind	14028.004 kg	57907.5 seconds
(3) 27 speed step, 8 weight step with weak wind	not able to solve	not able to solve

From above, we can see that, we are able to save more fuel by taking more smaller time step. However, this slows down the solving procedure a lot. For this small time step, we only able to solve the instance with up to 7 speed steps and 15 weight grids. Although, we are benefiting by taking more small time step, but we need to consider passenger's comfort level and pilot's workload. Changing direction every 15 minutes may not a good choice from passengers point of view and also for pilot.

Chapter 7

Conclusion

7.1 Error Estimation for Model

First, we want to evaluate how reliable our model when calculating the fuel consumption. Considering the safety side, we need to give very accurate result about fuel consumption for whole trip. Besides the estimated minimum fuel for trip, we need to load some amount of extra fuel for safety side. This extra fuel must include the part of fuel due to the error coming from model. However, fuel burns fuel, we never fill up the tanks, so estimating the error is as important as estimating the optimal fuel consumption. We give the error estimation from our model side, which should be considered when the captain load the extra fuel before taking off. The table (7.1) shows the error estimation. We simulate the path from our solution, calculate the fuel consumption for the optimal trip manually. It shows that, the result of our model is quite reliable when the data from industry is accurate enough. We estimate the error for model 6.3 in strong wind and weak wind, with 4 different instance by taking different speed steps, weight grids and wind grids.

This error estimation table indicates that, we can reduce the error from 0.15% to 0.007% by taking more possible speed steps and more fine weight grids.

Estimated error for model 6.3 with wind grid at each 5 degree(11×9)			
3×5 , weak wind	Model: 15931.5023	Simulation: 15955.585852kg	Error 0.1509%
7×15 , weak wind	Model: 14953.982 kg	Simulation: 14954.1005kg	Error 0.007 %
3×5 , strong wind	Model: 14176.52	simulation: 14201.3527kg	Error 0.174 8%
7×15 , strong wind	Model :14953.982 kg	Simulation: 14954.1005kg	Error 0.004 %

TABLE 7.1: Error estimation

7.2 Conclusion

We formulate the Free-Flight routing problem using Mixed Integer Programming and presented different approaches to interpolate the black box functions (fuel consumption and wind) with piecewise linear functions. We also tried different methods such as second order quadratic conic linearization, speed discretization to linearise the non linear constraint and speed up the program. Moreover, we presented the methods to project the solution space on earth surface to the X-Y plane. By this mapping, we successfully handled the roundness of the earth and made the model more accurate and practical. Most importantly we developed an algorithm that produce an optimal path for Free-Flight. We test our model for aircraft type A320, in different wind field, for different time steps and different grid size (wind grid, speed steps, weight grid).

The result of our study indicates that when the conditions are favourable, aircraft flight profile optimization with respect to known wind conditions can improve performance more than achieved in this study. The performance of our algorithm depends on the wind grid size, time step, possible speed steps and weigh grids. Therefore, it is important to choose a proper grid size, which can speed up the program and does not lower accuracy at the same time. The experiment result shows that, taking wind at each 5 degrees can give better solution than taking wind on more bigger grids, but taking more finer grids than 5 degrees don't benefit the model, but slows down. Allowing more speed steps benefits a lot in most of the cases, especially taking speed steps which include both speeds possible to speed up and speed down from optimal speed. For weight grids, too fine weight grids may slow down the solving procedure, but it make our fuel consumption estimation more accurate. Therefore, if more weight grids available for interpolating fuel consumption function, it is good to use more fine weight grids.

For our model, we came to the conclusion that, wind grids at each 5 degrees, 27 speed steps (including speeds that are smaller than the optimal speed), 15 weight grids, $\Delta t = 0.5$ is the best choice for model. From our model, the result shows that with weak wind we can save from 1.5% fuel to 3.5% , in strong wind we can save around 6% to 13% fuel.

7.3 Future Plan for Project

As we mentioned in chapter 1, the thesis was carried out on the project by Lufthansa System, aiming to be practically used in near future in industry. The Free-Flight routing is 4D problem, 2D in horizontal direction, 1D in vertical, and 1D in time. With our algorithm, for a fixed altitude, we are able to produce a wind optimal route in horizontal direction with guaranteeing that the trip will be finished on time.

From long term expectation, next main task for the project will be find the optimal route in vertical direction, and combine with horizontal profile, so that we are able to combine the horizontal work with vertical profile, develop an algorithm which can produce 4D optimal trajectory. By combining the horizontal and vertical profile, we will be able to use more track wind field (wind is different at each altitude) and optimal altitude (remember that fuel consumption function also depend on altitude, there is optimal altitude exists for specific weather conditions, aircraft speed and weight). Therefore we are expecting to save more fuel in 4D optimization. This project is planned for two years, because of the huge data, big amount of work, and technical restrictions such as no efficient solvers available for MIP (the current solvers either slow or very restrictive.

From short term, we have few aspects for horizontal optimization profile need to be improved further. First, there is still nonlinearities in our model which slowing down the solving procedure. In model 6.3, we still have quadratic constraint (6.15) which we can be linearized by conic linearization but only for the cases to speed up. In next step we would like to try SOS Type 2 method, to approximate the quadratic functions by piecewise linear functions. This linearization procedure may will speed up the algorithm, and also enable us to use CPLEX (only be able to solve MILP, can't handle with either nonlinear bijective functions or constraint).

Secondly, we want to connect the algorithm with preprocessing part (mapping, generating wind data, fuel consumption data, triangle ordering), produce an interface that just with one comment, telling the location of departure airport and arrival airport to the algorithm, we can have the optimal trajectory in few seconds.

Third, evaluate the error for preprocessing part. Above the error estimation only given from model side. We didn't consider the error coming from mapping, next step we would like to find out how big error we are introducing by mapping the earth surface onto the plane, this will help us improve the accuracy.

Forth, currently, we are using the static wind. However, wind change over time, we have wind data from Lufthansa measured every three hour. This means, if we are having longer trip than three hours, wind may change. Therefore, we need to consider to use dynamic wind. This might have crucial impact if we have longer trip. For example, Flying from China to the United states, if the trip take 13 hours, using dynamic wind would be the best choice.

So this work may take 1 year to 2 year to finish all. Hoping we can get a good solution that Lufthansa System can use and save more fuel. We are looking forward to get Free-Flight practically used in future.

- [1] Airbus global market forecast future journeys 2013-2032.
- [2] Andrea Peter. Ein MILP, ein MINLP und ein graphentheoretischer Ansatz Für die Free-Flight Optimierung. Diplomarbeit. 2007.
- [3] Hok K.Ng, Banavar Sridhar, Shon Grabbe. Optimizing Aircraft Trajectories with Multiple Cruise Altitude in the Presence of Winds. 2010.
- [4] Alexander Martin, Armin Fügenschuh, Susanne Moritz. Computational integer programming and cutting planes. Preprint 2221, Technical University Darmstadt, 2002.
- [5] H.Marchand, A.Martin, R.Weismantel, and L.A.Wolsey. Cutting planes in integer and mixed integer Programming. *Discrete Applied Mathematics*, 123/124:391-440, 2002.
- [6] E.M.L.Beale, J.J.H. Forrest. Global Optimization Using Special Ordered Sets. *Mathematical Programming*. 1975.
- [7] E.M.L.Beale. Branch and Bound Methods For Mathematical Programming Systems. *Analysis of Discrete Mathematics*. 1979.
- [8] Susanne Moritz. A Mixed Integer Approach for the Transient Case of Gas Network Optimization. PhD thesis, TU Darmstadt, 2007.
- [9] Ashish Singhai. Optimal triangulation and mesh generation, august 1994.
- [10] A.Fügenschuh, A., Nagy, C., Peter. Modelling alternatives for free-flight optimization. Technical report. Fachbereich Mathematics, Tu Darmstadt
- [11] Wilson. Polyhedral Methods for Piecewise-Linear Functions. PhD thesis, University of Kentucky , 1998.
- [12] Paolo Dellolmo and Guglielmo Lulli. A new hierarchical architecture for air traffic management: Optimisation of airway capacity in a free flight scenario. *European Journal of Operational Research*, 144(1):179-193, 2003.
- [13] Roger Fletcher, Sven Leyer. Solving mixed integer non-linear programs by outer approximation. *Math. Programming*, 66(3, Ser. A):322-349, 1994.
- [14] Mora-Camino F., Hagelauer P. A soft dynamic programming approach for on-line aircraft 4D-trajectory optimization. *European Journal of Operational Research*, 107:87-95(9), 16. Mai 1998.
- [15] Jimmy Krozel. Free flight research issues and literature search. *NASA Ames Research Center Moett Field, CA 94035*, 2000.

- [16] Jon Lee and Dan Wilson. Polyhedral methods for piecewise-linear functions. I. The lambda method. *Discrete Appl. Math.*, 108(3):269-285, 2001.
- [17] Clyde F. Martin Magnus Egerstedt. Optimal trajectory planning and smoothing splines. *Automatica*, 37(issue 7):1057-1064, July 2001.
- [18] Nicoletta De Francesco Mieke Massink. Modelling free flight with collision avoidance. 2001.
- [19] Alexander Martin, Markus Möller, Susanne Moritz. Mixed integer models for the stationary case of gas network optimization. *Math. Program.*, 2006.
- [20] Lorenz T.Biegler, Andrew R. Conn, Gerard Cornuejols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, Francois Margot, Nicolas Sawaya, Pierre Bonami, Andreas Waechter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*. 2008.
- [21] D. Wilson. Polyhedral Methods for Piecewise-Linear Functions. PhD thesis, University of Kentucky, 1998.
- [22] M.Padberg Approximating separable nonlinear functions via mixed zero-one program. *Operations Research Letters*, 27:1-5,2000
- [23] Aharon Ben-Tal and Arkadi Nemirovski *On polyhedral approximations of the second-order cone*. May 30 1999.
- [24] I.R. de Farias A.B. Kehe and G.L.Nemhauser. *Models for representing piecewise linear cost functions*. Technical report, 2002.
- [25] E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *Proceedings of the Fifth International Conference on Operations Research, Tavistock Publications*, pages 447-454, 1970.
- [26] Armin Fügenschuh. Nonlinear Mixed Integer Programming-The MILP Perspective. Technical Report. 2014
- [27] Aharon Ben-Tal, Arkadi Nemirovski. On polyhedral approximations of the second-order cone. 1999.
- [28] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.
- [29] M.W. Padberg. *Linear Optimization and Extensions*. Springer, 1995.
- [30] Christos H. Papadimitriou. *Combinatorial Optimization*. Prentice-Hall, 1982.

-
- [31] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.
 - [32] H.P. Williams. *Model building in mathematical programming*. Wiley, 1990.
 - [33] L.A. Wolsey. *Integer Programming*. Wiley, 1998.
 - [34] J.Cole Smith, Z.Caner Takkin. A tutorial Gide to Mixed-Integer Programming Models and Solution Techniques. March 26,2007.
 - [35] Armin Fügenschuh, Zhi Yuan. Free-Flight routing: The problem definition. Technical report. 2014.

