Ingénieur ENAC Majeure SITA_IA
Double diplôme Master Recherche opérationnelle

Céline Demouge

IENAC18

End of Studies Project Memoir

05/09/2021

Finding geodesic trajectories thanks to propagation methods

2021

Acknowledgments

First, I acknowledge the welcome of all the members of the OPTIM team at the ENAC.

I thank my supervisor, Daniel Delahaye, head of the OPTIM team, for his help and sharing of his expertise as well as the time spent listening to me despite his busy schedule. Thanks to his confidence, I was able to go to the end of my ideas and to have been free in the research. I gratefully recognize the help of Marcel Mongeau, teacher-researcher and my academic supervisor, for his time and for his help, especially when writing reports.

I would also like to thank Banavar Sridhar, Senior Scientist at NASA, for his time and his always pertinent remarks.

I am grateful for Andréas Guitart, PhD student, help. Thank you for answering everyday questions and helping me when I had problems.

I am fortunate to have been a part of a trainees team composed by Lucas Ligny and Alexis Brun. Thank you for the daily work atmosphere and for the mutual support.

Abstract

In a context of worldwide development of air transport, air pollution from aircraft is a major issue. The Covid-19 crisis and the response of governments to support air transport have put this issue back at the center of concerns. It is therefore necessary to develop solutions to reduce the environmental impact of aircraft by reducing their fuel consumption, especially since fuel is the largest expense for airlines. The objective of the presented study is to develop an algorithm to determine the wind optimal trajectory that can be flown by an aircraft in its cruise phase. Answers have already been given for this problem but have various disadvantages, notably the consideration of operational constraints in the plane and in 3D space. Two solutions are proposed to answer the problem in 2D and 3D, both based on algorithms from robotics: FMT* and RRT*. The literature shows that these algorithms have been little used for aircraft but rather for drones and rarely in the case of the presented study (airliner cruise phase). The proposed solutions integrate the consideration of great circle distances and aeronautical constraints, in particular on the turn radius and the aircraft ability to climb. The results show that the algorithms can compute a good quality solution taking into account the mentioned constraints with satisfactory computation times.

Keywords— trajectory, wind, 3D, RRT*, FMT*

Résumé

Dans un contexte de développement mondial du transport aérien, la pollution atmosphérique due au trafic aérien est un enjeu majeur. La crise du Covid-19 et la réponse des Etats pour soutenir le transport aérien ont remis cette question au centre des préoccupations. Il devient alors nécessaire de développer des solutions permettant de réduire l'impact environnemental des avions en réduisant par exemple leur consommation de carburant, d'autant plus que le carburant présente la plus grande dépense des compagnies aériennes. L'objectif de l'étude présentée est de développer un algorithme permettant de déterminer la trajectoire de vent optimal volable par un avion de ligne dans sa phase de croisière. Des réponses ont déjà été apportées pour ce problème, mais présentent divers désavantages notamment le manque de prise en compte de contraintes opérationnelles dans le plan et dans l'espace 3D. Deux solutions sont proposées pour répondre au problème en 2D et en 3D toutes deux basées sur des algorithmes issus de la robotique : FMT* et RRT*. La littérature montre que ces algorithmes ont peu été utilisés pour l'aéronautique, ils l'ont été pour des drones mais rarement dans le cas de l'étude menée ici (avion de ligne en phase de croisière). Les solutions proposées intègrent la prise en compte de distances orthodromiques et de contraintes aéronautiques en particulier sur les rayons de virage et les capacités de montée de l'avion. Les résultats montrent que les algorithmes permettent de calculer une solution de bonne qualité en prenant en compte les contraintes mentionnées dans des temps de calcul satisfaisants.

Mots Clés— trajectoire, vent, 3D, RRT*, FMT*

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Air transport has seen many technical and technological evolutions over the last few decades, enabling a high level of safety to be achieved while allowing significant growth. Aircraft and ground systems (work tools for air traffic controllers, airlines, airport operators, etc.) have been the subjects of these developments, which have made air travel the safest mode of transportation in the world, and in a rapidly expanding market.

For this growth to continue, the air transport industry must now turn its attention to the environment. It must continue to guarantee a high level of safety and be able to accept growth like that of the 2010s while reducing its environmental impact in a sustainable development approach. This approach is essential to ensure the sustainability of the sector and is encouraged by the authorities, in particular in Europe, through projects such as SESAR.

The Covid-19 crisis threatens the growth of air traffic and weakens the whole sector. Air transport actors must reinvent their ways of acting in order to recover the past demand. Today, the public is almost sure of the safety of air transport, which must now prove its ability to meet current and future challenges and in particular the environmental issue without decreasing the level of service offered so far. Thus, measures yielding to save fuel are welcome, and even necessary. Moreover, in these times of crisis, such measures also allow financial savings since fuel is a major source of expenditure for an airline.

This report first presents the work environment for this internship and then the context and the detailed objectives of the project. Then, a state of the art of what have been already done in the field of aircraft trajectory optimization is presented. According to this state of the art, several methods depending on the dimension of the problem have been implemented during this internship and they are presented then.

Chapter 1

Presentation of the work environment

This internship took place in the ENAC laboratory, more specifically in the OPTIM team of this laboratory. This chapter first presents a general overview of the host laboratory, the ENAC laboratory, before focusing more specifically on the host team. Finally, the working environment of the internship is presented.

1.1 General presentation of the hosting lab

The ENAC lab is characterized by a combination of scientific excellence and operational expertise, as well as experimental resources covering all air transport sector. Research activities allow ENAC to ensure the excellence and the precursory character of all its training courses, by integrating the latest scientific and technological advances.

The lab is divided into four research teams. It conducts four transverse research programs and host two technological platforms of international scope. These different parts are listed in Figure 1.1.

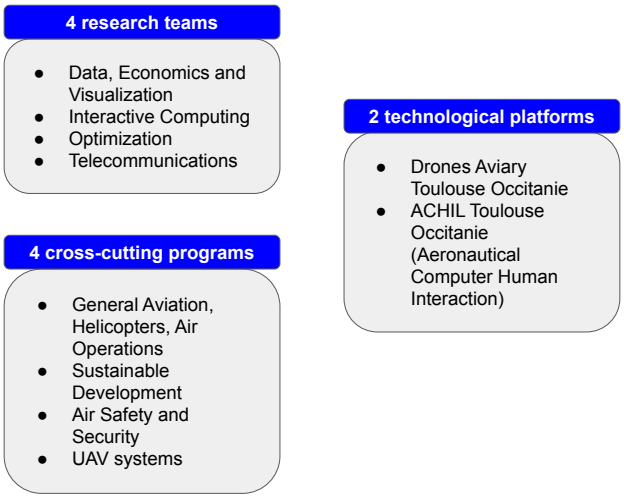


Figure 1.1: ENAC lab organization.

1.2 General presentation of the hosting team

The OPTIM team, headed by Daniel Delahaye (professor-researcher (HDR) in this team), my supervisor, is involved in optimization research, modelling and implementation of algorithms for problems mainly related to aeronautics. The combination of professional skills in the field of aviation and solid scientific knowledge (especially in mathematics and computer science) allows the team to solve the problems proposed. Among the team's skills, can be mentioned:

- **Mathematical modelling** of combinatorial, continuous and mixed optimization problems;
- **Deterministic and stochastic optimization methods**;
- **Artificial intelligence** especially artificial learning and constraint programming.

Here are some examples of projects handed by the team:

- **Aircraft trajectory optimization**: oceanic planning, UAV fleet planning, conflicts forecast and avoidance...
- **Airspace structure and complexity** analysis;
- **Air operations optimization** from several points of view (airports, airlines, air traffic control, manufacturers, passengers);
- Practical problems from other application areas: medicine, signal processing, social networks, transports, satellites...

1.3 Work environment during the internship

In practice, I work in collaboration with Lucas Ligny (IENAC 18 SAT) who is also doing his End of Study Project in the OPTIM team on a subject related to mine: trajectory optimization for emergency landing trajectories. He has to take into account many obstacles and must have a computation time very small. Likewise, another trainee, Alexis Brun (IENAC 18 OPS) is also working with us, but on a different subject. A PhD student, Andréas Guitart, is also present on a daily basis to exchange and help us. His PhD subject is trajectory optimization for emergency landing, so he can help me with the subject of trajectory optimization and give me advice on different methods. Moreover, my supervisor, who has worked extensively on the subject and supervised PhD students working on this subject, is there to help me. Finally, I am in contact with Banavar Sridhar, Senior Scientist at NASA, who has worked on trajectory optimization, particularly taking into account wind and contrails, and has published several papers on the subject.

To follow my progress, weekly meetings are set up with Andréas Guitart and fortnightly meetings are set up with Daniel Delahaye and the two other trainees, Alexis Brun and Lucas Ligny, to exchange on the progress of each one. Moreover, meetings are scheduled with Banavar Sridhar every two weeks in order to have his point of view on our work.

This chapter having presented the working environment of the project, the next chapter explains in detail the problem addressed during this project.

Chapter 2

Project presentation

This chapter presents the general context of this End of Study Project before explaining the problematics under study and then expose the different objectives of the internship.

2.1 General context

In recent years there has been a growing concern about the environmental impact of aviation, firstly about CO_2 under the impulse of various public policies but also to reduce fuel costs. More generally, all the effects that could influence global warming are studied, especially condensation trails (also named contrails).

This End of Study Project is a preliminary project to a PhD in line with this dynamic, on trajectories optimization to minimize the overall environmental impact of the French air network, in particular by minimizing CO_2 and NO_x emissions but also by working on the avoidance of areas favorable to contrails. This preliminary project is focused on **finding wind optimal trajectories for cruise, yielding less CO_2 emissions**.

2.2 Problematics

The issue addressed during this internship is to compute a flight path for an aircraft in its cruise phase in 2D and 3D in order to take benefits from wind, that is to say to consume as little as possible or as a first approximation **to obtain the route with the lowest flight time**. Obstacles can be taken into account to do a preliminary work for future works on contrails avoidance.

2.3 Objectives

These problematics can be declined into different objectives:

1. Compute the wind optimal trajectory in a 2D plane with Cartesian coordinates;
2. Compute the wind optimal trajectory in a 2D plane taking into account great circle distances (Earth coordinates);
3. Compute the wind optimal trajectory in a 3D space for a cruise phase.

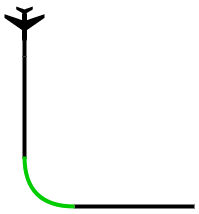
These objectives will be treated **for a single aircraft and will not take into account other traffic**.

Two cross-cutting objectives can be added:

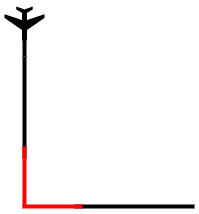
1. Avoid obstacles;
2. Compute flyable trajectories.

This project being placed in a cruising case, the objective 1 will not be treated at first. However, obstacles can still be encountered during a cruise, such as stormy areas.

Objective 2 is essential. Because an aircraft must be able to fly the computed trajectory, some constraints in the horizontal and vertical planes have to be taken into account. In the horizontal plane, some constraints on the turn radius have to be taken into account to obtain flyable turns, as shown by Figure 2.1. In the vertical plane, constraints on climb/descent slope must be respected and the flight levels computed must be cruise flight levels. For example, flying at FL 100 is not acceptable for a desired cruise around FL 360. Besides, the changes in flight level have to be penalized because it is not suitable for a cruise phase as shown by Figure 2.2. Changes in flight levels are accepted only if the benefit is high.

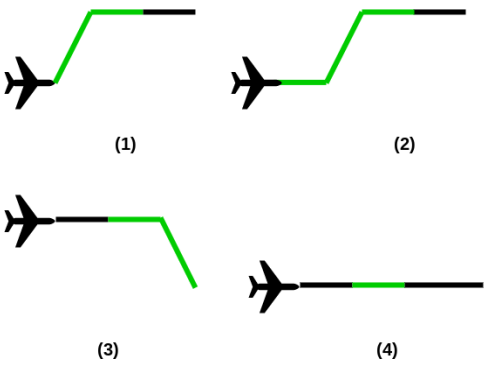


(a) Acceptable turn.

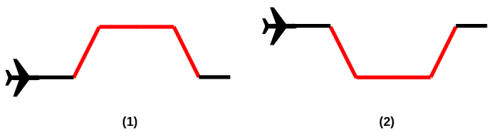


(b) Non-acceptable turn.

Figure 2.1: Turns have to be acceptable.



(a) Friendly vertical profiles.



(b) Unfriendly vertical profiles.

Figure 2.2: The flight level must be as constant as possible.

The next chapter presents a state of the art of the different existing methods which answer in part or completely the addressed problem.

Chapter 3

State of the art

This chapter presents a state of the art of the main works already done in the trajectories planning and optimization for a single aircraft, especially works oriented on wind optimal trajectories. First it presents methods using optimal control, then methods based on space discretization before focusing on fast marching methods.

3.1 Optimal control

The problem under study requires finding an optimal trajectory for an aircraft. This type of problem can, in the most intuitive way, thanks to the **equations of the flight dynamics**, be written as follows:

$$\min_{U} \quad J(X,u) \tag{3.1a}$$

$$\text{under} \qquad \dot{X}(t) = g(t,X,u) \tag{3.1b}$$

$$X(t_0) = X_0, \tag{3.1c}$$

$$X(t_f) \in X_f, \tag{3.1d}$$

$$\forall t \in [t_0, t_f], u(t) \in U. \tag{3.1e}$$

where X is the state of the aircraft, X_0 the initial state of the aircraft at time t_0, X_f the set of final states of the aircraft at time t_f, u the control applied to the aircraft, U the set of admissible control and $J(X,u)$ the criterion to be optimized, for instance inversely proportional to the wind encountered or proportional to the fuel consumption and/or flight time. g is a function defined from the **equations from the flight dynamics**. A problem of this type is called an **optimal control problem**. Several authors have modeled the problem of wind-optimal trajectories for one aircraft in this form. This is notably the case of Chakravarty [1], Sridhar *et al.* [2], or Palopo *et al.* [3] who used a method from Jardin and Bryson [4].

The oldest paper, *i.e.* that of Chakravarty [1] poses the problem using the equations of aircraft dynamics and chooses to minimize the following criterion:

$$J(X,u) = \int_{t_0}^{t_f} C_f f \, dt \tag{3.2}$$

where f is the fuel flow, and C_f the cost of fuel per liter per time step. Then, to solve this problem, the author uses **Pontryagin's principle**. This principle, published in [5], gives the

necessary conditions for optimality. Through the determination of an adjoint state $p : [t_0, t_f] \rightarrow \mathbb{R}^n$ and this principle, the problem results in a system of differential equations with an initial condition and a final condition.

The study from [1] is done in cruise but also in descent and climb, with or without wind. This article presents a complete study done on the Boeing 767, without mentioning any computation time. With the same idea, in 2011, Sridhar *et al.* [2], take up the modelling done by adding the impact of contrails. Thus, the objective to be minimized is in this case:

$$J(X, u) = \int_{t_0}^{t_f} (C_t + C_f f + C_r * r(x, y))\, dt \tag{3.3}$$

where C_r is the cost associated with contrails and $r(x, y)$ the associated penalty, for example, $r(x, y)$ is inversely proportional to the distance to an area predicted to be favorable for contrails. Moreover, a coefficient associated with time C_t is added, This allows the flight time to be minimized and therefore the fuel consumption to be reduced. The consumption was computed using **BADA database**. Finally, in 2010, Jardin and Bryson [4] use **the neighboring optimal control technique** particularly adapted to time-dependent parameters, as the wind can be.

The last article cited [4] mentions the **Zermelo problem** [6] which was proposed in 1931. The aim of this problem is to optimize the navigation of a boat to take the best advantage of the marine currents, a problem which is similar to the problem addressed in this project. Thus, it often serves as a test problem for methods implemented in the literature as it has been solved in some wind cases.

Let us summarize the positive and negative points of the optimal control approach. First, the advantages include:

- ✓ Natural modelling from flight dynamics equations;
- ✓ Reaching an optimal solution, which can be flown since it follows the laws of flight dynamics.

Drawbacks are:

- × Sensitivity to start and end points: if the computation is performed from a point A to a point B, it must be completely redone for a trajectory from A to a different destination point B';
- × The computation time can become high;
- × Need to know analytically all the parameters involved;
- × A flight path that is flyable is not necessarily acceptable: for example, the result may be a succession of turns with a radius of curvature acceptable to the aircraft, but the succession of turns will not be acceptable to the pilot.

Although this model seems to be the most intuitive and straightforward, it will not be used in the sequel as there are too many negative points compared to the number of positive ones.

3.2 Methods based on space discretization

Several methods choose instead to discretize the environment and work on the resulting graph, making the problem simpler and more similar to **well-known shortest path problems** or tree search. This graph is built from a discretization of the space and the cost of edges are computed thanks to the wind and according to great circle distances. Among these methods based on discretization, there are deterministic ones and non-deterministic ones, also known as sampling based methods.

3.2.1 Deterministic discretization methods

Deterministic methods search the shortest path (or the path with the lowest cost) on a graph in a deterministic way, that means they always do the same actions at each execution on the same problem instance and find the optimal path **on the graph**.

Different algorithms can then be used such as **Dijkstra** [7] or **A*** [8]. These methods can be very efficient but several problems arise:

- × The trajectory created is sub-optimal since the search space is restricted because it is discretized. Indeed the trajectory found is optimal regarding the graph but not considering the whole space;
- × The trajectory created is not be directly "flyable" and then additional work for smoothing the trajectory is required and therefore the optimum may move further away.

However, by working on the discretization of the space so that the loss of information is avoided as much as possible, **robust methods** can be implemented. Indeed, many uncertainties are due to the wind and its predictions, especially at small scales. This is done by Legrand *et al.* [9]. In this paper, **Bellman's algorithm** [10] is adapted to better fit the needs and the method is tested on transatlantic trajectories where wind is a major issue, using real wind predictions. Pre-processing is a particularly important issue for this kind of methods where the computation of the wind at each node of the grid, with possible interpolations, will be essential to the quality of the solution found. Moreover, this method can **easily take into account possible static obstacles and be adapted in 3D**. However, a major problem remains: the **time complexity** of the algorithm. Indeed, considering a grid such as the one shown in Figure 3.1, with $N = n^2$ nodes and about $N = n^2$ edges (a constant number for each node), Bellman's algorithm has a complexity in $O(N^2)$, and when moving to 3D, the complexity rising to the order of $O(N^3)$.

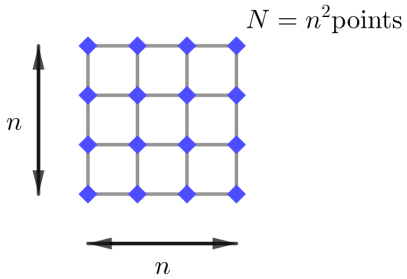


Figure 3.1: Square grid $n \times n$.

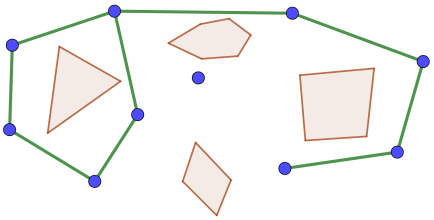
To conclude, even if discrete deterministic methods allow to have first approximations of trajectories by taking easily into account the distances between two points or possible obstacles, they do not give satisfactory results in the absolute.

3.2.2 Sampling based methods

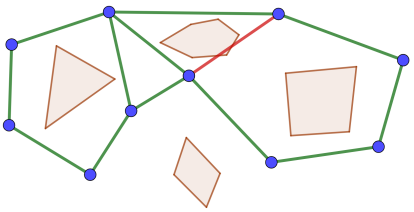
Other methods using graphs have been used. Research on this type of methods was first driven by robotics. The goal was initially to find a path, as short as possible, for a robot, avoiding obstacles. If all the information of the environment is taken into account, the time complexity of the algorithms to solve this problem is also very large, as mentioned before (cf Section 3.2.1). To overcome these issues, some methods have been developed to **sample the environment**, first by more or less randomly choosing a set of points, see for instance Simeon *et al.* [11] or Kavraki *et al.* [12]. These methods, named **Probabilistic RoadMaps (PRM)**, are based on two steps:

1. Creation of a graph from the sampling of the environment;
2. Finding the shortest path from one point to another.

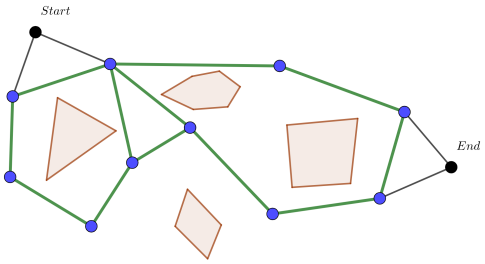
Figure 3.2 shows the process used by PRMs methods.



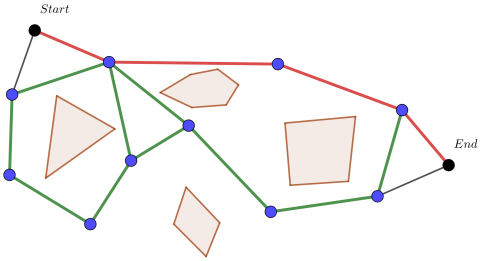
(a) Graph built at a given iteration. The graph is in green, the sample points in blue and the obstacles in red.



(b) Adding a node to the graph: edges are added with its neighbors when they do not meet an obstacle (green) and are not added if they do (red).



(c) Adding the start and end points of the trajectory to the graph.



(d) Search for the shortest path (in red) between the starting point and the end point in the graph by classical methods.

Figure 3.2: Probabilistic Roadmap principle.

PRM are based on random sampling and this can be problematic regarding the quality of the solution. To avoid relying too much on random sampling, other **sampling methods** have been developed, particularly **Rapidly-exploring Random Tree (RRT)** [13].

RRT

The goal of this type of algorithm is also to reduce the number of points in the environment, and to find the most interesting ones by **creating a graph in an efficient way**, to recover a maximum of useful information on the environment in order to find a path close to the optimal one while saving a lot of time. Figure 3.3 shows an example of a construction of an RRT from the center point of the square. It shows how **RRT expands naturally into unexplored areas**. It is this property based on a natural bias, called Voronoi bias, that makes this method effective.

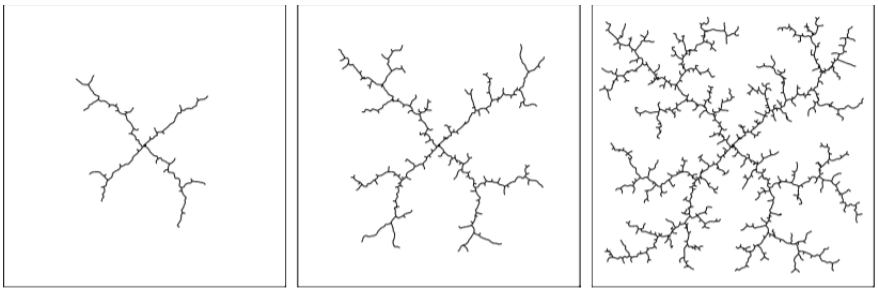


Figure 3.3: Example of RRT construction, from [13].

RRT has then been improved into **RRT*** [14] which has been proven to be **asymptotically optimal**, unlike RRT. Figure 3.4 shows the construction of an RRT and an RRT* from the center of the square to find the optimal path (in red) to the magenta area. RRT* gives a solution which is much better than the one provided by RRT.

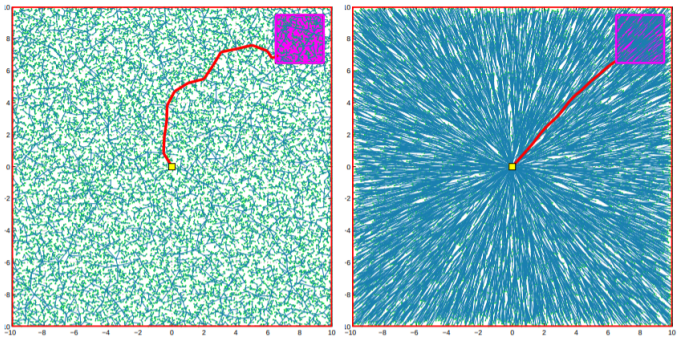


Figure 3.4: Comparison between RRT (left) and RRT* (right) to find a short path from the center to the magenta area, from [14].

RRT* has been used for aeronautical applications but not for the case under study during this project. In 2017, Pharpatara *et al.* [15] used RRT* to compute obstacle avoidance trajectories in 3D. This paper therefore shows that RRT* is easily usable in 3D. However, the authors consider the wind to be negligible given the speed of the aircraft. So their article is not directly relevant to this project, although it does give a lead to the study of obstacle avoidance and shows that the use of these methods for aircraft trajectories can be interesting. The 3D wind track using RRT was however studied earlier, for glider-type drones by Chakrabarty and Langelaan [16]. In their paper, encouraging and interesting results are shown for maximizing the glider's flight time or getting it to reach a given point by using air currents.

FMT*

A new improvement to RRT* is **Fast Marching Tree (FMT*)** [17]. This method executes concurentially, and thus "at the same time", the two steps of RRT*. Indeed, RRT* first builds the graph and then tries to optimize locally the connections while FMT* builds the graph by choosing the locally optimal connection. This saves a lot of time. Another idea is added to save even more time: FMT* performs a dynamic "lazy" computation, *i.e.* the algorithm initially ignores obstacles and if a locally-optimal edge is created and crosses an obstacle, then it ignores it.

T-RRT

When the cost function is continuous or when the space can be covered with a cost map, a method based on RRT named **Transition-based RRT (T-RRT)** [18] can be used. It adds to the classical RRT a **stochastic decision** to accept or not a transition in the built tree:

$$P(\text{accept transition from } i \text{ to } j) = \begin{cases} 1 & \text{if } c_j < c_i \\ \exp(\frac{-(c_j - c_i)/d_{ij}}{K * T}) & \text{else.} \end{cases} \tag{3.4}$$

where c_i is the cost in i, d_{ij} the Euclidean distance between node i and node j, T the temperature which evolves during the exploration of the space, and $K = \frac{c_{init} + c_{goal}}{2}$ the average between the cost of the starting point and the cost of the ending point.

To better understand the purpose of this type of method, the paper [18] takes the analogy with a mountainous environment and it appears that valleys are favored as shown by Figure 3.5.

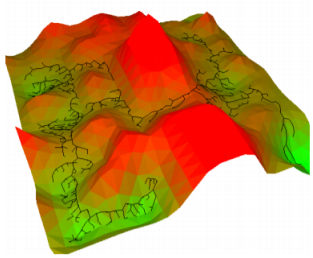
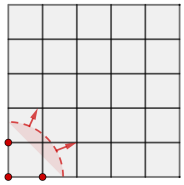


Figure 3.5: Example of mountainous environment exploration by T-RRT, from [18].

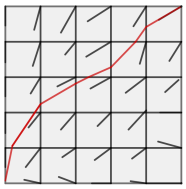
To conclude, these methods seem interesting because they can answer the problem addressed even if it has only been rarely used for aeronautics and not in the case under study here. Nevertheless, the quality of the sampling of the environment is essential. Moreover, the trajectories can be sub-optimal because of the sampling.

3.3 Fast marching methods

From the analogy of light propagation in an open environment and the Eikonal equation governing this propagation or other Hamilton-Jacobi type equations, very efficient methods have been created, called propagation methods. The principle of propagation methods is therefore **to propagate a "wave"** (or equivalent front) on a grid and then **back propagate the gradients obtained on this grid to draw a continuous trajectory** as shown by Figure 3.6.



(a) Propagation of the front.



(b) Gradient back propagation.

Figure 3.6: Principle of propagation methods.

The fast marching [19] and ordered upwind [20] methods are based on an older method developed by Sethian called level set method [21]. These methods were developed to solve stationary Hamilton-Jacobi equations, *i.e.* equations of the type :

$$||\nabla u(X)|| = f(X, \frac{\nabla u(X)}{||\nabla u(X)||}) \tag{3.5}$$

where X is the state vector, u is the optimal cost representing the minimum arrival time, and f is the "slowness" of the domain at any point, for example the inverse of the propagation speed of the front.

These methods can be used in the context of this End of Studies project since the optimal control problem 3.1 can be written under the form of a particular Hamilton-Jacobi equation since it becomes stationary.

In the case of an isotropic medium, *i.e.* when the characteristics depend only on the position and not on the directions of space, this equation can be rewritten in another form to obtain an Eikonal-type equation :

$$||\nabla u(X)|| = f(X) \tag{3.6}$$

The fast marching type approach is particularly adapted for this type of equations. However, our project does not fit in such an isotropic context since the wind speed does depend on the direction. It would have been possible to implement fast marching methods in an anisotropic environment, citing the work of Mirebeau *et al.*[22], but the underlying mathematics are complex. Another method, simpler and already proven to be efficient, is preferable: the **ordered upwind** method. It has been used in the PhD thesis [23] in the 2D case.

For the transition to a 3D space, the question that arises is what mesh (computing grid) should be used. For this, different authors have studied structures. In her PhD thesis [23], Girardet used a triangular grid and has shown that this structure is effective. Then, a natural switch would be to use a tetrahedral mesh as done by Lelievre *et al.* in [24] in the case of a seismic study. However, Girardet mentions in her conclusions [23] a time complexity issue for a 3D switch of the algorithm as it is.

For taking into account 3D spaces and also both static and dynamic obstacles, other methods using an analogy with wave propagation could be used such as **Light Propagation Algorithm (LPA)** introduced by Dougui in her PhD thesis [25]. This algorithm was developed to avoid a major problem induced by the initial fast marching method: the time complexity becomes too high when the dimension increases. For instance, the execution would be too high to compute

trajectories in 3D spaces, even worse in 4D spaces (when dynamic environments are taken into account, such as moving obstacles or dynamic wind). However, this algorithm has a major drawback for the problem addressed here: it is not adapted to the anisotropic case and therefore is not suitable in the case of a wind field. If this algorithm has to be used during this project, it will have to be adapted in order to take into account an anisotropic propagation.

Let us detail the underlying idea of this algorithm, first in the case of static obstacles. The goal is to create a tree from a source point by "emitting light rays" representing the propagation of these rays and then to find the shortest path between the start and the destination thanks to this tree as shown by Figure 3.7. A time step, dt, and an angle, $d\gamma$, between each emitted light ray are fixed at the beginning of the algorithm by the user.

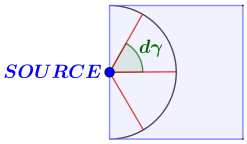


Figure 3.7: LPA - Emission of rays from the source point.

Each branch of the tree is then constructed according to the distance travelled by the emitted ray during the time step dt. This distance travelled then depends on the speed of propagation locally, as the speed of propagation of light depends on the medium of propagation, which depends on the index, n, of the medium in which the light propagates[1]. In this analogy, the speed of fronts is driven by the optimal speed of the aircraft at a given point.

The next node chosen to continue developing the tree is chosen according to a Branch-and-Bound[2] principle thanks to computed lower bounds on the remaining distance. Obstacles are then taken into account through the indices. An index close to 1 is used in free space and an index greater than 1 is considered in areas where there are obstacles. This way of doing things makes it possible to take into account soft obstacles, *i.e.*, zones that one would prefer not to cross but which are not impossible to cross, for example zones favorable to condensation trails. In this case, an index greater than 1 but still close to 1 is used. In the case of moving obstacles, it is sufficient to move to a dimension greater than the working space: from $\mathbb{R}^n$ to $\mathbb{R}^n \times [0, +\infty[$ so that the time dimension is represented. The search for a geodesic is done in space-time with added constraints to match the different points with a durations depending on the speed and the path traveled.

To conclude on these propagation methods, they have various advantages for the problem at hand:

- ✓ Acceptable time complexity: generally $O(NlogN)$, where N is the number of grid points used;
- ✓ Obtaining a continuous trajectory;
- ✓ Possibility of managing static or dynamic obstacles;
- ✓ Work in 2D space as well as in 3D space;

[1] $n = c/v$, where c is the speed of light in vacuum, and v is the speed of propagation of light in the medium.

[2] Branch-and-Bound algorithm relies on the exploration of a tree which represent subsets of the solution set. Before the exploration of a branch, an upper and/or a lower bound of the solution in this branch is estimated and can allow to discard this branch if the solution will not be better than the best one.

✓ Consideration of wind already studied and proven to be relevant.

However, as for discrete methods, the quality of the results is highly sensitive to the choice of the computation grid (the mesh).

Computation grid (mesh)

In her PhD thesis [23], Girardet uses an irregular triangular mesh, as illustrated on Figure 3.8.

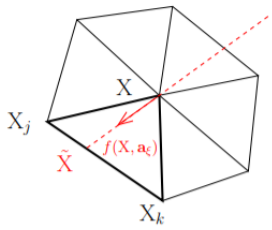


Figure 3.8: Irregular triangular mesh used by Girardet, from [23].

On the one hand, the fact that the mesh is triangular increases the information given, and so can increase the quality of the solution. On the other hand, the fact that the mesh is irregular allows one to take into account more complex geometries and also yields a **smoother trajectory** than when relying on a regular mesh. This type of mesh is named **unstructured mesh**.

3.4 Conclusion

At the end of this state of the art, ordered upwind seems to be the most efficient way to solve the problem addressed in 2D. However, it has been said less efficient especially because of time complexity in 3D. To solve the problem addressed in 3D, sampling based methods seem to be the best methods a priori. Indeed, they are able to compute trajectories very quickly and to take easily into account constraints that could be useful to make a trajectory flyable. However, they have been used very little in aeronautics except in the case of drones, a case that can be quite different from the case of an aircraft.

The problem addressed during this project is treated in two steps. The next chapter treats the problem in 2D using ordered upwind and sampling methods, and chapter 5 treats the problem in 3D using sampling methods.

Chapter 4

2D problem resolution

This chapter presents the treatment of the problem in a 2D space. After having presented the problem statement, it presents two methods used to solve it.

4.1 Problem statement

The goal of the 2D problem is to find a trajectory from a start point to a goal point minimizing the flight time between these two points. For this, the **wind** must be taken into account. Moreover, since the problem is located in the cruise phase and therefore the distances are large, the **great circle distances** must be taken into account. Finally, constraints in the horizontal plane must be considered to obtain a flyable trajectory. In particular, **the turn radius must be acceptable** for a standard airliner. A typical minimum radius of 2 NM is used in the sequel.

4.2 First method used: ordered upwind

This section presents a first method to solve the 2D problem. This method is the one that emerges from the state of the art which presents ordered upwind as the most adequate method to solve this problem, even if in 3D this algorithm is no longer the most adequate.

4.2.1 Presentation of the algorithm

The ordered upwind algorithm is based on the **propagation principle** already explained in the state of the art (cf Section 3.3). It has been developed by Sethian and Vladimirsky in [20] and adapted to aeronautical case by Girardet in her PhD thesis [23]. Based on fast marching method, it adapts it in the case of **anisotropic** propagation, that is to say the case in which the propagation speed does not depend only on the point location but also on the propagation direction. For instance, an environment where a mobile can move at a constant speed anywhere but must avoid obstacles is an isotropic case. At the opposite, a water environment with currents and, so in which the speed of the mobile depends on these currents, is an anisotropic case.

The purpose of the algorithm is to compute the cost function u on each point of a mesh and to keep in memory for all these points the gradients that allowed the algorithm to compute the cost. These gradients will be used in a second step to compute the optimal path from the

initial point X_0 (from which the propagation starts) to any other point in the mesh. The largest distance between two adjacent nodes in the mesh is noted h. It is set by the user. Two points X_1 and X_2 are therefore adjacent if $||X_1 - X_2|| \leq h$. Let f_1 the minimum speed of the aircraft and f_2 its maximum speed. At each step of the execution, each node of the mesh is in one, and only one, set among the following three node sets:

- *Accepted*: nodes for which u has already been computed, and its value is definitely fixed.
- *Considered*: nodes for which u has already been computed, and its value is not definitely fixed.
- Far: nodes not yet visited.

Let us define two other node sets:

- *AcceptedFront*: points from *Accepted* which are adjacent to at least one point from the set Far or *Considered*.
- AF: set of segments whose ends are in *AcceptedFront* and are adjacent points and adjacent to a point in the *Considered* set.

The Figure 4.1 shows this different sets.

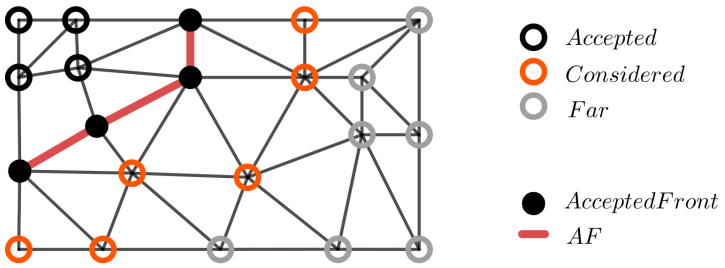


Figure 4.1: Sets defined for ordered upwind.

Finally, for each point X, a set $NF(X)$ is defined as follows:

$$NF(X) = \{(X_j, X_k) \in AF \mid \exists \tilde{X} \in (X_j, X_k) \; s.t. \; ||\tilde{X} - X|| \leq h\frac{f_2}{f_1}\} \tag{4.1}$$

That is to say $NF(X)$, called "Near Front" of X, is the set of AF segments closest to X, segments at a distance of $h\frac{f_2}{f_1}$ or less from X.

Algorithm 1 shows the ordered upwind algorithm in pseudo-code.

Algorithm 1: Ordered upwind algorithm from a start point X_0.

Input: Far which contains all nodes of the mesh, X_0 the origin node ($X_0 \in Far$

`// Initialization`

for $X \in Far$ **do**
  $u(X) \leftarrow +\infty$
end
$Accepted \leftarrow \{X_0\}$, $u(X_0) \leftarrow 0$
$Considered \leftarrow \emptyset$
for $X \in Far$ *s.t.* X *is adjacent to* X_0 **do**
  $v(X) \leftarrow \min_{X_i \in NF(X)} v_{X_i}(X)$
  $Considered \leftarrow Considered \cup \{X\}$
end

`// Propagation`

while $Considered \neq \emptyset$ **do**
  $\tilde{X} \leftarrow \arg\min_{X \in Considered} v(X)$
  $Accepted \leftarrow Accepted \cup \{\tilde{X}\}$, $u(\tilde{X}) \leftarrow v(\tilde{X})$
  for $X \in Far$, *adjacent to* $\tilde{X}$ **do**
    $Considered \leftarrow Considered \cup \{X\}$
    $v(X) \leftarrow \min_{X_j X_k \in NF(X)} v_{X_j X_k}(X)$
  end
  for $X \in Considered$, *adjacent to* $\tilde{X}$ **do**
    $v(X) = \min(v(X), \min_{X_j X_k \in NF(X)} v_{X_j X_k}(X)))$
  end
end

The cost of a point is updated all along the propagation thanks to a trial value v which becomes the definitive cost value u when the point becomes accepted. This trial value is updated thanks to the Equation 4.2.

$$v(X) = \min_{X_j X_k \in NF(X)} v_{X_j X_k}(X) \tag{4.2}$$

In the equation 4.2, $v_{X_j X_k}(X)$ consists in the **resolution of Hamilton-Jacobi equation** thanks to the points X_j and X_k. As done by Girardet in her PhD thesis [23], it has been chosen to use a quadratic reformulation of the problem to evaluate $v_{X_j X_k}(X)$ and so to avoid a local minimization. $v_{X_j X_k}(X)$ is the solution of the following quadratic equation $A v_{X_j X_k}^2 + B v_{X_j X_k} + C = 0$ where:

$$A = V_a^2 \langle P^{-1}\alpha, P^{-1}\alpha \rangle - \langle P^{-1}\alpha, W \rangle \tag{4.3}$$

$$B = 2 * V_a^2 \langle P^{-1}\alpha, P^{-1}\beta \rangle - 2 * \langle P^{-1}\alpha, W \rangle * (\langle P^{-1}\beta, W \rangle + 1) \tag{4.4}$$

$$C = V_a^2 \langle P^{-1}\beta, P^{-1}\beta \rangle - (\langle P^{-1}\beta, W \rangle + 1)^2 \tag{4.5}$$

where V_a is the airspeed of the aircraft, W is the wind vector and

$$\alpha = \begin{pmatrix} \frac{1}{||X-X_j||} \\ \frac{1}{||X-X_k||} \end{pmatrix} \tag{4.6}$$

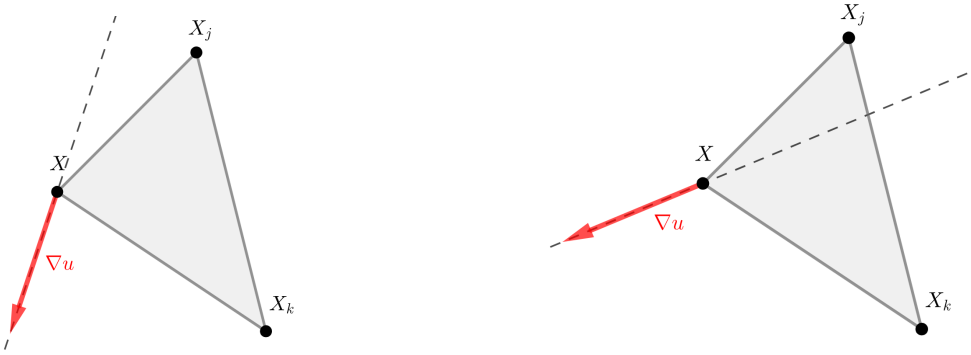
$$\beta = \begin{pmatrix} \frac{-u_j}{||X-X_j||} \\ \frac{-u_k}{||X-X_k||} \end{pmatrix} \tag{4.7}$$

$$P = \begin{pmatrix} P_j \\ P_k \end{pmatrix} \text{ with } P_i = \frac{X - X_i}{||X - X_i||} \tag{4.8}$$

From this trial value, an estimation of the gradient can be computed:

$$\nabla u(X) \approx P^{-1}(\alpha v_{X_j X_k}(X) + \beta) \tag{4.9}$$

Nevertheless, all this computation relies on one principle: the causality principle. This principle is based on the following affirmation: "If $u(X)$ is computed from X_j and X_k then $u(X) \geq \max(u(X_j), u(X_k))$". To satisfy this principle, the gradient must be inside the triangle XX_jX_k as shown by Figure 4.2.



(a) Causality principle violated.

(b) Causality principle respected.

Figure 4.2: Causality principle.

If the causality principle is violated, then the computation is directly done by the edges of the triangle XX_jX_k:

$$v_{X_j X_k}(X) = \min(v_{X_j}(X), v_{X_j}(X)) \tag{4.10}$$

where:

$$v_{X_i}(X) = \frac{||X - X_i||}{f(X, \frac{X-X_i}{||X-X_i||})} + u(X_i) \tag{4.11}$$

where $f(X, a)$ is the ground speed at the point X in the direction a. The ground speed is the speed of the aircraft relative to the ground, *i.e.* the air speed (speed relative to the air mass) plus the wind vector, as shown in Figure 4.3.

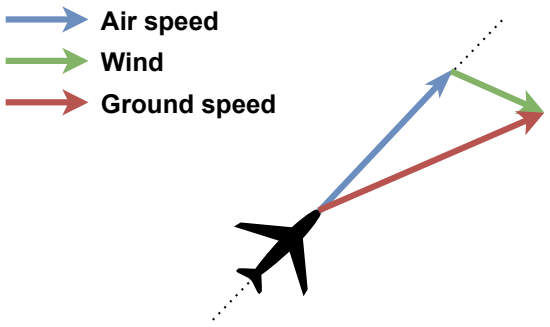


Figure 4.3: Speed triangle.

For instance, if the causality principle is violated and if $v_{X_j}(X) > v_{X_k}(X)$, the computation will be done from X_k that is to say that the cost in X is the cost in X_k plus the flight time from X_k to X and the gradient vector is supported by the line between X_k and X. The Figure 4.4 shows this situation.

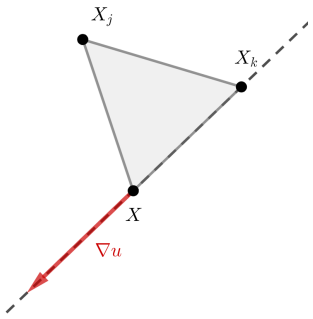


Figure 4.4: Causality principle is violated and $v_{X_j}(X) > v_{X_k}(X)$ so the computation is done from X_k.

4.2.2 Implementation

This subsection presents the different implementation choices which have been made.

Language chosen This algorithm has been implemented **in Java** in order to take advantage of performances of a compiled language on iterative programs associated to coding facilities offered by a high-level language.

Complexity and priority queue Ordered upwind is said in complexity $O(\frac{f_2}{f_1} * NlogN)$ with N the number of nodes in the mesh, f_1 the minimum speed of the aircraft and f_2 its maximum speed. Indeed, if a priority queue is used to implement this algorithm, the withdrawal of the lowest cost value from the set *Considered* is in a constant time operation and adding value to the set is in $O(logN)$. A priority queue object is already available in Java and follows these complexities.

Mesh used As a first development it has been chosen to implement a regular squared mesh as shown by the Figure 4.5. As mentioned in the state of the art, this mesh is not the one used by Girardet in her PhD thesis, however even if the performances will not be the best ones with this mesh, first results can be obtained.

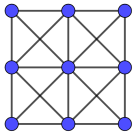


Figure 4.5: Mesh used for ordered upwind implementation.

4.2.3 Taking into account great circle distances

The algorithm is supposed to be used in cruise phase so the trajectories computed are long. Consequently, great circle distance has to be taken into account and not the Euclidean one. Indeed, between two points on the Earth, the shortest path is not the straight line (also named rhumb line[1]) but another named "great circle" as shown by the Figure 4.6.

Figure 4.6: Comparison between the straight line and the great circle trajectory between Paris and New York.

The great circle distance between two points $A(lat_A, lon_A)$ and $B(lat_B, lon_B)$ can be computed by the Equations 4.12 and 4.13.

$$d_{ortho} = R * c_{radians}(km) \tag{4.12}$$
$$= 60 * c_{degrees}(NM) \tag{4.13}$$

where $R = 6371$ km is the Earth radius, $c_{radians}$ and $c_{degrees}$ are respectively c expressed in radians and in degrees with:

$$c = \arccos(\sin(lat_A) * \sin(lat_B) + \cos(lat_A) * \cos(lat_B) * \cos(lon_B - lon_A)) \tag{4.14}$$

These formulas allow to compute the exact great circle distance between two points however it is time-consuming. Between two near points, the great circle trajectory can be approximate

[1] Arc crossing all meridians of longitude at the same angle, that is, a path with constant bearing as measured relative to true north.

by the straight line as shown in Figure 4.7. It has also been decided to do this approximation between two adjacent points of the mesh and to multiply the airspeed depending on the propagation direction and the location of the point under study as shown by Figure 4.8 and Equation 4.15.

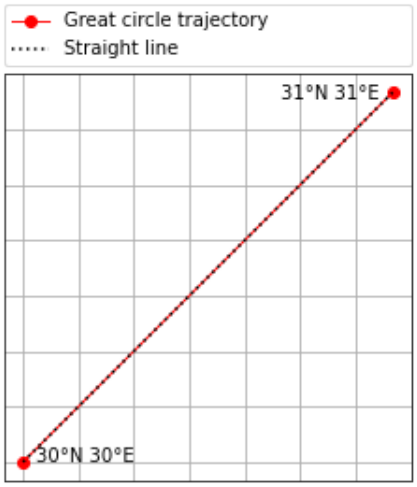


Figure 4.7: Straight line and great circle trajectory between two near points.

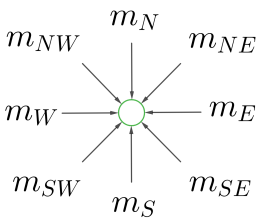


Figure 4.8: Different multiplying factors depending on the propagation direction.

$$m_{direction} = \frac{\text{distance on the Cartesian plane} \approx \text{distance at the equator}}{\text{distance on the Earth}} \tag{4.15}$$

Computation details for each multiplying factor can be found in Appendix A.

4.2.4 Results

This subsection presents different results obtained thanks to ordered upwind algorithm. The results presented throughout this report were obtained with a PC equipped with an Intel Core i5-6200U 2.30 GHz processor and 6 GB of RAM. First, it focuses on Zermelo's benchmark, then on great circle trajectory computation without wind. These particular cases were taken to evaluate the performance of the algorithm on known cases.

Zermelo's benchmark

As mentioned in the state of the art (cf Section 3.1), a benchmark problem that can be used to evaluate algorithms computing wind optimal trajectories is the Zermelo's problem. It

was first posed in 1931 by Ernst Zermelo in [6]. The goal of the problem is to minimize the travel time from point A to point B of a boat subjected to water currents which can easily be assimilated to the wind in the case of the project under study and the boat to an aircraft. In some particular cases, this problem has been resolved. More details can be found in Appendix B. In the following, a particular case of wind will be studied: the **case of a linear wind**.

The problem is treated using **Cartesian coordinates**. Let $W(x, y)$ be the wind vector.

$$\forall (x, y), W(x, y) = \begin{pmatrix} \frac{V_a}{10} * y \\ 0 \end{pmatrix} \tag{4.16}$$

where V_a is the airspeed of the aircraft.

Figure 4.9 shows the wind encountered in this setup.

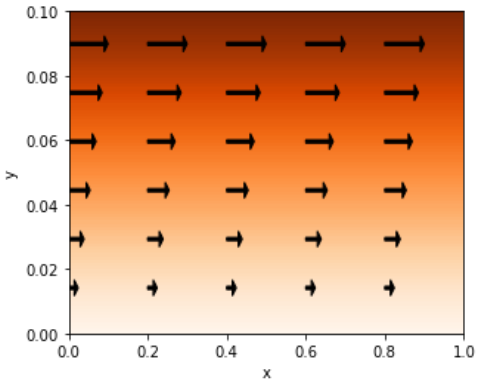


Figure 4.9: Linear wind case for Zermelo problem.

Figure 4.10 shows the computed trajectories with different mesh sizes and the Table 4.1 shows the different execution times to obtain them.

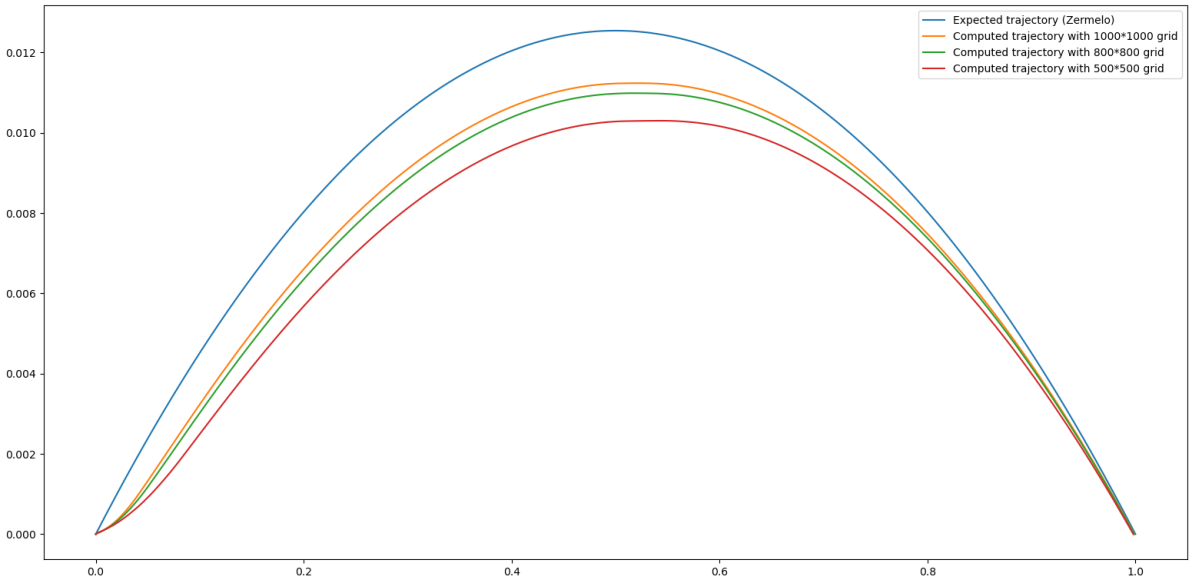


Figure 4.10: Results from ordered upwind - Zermelo problem in the case of linear wind.

Mesh size	Number of points	Computation time (s)	Relative error on flight time (%)
500 × 500	250000	46.87	3.42
800 × 800	640000	206.13	2.27
1000 × 1000	1000000	308.16	1.91

Table 4.1: Quantitative results for computation of Zermelo problem solution with ordered upwind.

Figure 4.10 shows that even for a relatively accurate grid, the error remains large. Moreover, the execution time is very long to obtain such solutions with a lot of error as shown by Table 4.1. **The error is mainly located at the beginning of the trajectory** and this error has too much impact on the rest of the trajectory.

Great circle computation

Let us now consider a case where there is no wind but the coordinates used now are not the Cartesian ones anymore but great circle distances are taken into account.

Figure 4.11 shows results obtained from ordered upwind to compute the great circle trajectory between two points with different mesh sizes and Table 4.2 shows the different execution time to obtain them.

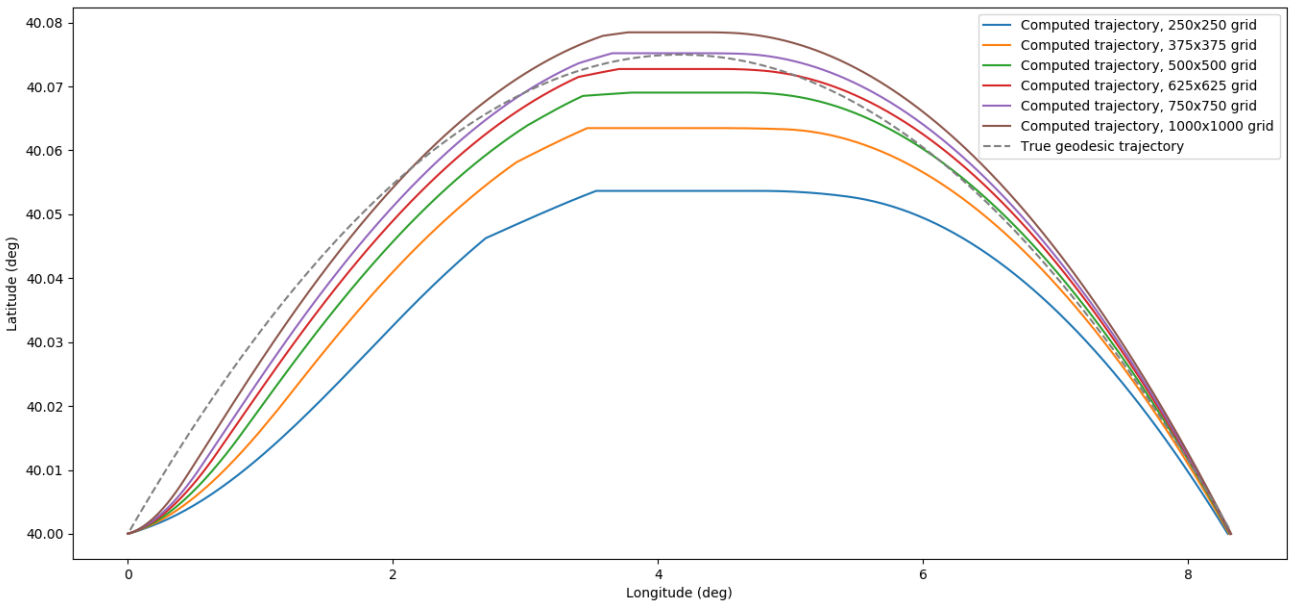


Figure 4.11: Results obtained from ordered upwind - Computation of the great circle trajectory between two points at different mesh sizes.

Mesh size	Number of points	Computation time (s)	Relative error on distance flown (%)
250 × 250	62500	5.787	6.87
375 × 375	140625	16.975	4.24
500 × 500	250000	34.751	2.65
625 × 625	390625	68.821	2.05
750 × 750	562500	128.922	1.97
1000 × 1000	1000000	294.311	1.23

Table 4.2: Results obtained from ordered upwind - Computation of the great circle trajectory between two points at different mesh sizes.

Again, Figure 4.11 shows that the start of the trajectory is very erroneous, which is symptomatic of fast marching and ordered upwind methods. It would therefore be better to favor this method for large distances, however this is prevented by a too high computation time when the number of points becomes large as shown in Table 4.2. Nevertheless, even if the execution time becomes large, the trajectory is almost the expected one when the number of points on the grid is large as shown in Figure 4.11.

4.2.5 Critical viewpoint

To conclude on this method, various positive and negative points can be raised. Firstly, the advantages include:

- ✓ The resulting trajectory is smooth;
- ✓ The resulting trajectory shows little error in the windless case and with a large mesh.

Drawbacks are:

- × The execution time is too long for good quality windless trajectories;
- × When wind is present, especially for the Zermelo problem, even when the number of points is large, the solutions are not of good quality with a very long computation time. This error is due to a large error at the beginning that propagates and to a poorly adapted mesh.

In conclusion, this method is not entirely satisfactory. It was decided to change the method to find a more efficient one rather than to continue working on this method. Especially for 3D, this method would not be usable.

4.3 Second method used: Fast Marching Tree

Because of the disadvantages of the last method presented in 4.2.5, it has been decided to change the method to find one which is faster. Sampling based methods allow to reduce the combinatory however they are not very used in the field of aircraft trajectory, even if some papers show that they have already been used but almost only in the case of UAVs, *i.e.* with dynamic constraints different from those of an aircraft and without taking into account great circle distances. Among these methods, the fastest, according to the state-of-art, is the method of Fast Marching Tree (FMT*) and so it has been chosen to avoid the problems mentioned.

4.3.1 Presentation of the algorithm

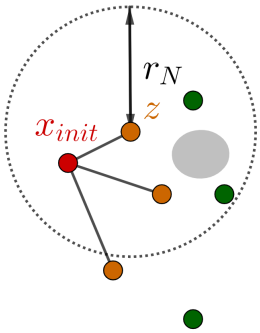
Fast Marching Tree (FMT*) has been introduced by Lucas Janson, Edward Schmerling, Ashley Clark and Marco Pavone in [17] in 2015. FMT* is proven to be asymptotically optimal and to converge faster than RRT*. Let us describe the algorithm.

Description of the algorithm The environment is at first (randomly) sampled with N points or nodes. Then, FMT* will explore each node to compute the optimal path from the origin to the goal. For this, at each iteration, nodes are split into 3 sets:

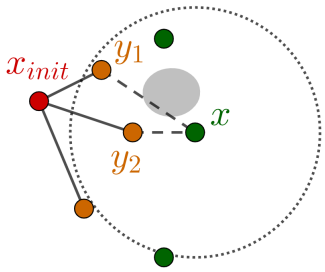
1. $V_{unvisited}$: nodes which have never been visited by the FMT* (in green in the following figures);
2. V_{open}: nodes which have already been visited but for which the cost value is not definitively computed (in orange in the following figures);
3. V_{closed}: nodes which have already been visited but for which the cost value is definitively computed (in red in the following figures).

Let us detail one iteration:

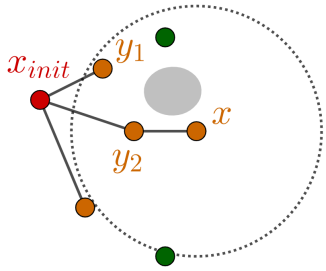
1. The lowest-cost open node z is selected and all its unvisited neighbors are considered. x_{init} is the starting point of the trajectory, and the root node of the tree.



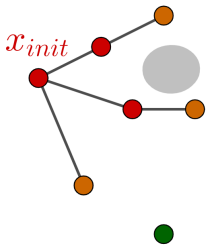
2. For each unvisited neighbor x of z, all x open neighbors are considered and an edge between x and one of these neighbors is added to make this connection locally-optimal (without considering any constraint).



3. If this connection does not violate constraints or is collision-free, it is added to the tree and x is removed from the set $V_{unvisited}$ and added to V_{open}.



4. Once all z unvisited neighbors have been visited, z is put in V_{closed} and the iteration is done.



All the nodes at a distance r_N of a node are considered as its neighbors. r_N is the neighborhood radius and is a function of the number N of samplings. The radius used in [17] is:

$$r_N = (1+\eta) * 2 * \left(\frac{1}{d}\right)^{1/d} * \left(\frac{\mu(\chi_{free})}{\zeta_d}\right)^{1/d} * \left(\frac{\log(N)}{N}\right)^{1/d} \tag{4.17}$$

where $\eta > 0$, d is the space dimension, χ_{free} the free space, $\mu(\chi_{free})$ its Lebesgue measure[2], ζ_d the volume of the d-dimensional unitary ball[3].

[2] $\mu([a1, b1] \times [a2, b2]) = (b1 - a1) * (b2 - a2)$ where $b1 > a1$ and $b2 > a2$.
$\mu([a1, b1] \times [a2, b2] \times [a3, b3]) = (b1 - a1) * (b2 - a2) * (b3 - a3)$ where $b1 > a1$, $b2 > a2$ and $b3 > a3$.

[3] In 2D space, ζ_2 is the surface of a disk of radius 1 ($\zeta_2 = \pi$).
In 3D space, ζ_3 is the volume of a ball of radius 1 ($\zeta_3 = \frac{4}{3}\pi$).

Detailed algorithm Algorithm 2 is the complete FMT* algorithm in pseudo-code. The set of neighbor of a node x is noted $Neighbors(x)$.

Algorithm 2: FMT* algorithm.

Input: x_{init}, sample set V (with $x_{init} \in V$), χ_{goal} ($\exists x \in \chi_{goal}, x \in V$)
Output: *succeed*

```
1  succeed ← false
2  V_open ← {x_init}
3  V_unvisited ← V\{x_init}
4  Tree initialized with its root x_init
5  while V_open ≠ ∅ ∧¬succeed do
6      z ← lowest-cost node in V_unvisited
7      V_successfully_connected ← ∅
8      for x ∈ V_unvisited ∩ Neighbors(z) do
9          e ← locally-optimal edge to x from a node in V_open ∩ Neighbors(x)
10         if e is collision-free then
11             Tree ← Tree ∪ e
12             V_successfully_connected ← V_successfully_connected ∪ {x}
13     V_unvisited ← V_unvisited\V_successfully_connected
14     V_open ← V_open ∪ V_successfully_connected
15     V_open ← V_open\{z}
16     V_closed ← V_closed ∪ {z}
17     if z ∈ χ_goal then
18         succeed ← true
19
```

To ensure the lowest possible complexity, the set V_{open} is implemented as a priority queue which allows an element to be extracted (line 6) in a constant time and an element to be added in $O(\log n)$ (with n the size of the queue).

The algorithm presented is general and has been adapted to the problem under study. The two following subsections will focus on this adaptation to the studied issue.

4.3.2 Dealing with great circle distances

As explained before, the computation of the exact great circle distance between two points is too time-consuming to be used for the trajectory computation. However, the same idea as before cannot be used here. Indeed, the former method works on a mesh and so the directions are defined and different multiplying factors could be computed according to these different directions. Here, the direction in which the algorithm will propagate is not known in advance and there is no direction on a grid to rely on. Another way of taking into account great circle distances has been found.

The new method chosen relies on the same hypothesis already mentioned to consider the straight line as the great circle trajectory between two near points and the **same factors as the ones computed for ordered upwind** are used (cf Section 4.2.3 and Appendix A). To approximate the great circle distance from a point $A(lat_A, lon_A)$ to a point $B(lat_B, lon_B)$, let:

$$m_{lon} = m_E(A) = m_W(A) \tag{4.18}$$

$$m_{lat} = \begin{cases} m_N(A) & \text{if } lat_A < lat_B \\ m_S(A) & \text{else.} \end{cases} \tag{4.19}$$

The distance between A and B can be approximated by:

$$d_{AB} \approx \sqrt{\left(\frac{lat_B - lat_A}{m_{lat}}\right)^2 + \left(\frac{lon_B - lon_A}{m_{lon}}\right)^2} \tag{4.20}$$

In practice, distances will mostly be squared to avoid time-consuming computation of a square root[4].

4.3.3 Taking into account aeronautical constraints

Because the trajectory computed is the result of a tree search, it is the concatenation of crossing segments and so it results in non-acceptable, or non-flyable trajectories.

At first, the crossing points between segments can be assimilated to waypoints and for the aircraft point of view once in the FMS, it will be computed as a flyable trajectory. To be more precise the computation of an acceptable turn has been done here, considering a minimum turn radius of 2 NM. Figure 4.12 shows an example.

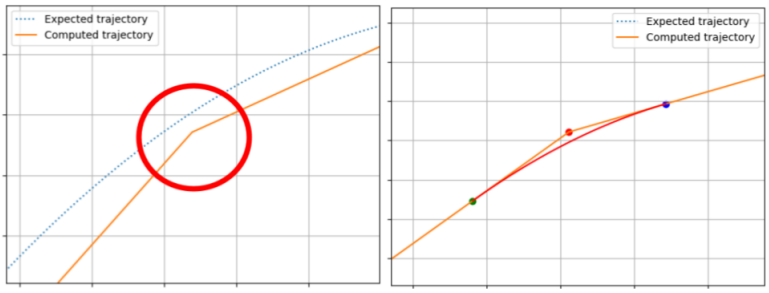


Figure 4.12: Turn radius constraint taken into account by FMT* post-processing.

This computation is done by computing the tangent circle of radius 2NM to the two segments. Nevertheless, this radius can be adapted according to the aircraft's performance. Appendix C details the computation to obtain this kind of trajectory smoothing.

4.3.4 Results

This subsection presents different results obtained thanks to the implemented method. First, it focuses on Zermelo's benchmark, then on great circle trajectory computation without wind and then gives an example of result on a realistic situation with wind and taking into account great circle distances.

[4]Minimizing a distance is the same as minimizing the square of a distance.

Zermelo's benchmark

As was done for the previously presented method (cf Section 4.2.4), a particular case (linear wind case) of the Zermelo problem has been studied to test the implemented method. The computations are done here in the case of Cartesian coordinates.

$$\forall (x, y), W(x, y) = \begin{pmatrix} \frac{V_a}{4} * y \\ 0 \end{pmatrix} \tag{4.21}$$

The value $\frac{V_a}{4}$ was chosen because it is a classical value of relatively strong wind at altitude.

The Figure 4.13 and Table 4.3 show results obtained from FMT* to compute the solution of Zermelo problem in this setup. To obtain such smooth trajectories, the chosen neighborhood radius is larger than the optimal one, it has been multiplied by 5 to obtain the following results.

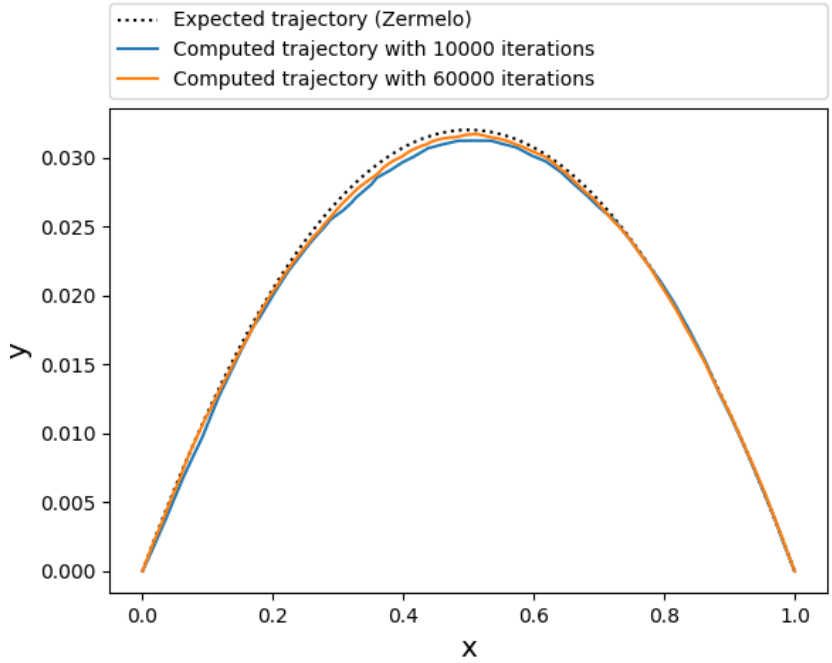


Figure 4.13: Results obtained from FMT* - Zermelo problem in the case of linear wind with different maximum of iterations.

Number of iterations	Relative error on distance flown (%)	Relative error on flight time (%)	Execution time (s)
10000	0.00995	0.00256	0.36
20000	0.01954	0.01542	1.64
30000	0.01764	0.01537	4.11
40000	0.01313	0.01534	8.66
50000	0.01462	0.01533	16.63
60000	0.01063	0.01532	25.35

Table 4.3: Relative errors and execution times for computing a solution to Zermelo problem with FMT*.

To obtain comparable results, the same experiments have been done on regular sampling of the environment. The Figure 4.14 shows some results obtained and the Table 4.4 shows the different relative errors and execution times to obtain these trajectories according to different number of iterations.

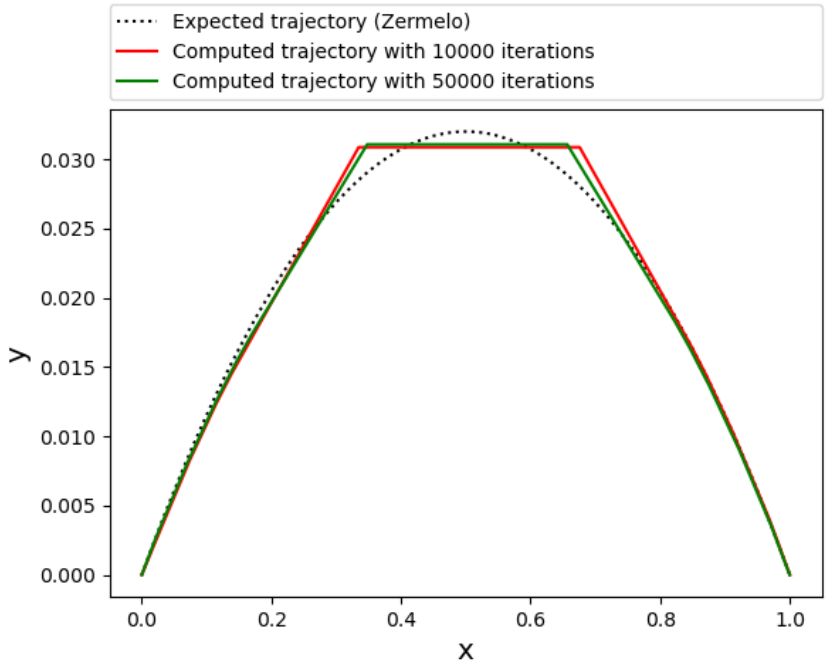


Figure 4.14: Results obtained from FMT* - Zermelo problem in the case of linear wind with different maximum of iterations and regular sampling.

Number of iterations	**Relative error on distance flown (%)**	**Relative error on flight time (%)**	**Execution time (s)**
10000	0.02194	1.09371	0.61
20000	0.01952	1.09311	1.89
30000	0.01856	1.09116	4.70
40000	0.01236	1.07859	8.15
50000	0.01462	1.8321	13.22
60000	0.01001	1.07381	19.75

Table 4.4: Relative errors and execution times for computing a solution to Zermelo problem with FMT* with regular sampling.

The presented results show that good results, **better than the one obtained with ordered upwind**, can be obtained in a rather short time, **much shorter than what was obtained with the ordered upwind method**. The comparison between random sampling and regular sampling shows that with random sampling smooth trajectories close to what is expected can be obtained, while on the contrary regular sampling gives angular trajectories. However, random sampling does not always improve the result by increasing the number of iterations.

Great circle computation

Moreover, the method has been tested in a windless case taking into account great circle distances as it was done in section 4.2.4 for the ordered upwind method. The Figure 4.15 and Table 4.5 shows results obtained from FMT* to compute the great circle trajectory between two points with different maximum of iterations.

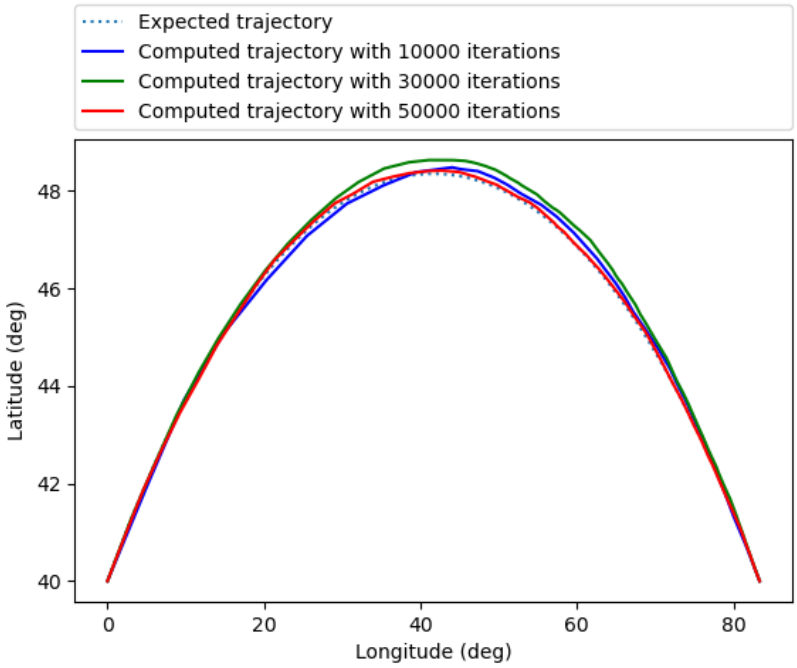


Figure 4.15: Results obtained from FMT* - Computation of the great circle trajectory between two points with different maximum of iterations.

Number of iterations	Relative error (%)	Execution time (s)
10000	0.43602	0.59
20000	0.42486	1.51
30000	0.43225	5.03
40000	0.43242	9.57
50000	0.42232	17.37
60000	0.42113	26.19

Table 4.5: Relative errors and execution times for great circle trajectory computation with FMT* and regular sampling.

To obtain comparable results, the same experiments have been done on regular sampling of the environment. The Figures 4.16 and the Table 4.6 shows the different relative errors and execution times to obtain these trajectories.

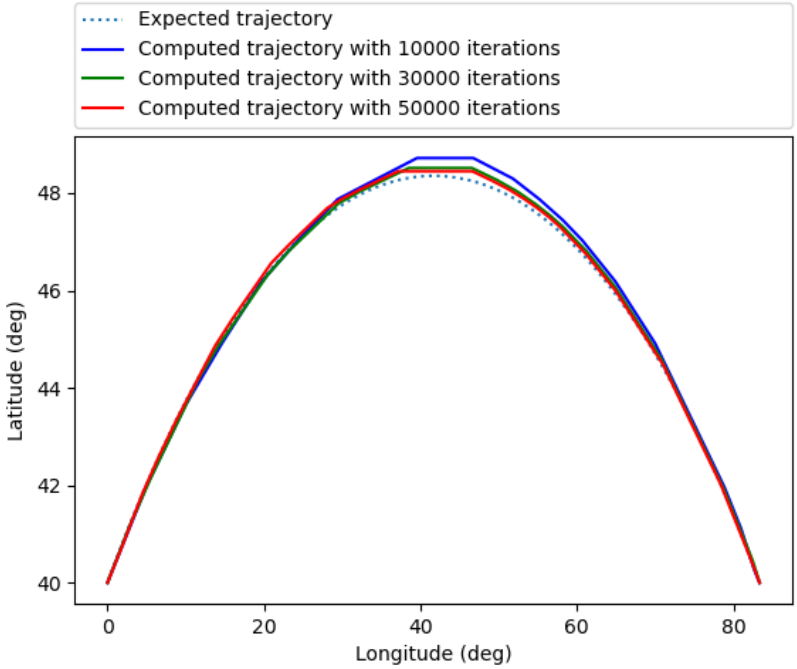


Figure 4.16: Results obtained from FMT* - Computation of the great circle trajectory between two points with different maximum of iterations with regular sampling.

Number of iterations	Relative error (%)	Execution time (s)
10000	0.44835	0.77
20000	0.43743	1.50
30000	0.43264	4.05
40000	0.43060	7.79
50000	0.42762	14.07
60000	0.42466	19.91

Table 4.6: Relative errors and execution times for great circle trajectory computation with FMT* and regular sampling.

Once again, the results presented are quite satisfactory, especially when the sampling is random, with low computation time.

Case study

Finally, the method was tested on real wind data taking into account great circle distances. These tests allow to evaluate the benefits obtained thanks to the implemented tool.

Wind data have been extracted from the website site Windy[5] on a square grid of size 0.1 degree from Paris to Copenhagen as shown by Figures 4.17 and 4.18.

[5]https://windy.com

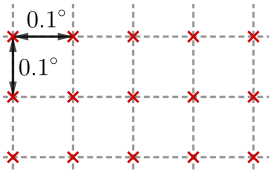


Figure 4.17: Grid used for wind data extraction from Windy for 2D study case (Paris-Copenhagen).

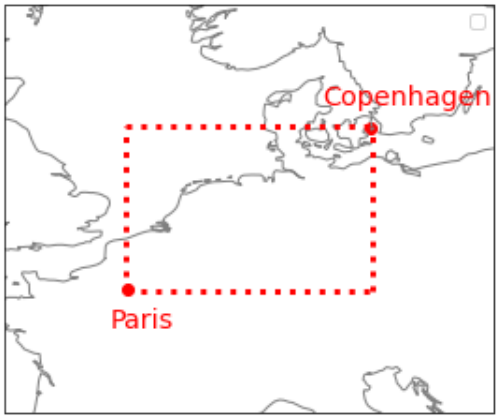


Figure 4.18: Area over which the wind data are extracted from Windy for 2D study case (Paris-Copenhagen).

Figure 4.19 shows the wind obtained.

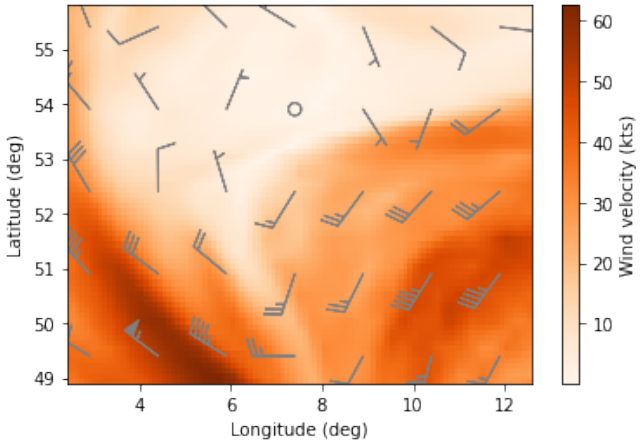


Figure 4.19: Wind extracted from Windy, 5th August 2021, FL 300.

Between the data grid points, a so-called Shepard interpolation [26] was used. More precisely, each point P is located in a square $P_1P_2P_3P_4$ of the data grid and the wind at this point is calculated from the points of this square using the distance to each of these points (respectively d_1, d_2, d_3 and d_4) using the following formula:

$$W(P) = \frac{\sum_{i=1}^{4} W(P_i) * d_i^{-p}}{\sum_{i=1}^{4} d_i^{-p}} \tag{4.22}$$

where $W(P)$ is the wind vector at point P and $p > 1$. In this study, $p = 2$ has been chosen. Figure 4.20 summarizes the different notations.

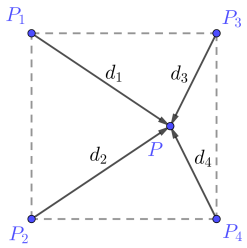


Figure 4.20: Notations for Shepard interpolation.

To conduct this study, air speed data from an A320 aircraft is used: $V_a = 440kts$.

From Paris to Copenhagen, the trajectory calculated by the FMT* is presented in Figure 4.21. Table 4.7 shows the different quantitative results obtained.

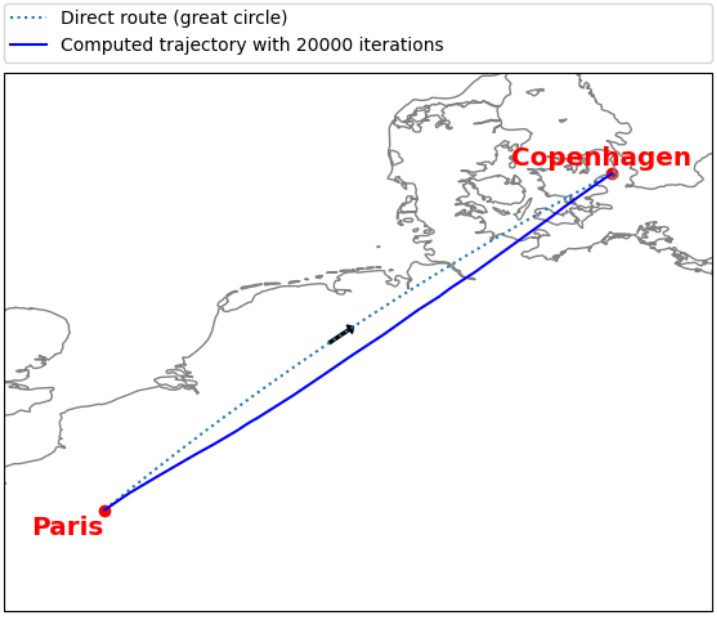


Figure 4.21: Results for the study case from Paris to Copenhagen. The dotted line represents the direct route, *i.e.* the optimal route without wind. The solid line shows the calculated route with the wind taken into account.

	Direct route (great circle)	**Computed trajectory**	**Gain/ Loss**
Distance flown	544.88 NM	550.82 NM	1.09 %
Flight time	1h12min01s	1h11min40s	-0.46 %

Table 4.7: Quantitative results for 2D case study - Paris to Copenhagen.

From Copenhagen to Paris (opposite direction), the trajectory calculated by the FMT* is presented in Figure 4.22. Table 4.8 shows the different quantitative results obtained.

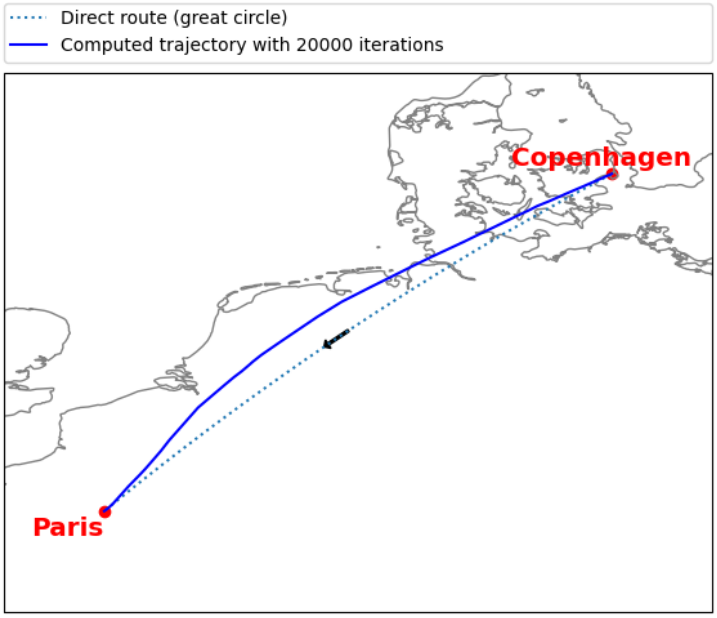


Figure 4.22: Results for the study case from Copenhagen to Paris.

	Direct route (great circle)	Computed trajectory	Gain/ Loss
Distance flown	544.88 NM	550.17 NM	0.97 %
Flight time	1h14min22s	1h13min55s	-0.61 %

Table 4.8: Quantitative results for 2D case study - Copenhagen to Paris.

In both case, the distance flown is higher but the flight time is lower for the trajectory computed. Moreover, the trajectory obtained is smooth and seems to be a flyable trajectory. Even if the gain seems to be small, however if this process applied to the whole European traffic, the gain will be huge. Finally, the results obtained are comparable to those obtained by Brunilde Girardet in her PhD thesis [23], which presents an average gain between 14 and 41s for the global traffic studied. Moreover the gain would be even larger for long range flights.

4.3.5 Critical viewpoint

The results presented are completely satisfactory. Indeed, the computation time is low and the quality of the results obtained is good. Moreover, the number of iterations does not need to be high to give very good quality results. The case study was done on 20000 iterations and gives results comparable to those already presented in the literature as for example in [23].

However, the algorithm needs to be improved to get closer to the operational level. It would be necessary to take into account the uncertainties on the wind predictions and also to take into account the operational constraints related to the airspace such as the presence of other traffic and the borders.

In the context of the future envisaged in the project, the algorithm can be adapted to take into account obstacles as this is one of the original goals of the algorithm. Moreover, the dimension could be increased to take into account a dynamic wind or a dynamic environment since the computation times are low in 2D and suggest that in higher dimension they will also be acceptable.

The algorithm developed in 2D thus shows good performances. However, the consideration of a 3D environment is necessary to completely answer the problem addressed by this project. The following chapter deals with this sub-problem.

Chapter 5

3D Problem resolution

This chapter presents the 3D problem resolution. First, the problem is presented and then the method developed to solve it is presented. Then, some results obtained from this method are shown.

5.1 3D problem

This section explains the problem addressed and then explains why the 2D method cannot be reused.

5.1.1 Problem statement

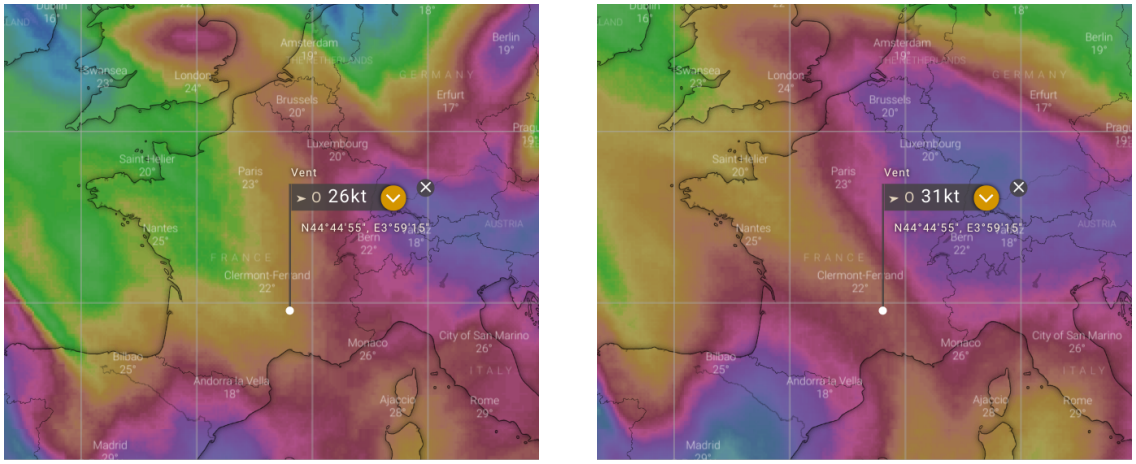
The objective is to minimize flight time of an aircraft from a root point to a goal point taking into account wind and great circle distances. This minimization is done in 3D, so any **change in altitude can be allowed under certain conditions**.

Constraints in the horizontal plane are taken into account to obtain a flyable trajectory. Typically, the turns radii have to be acceptable for a standard airliner. A typical radius of 2 NM is used.

Constraints in the vertical plane are taken into account to obtain a flyable trajectory:

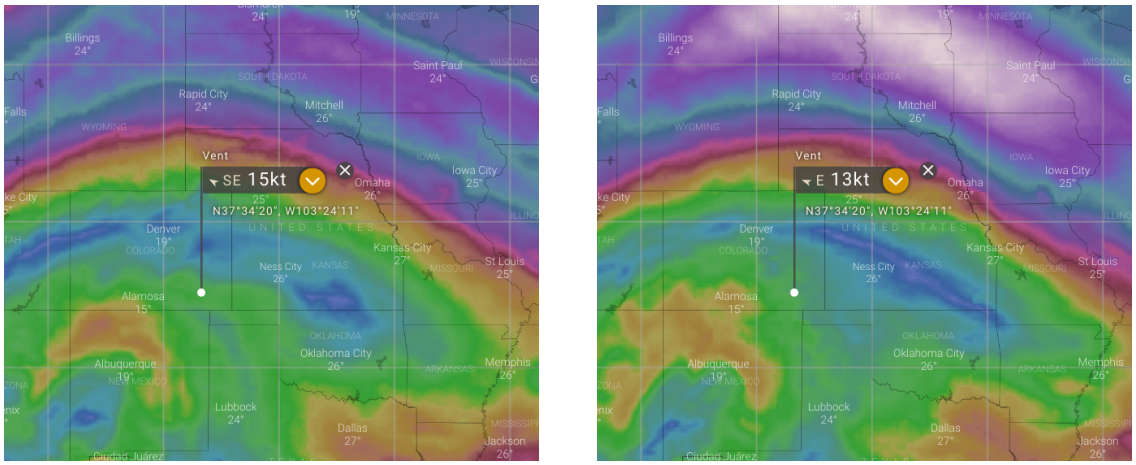
- The aircraft must fly in a flight level and not between two flight levels;
- The climb/descent slope must be acceptable for a standard airliner;
- The aircraft is always climbing and never descending before the final descent to respect operational preferences. The higher the aircraft is, the less it consumes, until a maximum Flight Level (FL) from which the aircraft can no longer fly because the air density becomes too low;
- In order to climb, an aircraft must have a weight below a certain limit which depends on the flight level. Since fuel consumption is considered proportional to flight time and the weight decreases when the fuel on board decreases, this upper limit in weight can be transformed into a lower limit in flight time. The aircraft will therefore be able to climb from a certain flight time.

In practice, the wind varies little in direction and velocity between two flight levels as shown in Figures 5.1 and 5.2 from the Windy[1] website. It is therefore necessary to find a balance between **climbing only when it is beneficial**, *i.e.* when the wind obtained at the upper level is more favorable, **or staying too long at the lower flight level** which could be much more penalizing than taking an unfavorable wind since this wind will not be completely different from the current wind.



(a) FL 340 (b) FL 390

Figure 5.1: Wind comparison at FL 340 and 390 in France.



(a) FL 340 (b) FL 390

Figure 5.2: Wind comparison at FL 340 and 390 in the USA.

5.1.2 Why the 2D method cannot be used is 3D?

The method used in 2D, FMT*, can be easily adapted in 3D as it has been proved to be efficient in such spaces [17]. However, it does not fit to the needs linked to the problem addressed in 3D for two main reasons.

First, the number of building constraints in 3D is too high for FMT*. As FMT* does the sampling before the propagation, it cannot easily take into account these constraints.

[1]https://www.windy.com/

Besides, the goal is to do as little as possible flight level change. Indeed, the aircraft is in its cruise phase and so it has to change its flight level only if it is really beneficial. FMT* is not able to minimize the number of flight level changes, nor is it able to find a balance between not climbing if the wind at the upper level is less favorable without staying too long on a flight level if it is possible to climb.

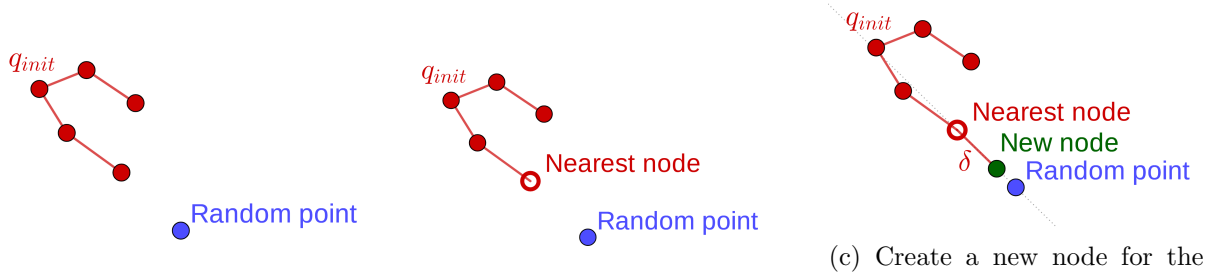
5.2 Solution implemented

This section presents the solution implemented. Before this, it presents more in details the RRT and T-RRT algorithms which have been used to develop this solution.

5.2.1 RRT

This subsection introduces RRT and its asymptotically optimal variant, RRT*.

RRT (for Rapidly-Exploring Random Trees) has been developed by Lavalle in [13]. The general principle of RRT construction is to randomly generate a point in the given configuration space, find the nearest neighbor of this random point and from this nearest neighbor create a node in the direction of the random point at a fixed distance (parameter of the algorithm) and then add this new node to the tree and create an edge between the nearest and the new nodes if it is collision-free/ not violating a constraint. This results in a tree on which a search will find the path of least cost. The tree is initialized with the start point (q_{init}) and complete through a process shown in Figure 5.3.



(a) Peek a random point in the free space, in red the tree already built until the current iteration.

(b) Get the node belonging to the tree that is nearest to the random point.

(c) Create a new node for the tree in the direction of the random point from its nearest node of the tree, over a distance δ, if the created edge is in the free space.

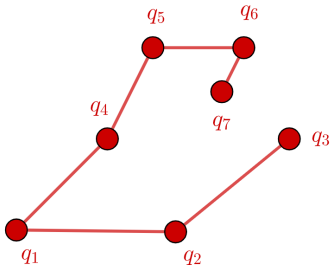
Figure 5.3: An iteration of RRT building algorithm.

Examples of results can be found in the state of the art (cf Figure 3.3).

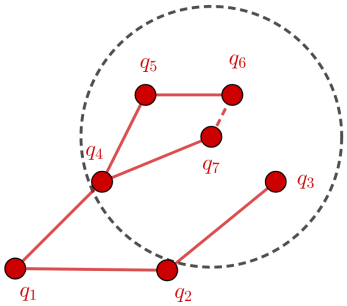
RRT*

RRT is not asymptotically optimal however one of its variant developed by Karaman and Frazzoli in [27], RRT*, is proven to be asymptotically optimal. The principle is RRT* is quite the same as the RRT one, the difference is that when a node is added to the tree, its neighbors will be rewired if necessary. This process consists in looking at each node in the neighborhood

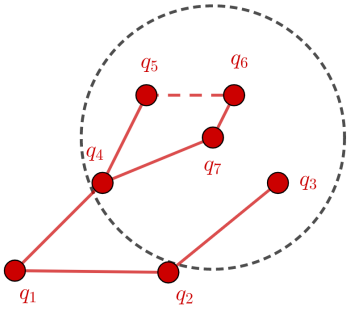
to see if replacing its parent with the node added would be beneficial or not. Figure 5.4 shows this process.



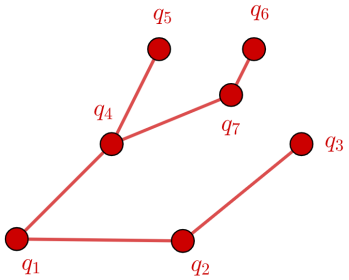
(a) The node q_7 is the latest node created and has been created from q_6.



(b) The best parent for q_7 in the neighborhood is not q_6 but q_4, so the edge from q_6 to q_7 is deleted and an edge from q_4 to q_7 is added.



(c) The node q_6, which is inside the neighborhood, would have a lower cost if its parent became q_7. So the edge from q_5 to q_6 is removed and an edge is created between q_7 and q_6.



(d) After the rewiring process, this tree is obtained.

Figure 5.4: Rewiring process.

The neighbors of node i at iteration k are the nodes at a distance less than r_k from this node with:

$$r_k = (1+\eta) * 2 * \left(1+\frac{1}{d}\right)^{1/d} * \left(\frac{\mu(\chi_{free})}{\zeta_d}\right)^{1/d} * \left(\frac{\log(k)}{k}\right)^{1/d} \tag{5.1}$$

where $\eta > 0$, d is the space dimension, χ_{free} the free space, $\mu(\chi_{free})$ its Lebesgue measure[2], ζ_d the volume of the d-dimensional unitary ball[3].

[2] $\mu([a1,b1]\times[a2,b2]) = (b1-a1)*(b2-a2)$ where $b1 > a1$ and $b2 > a2$.
$\mu([a1,b1]\times[a2,b2]\times[a3,b3]) = (b1-a1)*(b2-a2)*(b3-a3)$ where $b1 > a1$, $b2 > a2$ and $b3 > a3$.

[3] In 2D space, ζ_2 is the surface of a disk of radius 1 ($\zeta_2 = \pi$).
In 3D space, ζ_3 is the volume of a ball of radius 1 ($\zeta_3 = \frac{4}{3}\pi$).

Algorithm 3 details RRT* algorithm. The lines in blue are the lines common to RRT and RRT*.

Algorithm 3: RRT* algorithm.

Input: CS the configuration space, δ the distance increment, q_{init} the root, N the desired number of nodes

Output: the tree $\mathcal{T}$

```
1  T ← InitTree(q_init)
2  for k = 1 to N do
3      q_rand ← RandomPoint(CS) ;                          // Figure 5.3a
4      q_near ← BestNeighbor(q_rand,T) ;                   // Figure 5.3b
5      q_new ← Extend(T,q_rand,q_near);                    // Figure 5.3c
6      Q_near ← Near(T,q_new,r_k)
7      AddNewNode(T,q_new)
8      q_min ← q_near, c_min ← cost(q_near) + c(q_near, q_new)
9      for q ∈ Q_near do ;          // Finding the best parent for the new node
10
11         if cost(q) + c(q, q_new) < c_min then
12             q_min ← q, c_min ← cost(q) + c(q, q_new)
13         end
14     end
15     AddNewEdge(T,q_min,q_new)
16     for q ∈ Q_near do ;                          // Rewiring neighbor nodes
17
18         if cost(q_new) + c(q_new, q) < cost(q) then
19             q_parent ← Parent(q)
20             RemoveEdge(T,q_parent,q)
21             AddNewEdge(T,q_new,q)
22         end
23     end
24 end
```

The function $Near$ is the function used to search the neighbors of the node created according to the radius which can be computed thanks to equation 5.1.

An example of results comparison between RRT and RRT* can be found in the state of the art (cf Figure 3.4).

5.2.2 Details about T-RRT

This subsection describes more in details the T-RRT algorithm mentioned in the state of the art.

T-RRT (Transition Based RRT) has been published in [18] for path planning in continuous-cost spaces. It can be used for several applications in complex cost spaces such as terrain cost map motions or low cost motion for articulated robots. For this, it combines the **ability of RRT to explore unexplored regions to efficiency of stochastic optimization methods**. It includes the auto-tuning of a parameter regulating the exploration.

The main difference with RRT is that a transition between two nodes is accepted with a certain probability:

$$P(\text{transition accepted}) = \begin{cases} 1 & \text{if } \Delta_{ij}^* \leq 0 \\ \exp(\frac{-\Delta_{ij}^*}{K*T}) & \text{else.} \end{cases} \tag{5.2}$$

Originally, $\Delta_{ij}^* = \frac{c_j - c_i}{d_{ij}}$ in the case of a minimization of criteria c_k with d_{ij} the Euclidean distance between node i and j.

T is the temperature which evolves during the exploration of the space and $K = \frac{c_{init}+c_{goal}}{2}$ is the average between the cost of the starting point and the cost of the ending point. The use of stochastic decision is common in optimization methods (it is for example used by the method called simulated annealing). The exploration of the set of states then corresponds to the moment when the temperature is high (high probability of accepting a transition that gives a worse solution) and when the temperature decreases, the search is intensified towards an area of the set of states that seems to be the best. The original version of T-RRT developed in [18] includes an auto-tuning of this temperature parameter in function of a maximum admissible fails of the transition test. If this maximum number is reached, the temperature is increased thanks to a multiplying factor. Thanks to this process, the algorithm can be initialized with a low temperature and then it will be increased if necessary. **There is no tuning to do on the initial temperature of the algorithm.**

The Appendix D shows in details the transition test used in T-RRT and shows how auto-tuning of the temperature parameter is done.

An important point to keep in mind is that T-RRT is based on RRT and not on RRT*. That is, there is no questioning of the current tree. What is built will not be modified anymore. So there is **no proof of optimality** as for RRT.

5.2.3 New algorithm adapted to the case under study

The algorithm developed to solve the 3D problem is based on the same principle as T-RRT, *i.e.* using the efficiency of RRT methods while combining the power that stochastic decisions can bring to change or not the flight level at each node. The decision is taken according to two criteria: the wind encountered and the ability of the aircraft to climb. Indeed, this climb will only be possible if the aircraft is light enough, so **there is a lower limit on the flight time to climb or not**. Staying too long on a flight level, even if it is to have a more favorable wind, penalizes the consumption too much. **After a certain time, it is necessary to force the climb**, especially since the wind does not vary much according to the altitude as explained in section 5.1.1, so the wind penalty will not be very important if it exists.

The algorithm implemented to solve the 3D problem uses a temperature for each node T_{climb} which will evolve according to the flight time of each node as shown by Equation 5.3.

$$T_{climb}(j) = \beta * (ft_j - FT_{z_j}) \tag{5.3}$$

with $\beta > 0$ is a parameter to tune, z_j the altitude of node j, ft_j the flight time at node j and FT_{z_j} the flight time required for the aircraft to be able, from altitude z_j, to climb to the next usable flight level.

Moreover, even if classical T-RRT does not use the rewiring principle used by RRT*, the solution implemented uses this in order to improve the quality of the solution. This process is

used on each cruise stage of the trajectory (each piece of trajectory with the same flight level). Because the temperature of the node depends on the flight time computed at this node, **the temperature is recomputed when the node is rewired**.

The Algorithm 4 shows in details the developed algorithm. Lines in red are the lines which are specific to this algorithm, the lines which make the algorithm different from a classical RRT*. Some lines are not specific to this algorithm, but their processes are quite different, these lines are detailed thereafter.

Algorithm 4: 3D algorithm.

Input: CS the configuration space, δ the distance increment, q_{init} the root, N the desired number of nodes

Output: the tree $\mathcal{T}$

1 $\mathcal{T} \leftarrow$InitTree(q_{init})
2 **for** $k = 1$ *to* N **do**
3 $q_{rand} \leftarrow$ RandomPoint(CS)
4 $q_{near} \leftarrow$ BestNeighbor(q_{rand},$\mathcal{T}$)
5 $q_{new} \leftarrow$ Extend($\mathcal{T}$,q_{rand},q_{near})
6 $Q_{near} \leftarrow$ Near($\mathcal{T}$,q_{new},r_k)
7 AddNewNode($\mathcal{T}$,q_{new})
8 $q_{min} \leftarrow q_{near}$, $c_{min} \leftarrow cost(q_{near}) + c(q_{near}, q_{new})$
9 **for** $q \in Q_{near}$ **do** ; `// Finding the best parent for the new node`
10
11 **if** $cost(q) + c(q, q_{new}) < c_{min}$ **then**
12 $q_{min} \leftarrow q$, $c_{min} \leftarrow cost(q) + c(q, q_{new})$
13 **end**
14 **end**
15 AddNewEdge($\mathcal{T}$,q_{min},q_{new})
16 **for** $q \in Q_{near}$ **do** ; `// Rewiring neighbor nodes`
17
18 **if** $cost(q_{new}) + c(q_{new}, q) < cost(q) \wedge q$ *is not at a flight level change* **then**
19 $q_{parent} \leftarrow Parent(q)$
20 *RemoveEdge*($\mathcal{T}$,q_{parent},q)
21 *AddNewEdge*($\mathcal{T}$,q_{new},q)
22 **end**
23 **end**
24 **if** *TranstionTestClimb*(q_{near}) **then**
25 $q_{climb} \leftarrow$ createNodeClimb(q_{near})
26 AddNewNode($\mathcal{T}$,q_{climb})
27 AddNewEdge($\mathcal{T}$,q_{near},q_{climb})
28 **end**
29 **end**

At line 4, the best neighbor is chosen among the nodes at the same level as the random point. If there is no such node, the best neighbor is chosen among all the nodes.

At line 5, **the node is created at the same flight level as the one from which it is created** with a temperature computed from Equation 5.3.

At line 24, the transitions tests are made according to the same principle as in T-RRT with the respective temperatures of q_{near}. When the flight time at node q_{near} with lower than the

one necessary to climb, the transition is not accepted. The temperatures of nodes created are computed according to Equation 5.3. The probability to accept the transition is computed from Equation 5.2 with $\Delta^*_{ij} = -(GS_j - GS_i)$ where GS_i is the ground speed in node i (cf Figure 4.3 for ground speed definition). The cost of a node can be considered here as $-GS_i$ because the goal is to **maximize** the ground speed. The distance d_{ij} is not used in Δ^*_{ij} because the distance between the nodes i and j is constant as it is explained in subsection 5.2.5). And at last, $K = \frac{GS_{init}+GS_{goal}}{2}$ where GS_{init} is the ground speed at the initial point and GS_{goal} is the ground speed at the goal point. To summarize, the probability in each node i to climb and thus create a node j at the next flight level follows the equation 5.4.

$$P_{climb}(i) = \begin{cases} 0 & \text{if } ft_i < FT_{z_i} \\ 1 & \text{if } ft_i \geq FT_{z_i} \text{ and } GS_i \leq GS_j \\ \exp(\frac{GS_j - GS_i}{K * T_{climb}(i)}) & \text{otherwise} \end{cases} \tag{5.4}$$

with z_i the altitude of node i, ft_i the flight time at node i, FT_{z_i} the flight time required for the aircraft to be able, from altitude z_i, to climb to the next usable flight level and GS_i the ground speed at node i (taking into account wind).

Unlike in the T-RRT transition test, there is no auto-tuning of the parameters because the **temperature changes are done at the creation of new nodes**.

5.2.4 Taking into account great circle distance

To take into account great circle distance, the same process as the one used for the 2D case with FMT* has been used (cf Equation 4.20). The only modification is the consideration of the altitude variation:

$$d_{AB} = \sqrt{d^2_{2D_{AB}} + (z_B - z_A)^2} \tag{5.5}$$

where $d_{2D_{AB}}$ is the 2D-distance, and z_A and z_B the altitudes of nodes A and B. Typically, $z_B - z_A = 1000$ ft.

5.2.5 Taking into account aeronautical constraints

Some aeronautical constraints have to be taken into account, especially in the vertical plane.

Climb slope The first constraint to be taken into account is the climb slope. Considering it is fixed, for instance $\gamma = 3°$ then it can be easily integrated in the algorithm at the node construction in the *Extend* method as shown by Figure 5.5.

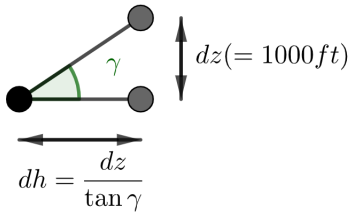


Figure 5.5: Horizontal distance in function of the climb slope.

The slope is fixed at the construction of the tree to ensure this constraint is not violated. In the function *Near* (line 6), only neighbors which have the same altitude as q_{new} are considered in order to avoid rewiring a node from another flight level and so change the slope. Likewise, nodes at a flight level change (nodes which are at the end of an edge between two flight levels) are not rewired in order to avoid changing the slope. As the graph is rewired only in 2D planes, **the radius used will be computed thanks to the equation 5.1 with $d = 2$.**

To go further than the first approximation of a fixed slope, a flight level dependent slope can be used: the higher an aircraft is, the slower it will climb. These values will depend on the type of aircraft used. It will also depend on the weather (temperature or wind for instance). However, the principle remains the same, the slope is fixed at the construction of the node.

Turns during a climb Another constraint is fixed thanks to the tree construction. When climbing, the aircraft keeps the same heading in order to avoid unfeasible turn angles during a climb or a descent. Nodes at a flight level change (nodes which are at the beginning of an edge between two flight levels) are not rewired in order to avoid changing the heading. This slightly reduces the optimality of the computed solution but makes it operationally acceptable.

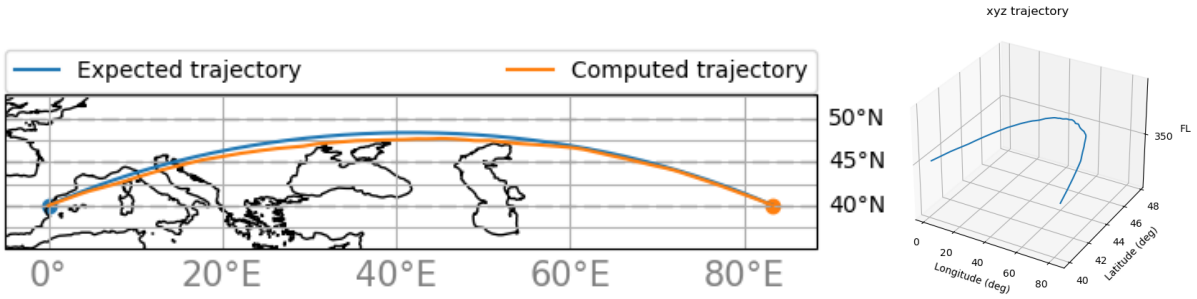
5.3 Results

5.3.1 Results on great circle computation

The algorithm was tested on the case of computing a windless great circle between two points by allowing more or less flight level changes to evaluate the performance of the developed algorithm.

The tests were done with 45000 iterations and the parameters were therefore tuned for this number of iterations. The parameters being very dependent on this number of iterations, the change of the number of iterations must be associated with a new tuning of these parameters.

0 climb authorized The case of no climb authorized allows to evaluate the performances of the algorithm because in this situation there will be no error due to the climb and the heading constraint when climbing. In the case of no climb, the algorithm is assimilated to a classical RRT* in 2D adapted to the case of great circle distances. Figure 5.6 shows the result of the algorithm. A relative error of 0.55% **on the distance flown** (and on the flight time since there is no wind) is obtained with a computation time of 61.03 s.



(a) Results on xy view.

(b) Results on xyz view.

Figure 5.6: Results obtained with the 3D method on great circle computation without wind, with 45000 iterations and no climb authorized.

1 climb authorized Let us now consider the possibility of climbing once, *i.e.* for the aircraft to climb from FL 350 to FL 360 after a certain time fixed in the algorithm. The slope chosen here is 3 degrees but it can be modified as explained previously (cf Section 5.2.5) according to the type of aircraft. Figures 5.7 and 5.8 show the result of the algorithm. A **relative error of** 0.72% **on the distance flown** (and on the flight time since there is no wind) is obtained with a computation time of 58.49s. The error has increased comparing to the case without climb. Indeed, when a climb is executed, the heading is kept and so it created an error which can be propagated.

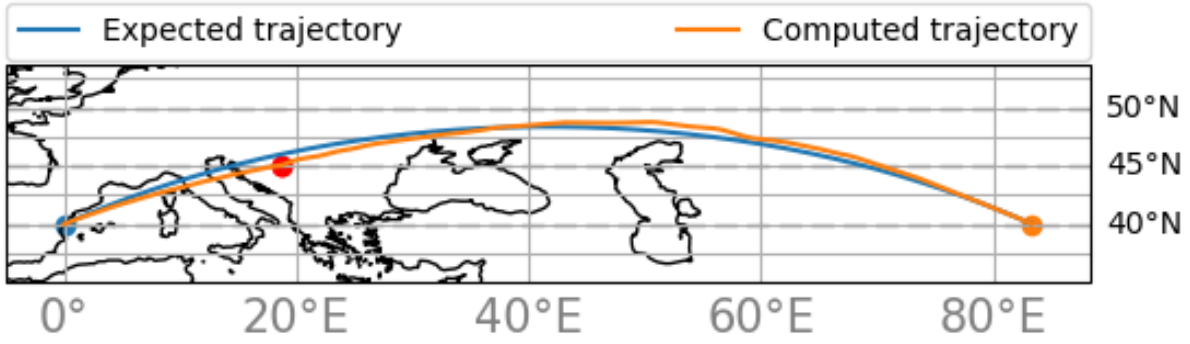
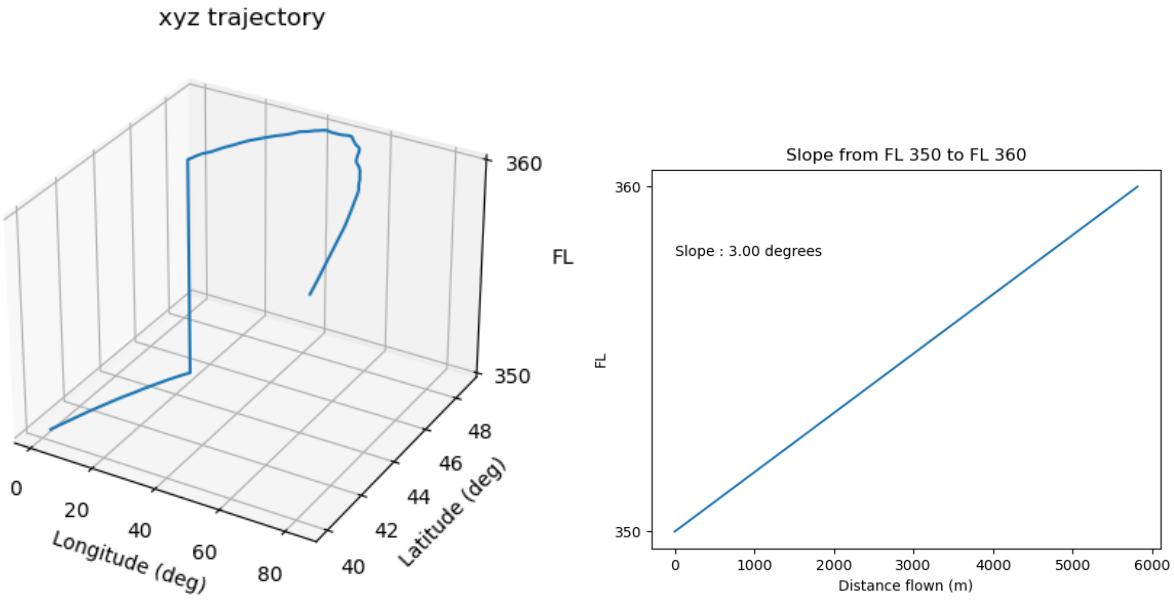


Figure 5.7: Results obtained with the 3D method on great circle computation without wind, with 45000 iterations and 1 climb authorized, on xy view. The point in red is the climb point.



(a) Trajectory on xyz view.

(b) Slope used for the climb.

Figure 5.8: Results obtained with the 3D method on great circle computation without wind, with 45000 iterations and 1 climb authorized, vertical view.

2 climbs authorized Let us now consider the possibility of climbing twice, *i.e.* for the aircraft to climb from FL 350 to FL 360 after a time fixed in the algorithm and then from FL 360 to FL 370 after another time fixed in the algorithm. The slope chosen here is 3 degrees for the first climb and then 2.75 degrees for the second climb but it can be modified as explained previously

(cf Section 5.2.5) according to the type of aircraft. Figures 5.9 and 5.10 show the result of the algorithm. **A relative error of** 0.98% **on the distance flown** (and on the flight time since there is no wind) is obtained with a computation time of 59.36s. Once again, the error has increased comparing to the case with zero or one climb because of the conservation of the heading while climbing.

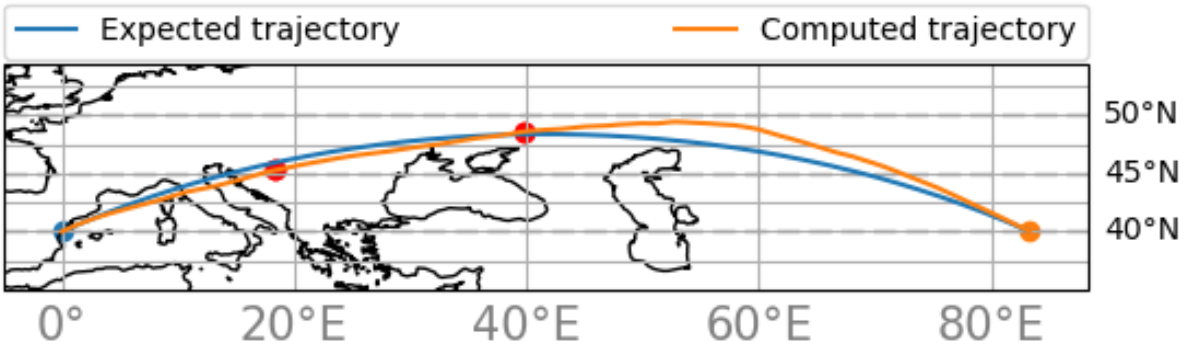
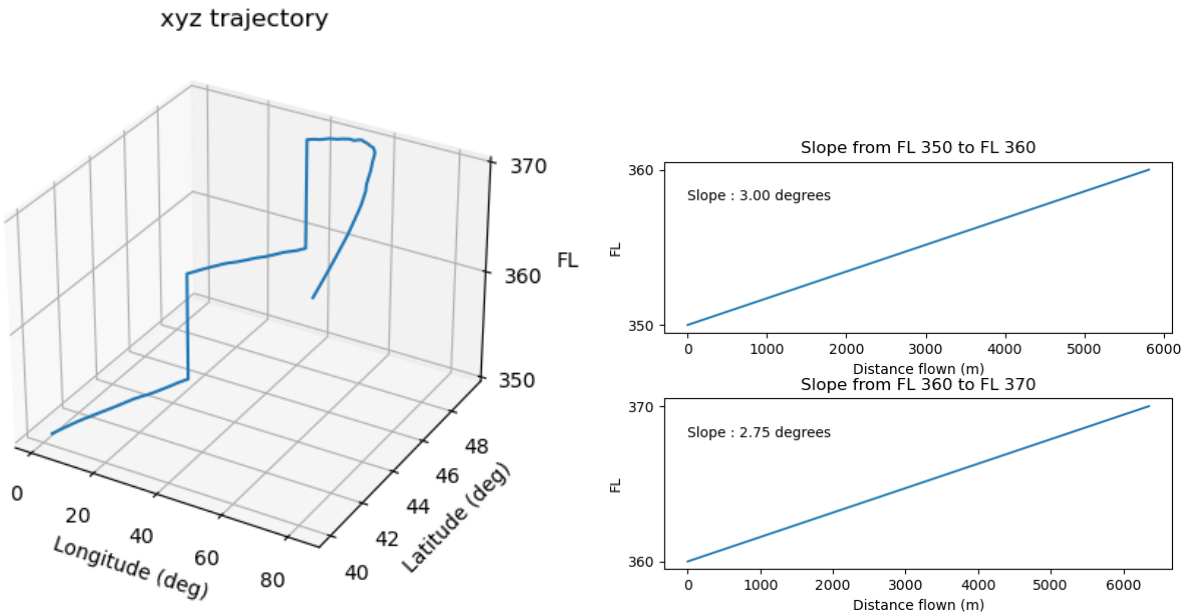


Figure 5.9: Results obtained with the 3D method on great circle computation without wind, with 45000 iterations and 2 climbs authorized, on xy view. The point in red is the climb point.



(a) Trajectory on xyz view.

(b) Slope used for the climb.

Figure 5.10: Results obtained with the 3D method on great circle computation without wind, with 45000 iterations and 2 climb authorized, vertical view.

5.3.2 Zermelo problem

To check that the algorithm works to compute the optimal wind path, it was tested again on the Zermelo problem, as were the 2D algorithms. The same particular case of linear wind was retained and once again the calculation is done in Cartesian coordinates.

$$\forall(x,y), W(x,y) = \begin{pmatrix} \frac{V_a}{4} * y \\ 0 \end{pmatrix} \tag{5.6}$$

It has been tested on 45000 iterations with no climb authorized. Figure 5.11 shows the result of the algorithm when no climb is authorized. The **relative error on the flight time is** 1.05% **and** 0.00359% **on the distance flown**, which are satisfactory results.

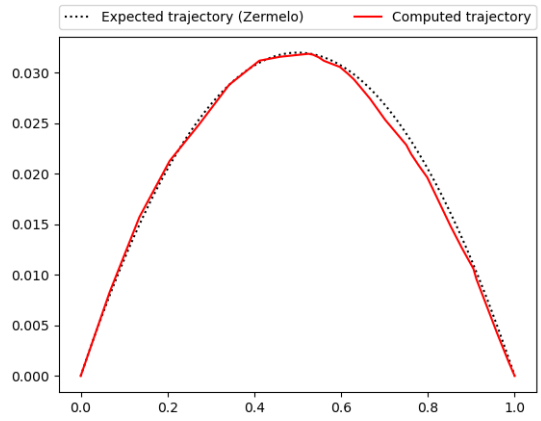
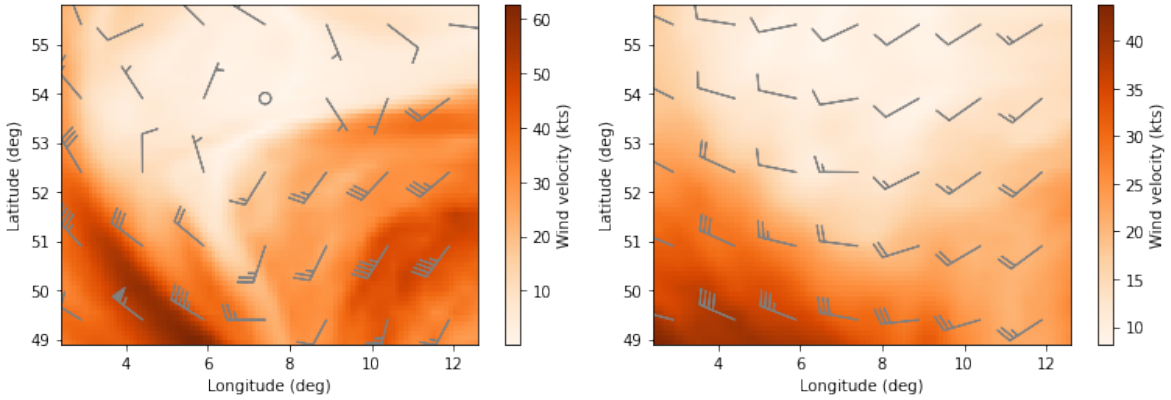


Figure 5.11: Zermelo problem solution computation in 3D - no climb authorized.

5.3.3 Case study

The study done in 2D between Copenhagen and Paris was repeated in the 3D case. This time the wind data were extracted at FL 300 and FL 390 and give what is shown in Figure 5.12.



(a) FL 300

(b) FL 390

Figure 5.12: Wind extracted for 3D case study.

The interpolation principle used is the same but this time in 3D and therefore with 8 points as shown in Figure 5.13.

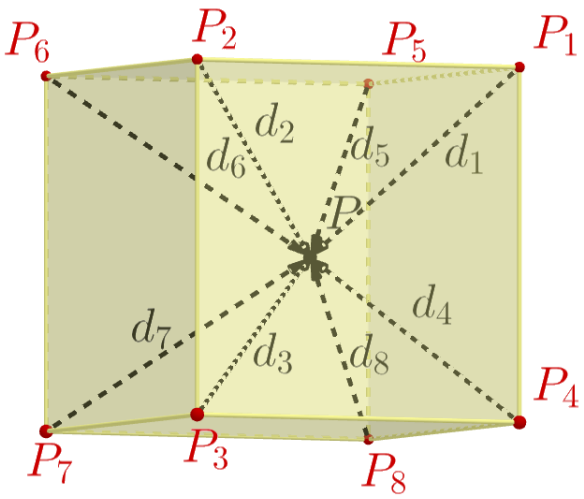


Figure 5.13: Wind interpolation in 3D.

No climb authorized A first test has been done when no climb is authorized to compare the results with the 2D method. Figure 5.14 and Table 5.1 show the results obtained.

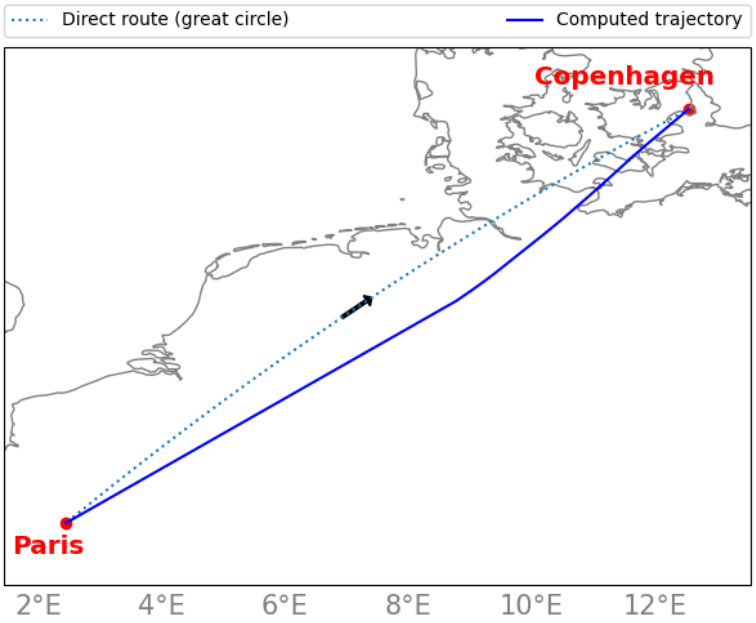


Figure 5.14: Results of the study case in 3D from Paris to Copenhagen without climb. The dotted line is the direct trajectory and the solid line is the trajectory calculated taking into account the wind.

	Direct route (great circle)	Results from 2D method	Computed trajectory	Gain/ Loss
Distance flown	544.88 NM	550.82 NM	555.26 NM	1.91 %
Flight time	1h12min01s	1h11min40s	1h11min43s	-0.42 %

Table 5.1: Quantitative results for the 3D case study from Paris to Copenhagen with no uphill clearance. The results obtained with the 2D method are recalled. The gain or loss in 3D is given in the last column.

Table 5.1 shows that results are comparable to those obtained in 2D are obtained, however with a much longer execution time.

One climb authorized Another test has been done when one climb is authorized. Figure 5.15 shows the results obtained. The slope expected to climb is 3 degrees and as shown by Figure 5.16 it is respected.

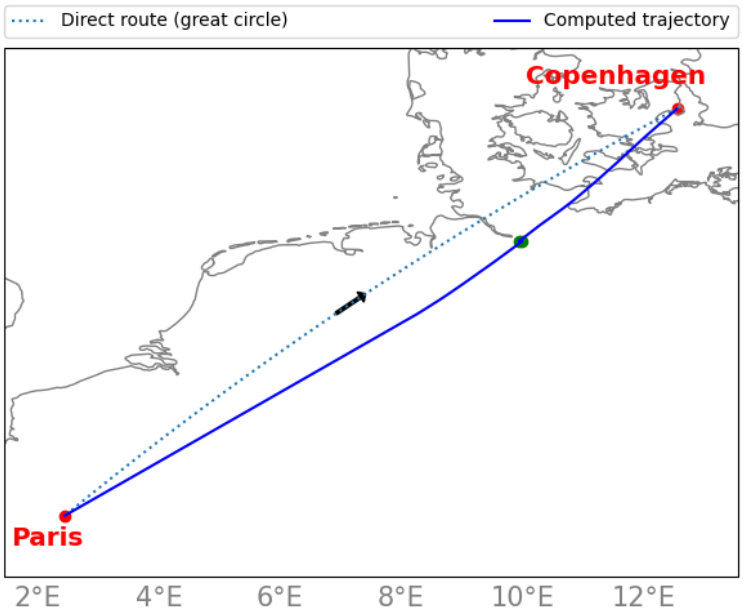


Figure 5.15: Result of the 3D case study from Paris to Copenhagen with an authorized climb. The point in green is the climb point.

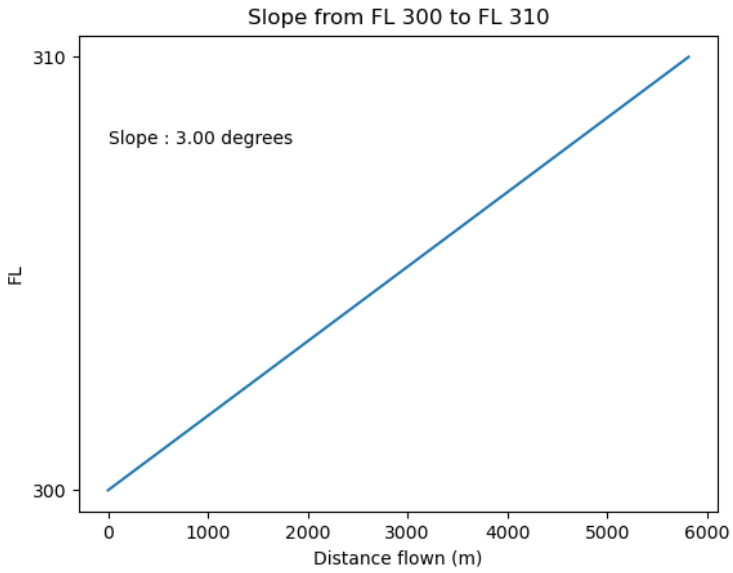


Figure 5.16: Slope used in 3D study case to climb from FL 300 to FL 310.

The wind encountered at FL 310 is not very different from the wind encountered at FL 300 as mentioned at the beginning of this chapter (cf Section 5.1.1), so the trajectory obtained is not very different from the one obtained when no climb is allowed. However, the result has the advantage of taking into account the possibility to climb to gain on fuel consumption.

5.3.4 Critical viewpoint

To conclude on this method, it gives good results on the benchmarks with quite satisfactory computation time for 3D problem. However, it should be improved on real wind cases even if it gives trajectories that look like those given by the 2D method which gives good results. The high sensitivity of the algorithm's parameters is an important point to take into account because they can prevent good results.

Moreover, the data used are not real aircraft data. For example, the slopes used are not from aircraft data and the time needed to fly before being able to climb is not associated with real aircraft data.

Chapter 6

Conclusion

The aim of this internship was to develop efficient tools to compute optimal wind trajectories in 2D and 3D during the cruise phase.

After a broad state of the art of trajectory prediction techniques, it turned out that methods initially developed for the robotics domain could be successfully adapted to the problem under study.

The resolution of the 2D problem is particularly efficient, giving satisfactory results in a short computation time. The resolution of the 3D problem required many adaptations to meet operational issues.

Perspectives are possible to continue the work done during this internship.

First, it is necessary to take into account real aircraft data and to compare the trajectories obtained with what is done every day by pilots, by taking into account airways too. Moreover, the 3D trajectories are not smooth and should become so.

In a second step, obstacle avoidance has to be added to the algorithm in order to take into account dangerous weather areas like storms or areas favorable to the appearance of condensation trails. The algorithms used during this internship can then be reused as they were initially created for robots wishing to avoid obstacles. Moreover, after having worked in 2D and then in 3D, it is necessary to work in 4D to take into account a dynamic environment and dynamic wind. For this purpose, different approaches can be considered, such as using an RRTx type algorithm which is a variant of RRT able to take into account dynamic environments or increasing the dimension of the search space (moving to 4D).

In addition, weather forecasts are subject to many uncertainties. These uncertainties must be taken into account to improve the robustness of the method.

Finally, a more elaborate and user-friendly graphical interface, especially for the launching of the algorithm, would be welcome.

List of Acronyms

BADA Base of aircraft data, from Eurocontrol, provides realistic data on aircraft to compute their performances. 24

FL Flight Level, aircraft's altitude at standard air pressure, expressed in hundreds of feet. 10, 22, 51, 55, 56, 64, 66, 69

FMS Flight Management System, the aircraft's on-board computer which automates many calculations such as the trajectory between two waypoints. 46

FMT* Fast Marching Tree. 4, 5, 9, 10, 13, 15, 28, 42, 43, 45–50, 52, 56, 57, 62

HDR "Habilitation à Diriger des Recherches", the highest French degree, allowing the supervision of young researchers and in particular the supervision of PhD students. 20

LPA Light Propagation Algorithm. 9, 29, 30

PC Personal Computer. 39

PhD Doctor of Philosophy, represents the highest university degree. 3, 20, 21, 29, 31, 33, 35, 38, 53, 80

PRM Probabilistic RoadMap. 26

RAM Random Access Memory, computer memory in which the processed information can be stored. 39

RRT Rapidly-exploring Random Tree. 8–10, 26–28, 57, 59, 60, 71

RRT* Asymptotically optimal variant of RRT. 4, 5, 9, 15, 27, 28, 43, 57, 59–61, 63

SESAR Single European Sky ATM (Air Traffic Management) Research, collaborative project on a European scale to improve and rethink European airspace and its management. 17

T-RRT Transition based RRT. 8, 9, 15, 28, 57, 59–62, 87

UAV Unmanned Aerial Vehicle. 20, 42

Bibliography

[1] A. Chakravarty, "Four-dimensional fuel-optimal guidance in the presence of winds," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 1, pp. 16–22, Jan. 1985.

[2] B. Sridhar, H. K. Ng, and N. Y. Chen, "Aircraft trajectory optimization and contrails avoidance in the presence of winds," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 5, pp. 1577–1584, Sep. 2011.

[3] K. Palopo, R. D. Windhorst, S. Suharwardy, and H.-T. Lee, "Wind-optimal routing in the national airspace system," *Journal of Aircraft*, vol. 47, no. 5, pp. 1584–1592, Sep. 2010.

[4] M. R. Jardin and A. E. Bryson, "Neighboring optimal aircraft guidance in winds," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 710–715, Jul. 2001.

[5] L. S. Pontryagin, *Mathematical Theory of Optimal Processes*. CRC Press, Mar. 1987, google-Books-ID: kwzq0F4cBVAC.

[6] E. Zermelo, "Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung," *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 11, no. 2, pp. 114–124, 1931.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[9] K. Legrand, S. Puechmorel, D. Delahaye, and Y. Zhu, "Robust aircraft optimal trajectory in the presence of wind," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 11, pp. 30–38, Nov. 2018.

[10] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, Apr. 1958.

[11] C. Nissoux, T. Simeon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, vol. 3, Oct. 1999, pp. 1316–1321 vol.3.

[12] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[13] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[15] P. Pharpatara, B. Hérissé, and Y. Bestaoui, "3-D Trajectory planning of aerial vehicles using RRT*," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1116–1123, May 2017.

[16] A. Chakrabarty and J. Langelaan, "UAV flight path planning in time varying complex wind-fields," in *2013 American Control Conference*. Washington, DC: IEEE, Jun. 2013, pp. 2568–2574.

[17] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, Jun. 2015.

[18] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based RRT for path planning in continuous cost spaces," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice: IEEE, Sep. 2008, pp. 2145–2150.

[19] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts." *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, Feb. 1996.

[20] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations," *Proceedings of the National Academy of Sciences*, vol. 98, no. 20, pp. 11 069–11 074, Sep. 2001.

[21] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, no. 2, pp. 269–277, May 1995.

[22] J.-M. Mirebeau and J. Portegies, "Hamiltonian fast marching: A numerical solver for anisotropic and non-holonomic Eikonal PDEs," *Image Processing On Line*, vol. 9, pp. 47–93, Feb. 2019.

[23] B. Girardet, "Trafic aérien : détermination optimale et globale des trajectoires d'avion en présence de vent," PhD thesis, INSA, Toulouse, Dec. 2014.

[24] P. G. Lelièvre, C. G. Farquharson, and C. A. Hurich, "Computing first-arrival seismic traveltimes on unstructured 3-D tetrahedral grids using the fast marching method: Computing first-arrivals on tetrahedral grids," *Geophysical Journal International*, vol. 184, no. 2, pp. 885–896, Feb. 2011.

[25] N. E. Dougui, "Planification de trajectoires avion : approche par analogie lumineuse." PhD thesis, Université Paul Sabatier - Toulouse III, Dec. 2011.

[26] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, ser. ACM '68. New York, NY, USA: Association for Computing Machinery, Jan. 1968, pp. 517–524.

[27] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 1478–1483.

[28] J. Talbot, "Going against the flow: Finding the optimal path," *European Journal of Physics*, vol. 31, no. 1, pp. 205–213, Jan. 2010.

Appendix A

Approximation of great circle distance

To approximate the great circle distances, some multiplying factors depending on the propagation direction are used. Eight factors are computed according to the direction, as shown in Figure A.1.

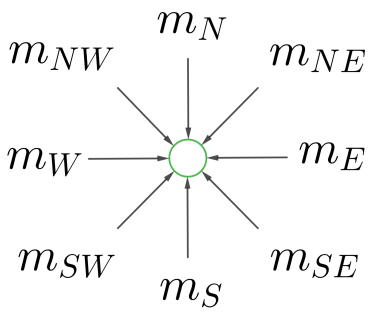


Figure A.1: Different multiplying factors depending on the propagation direction.

where

$$m_{direction} = \frac{\text{distance on the Cartesian plane} \approx \text{distance at the equator}}{\text{distance on the Earth}} \tag{A.1}$$

The remainder of this appendix details the calculation of factors for grid accuracy h. These computations are done thanks to the formula of the true great circle distance between two points $A(lat_A, lon_A)$ and $B(lat_B, lon_B)$:

$$d_{ortho} = R * c_{radians}(km) \tag{A.2}$$
$$= 60 * c_{degrees}(NM) \tag{A.3}$$

where $R = 6371km$ is the Earth radius, $c_{radians}$ and $c_{degrees}$ are respectively c expressed in radians and in degrees with:

$$c = \arccos(\sin(lat_A) * \sin(lat_B) + \cos(lat_A) * \cos(lat_B) * \cos(lon_B - lon_A)) \tag{A.4}$$

Let $c_{EQ_{direction}}$ the distance in degrees or in radians at the equator between from a point to an adjacent one in the direction cited.

$$c_{EQ_N} = \arccos(\sin(0) * \sin(h) + \cos(0) * \cos(h) * \cos(0)) = h \quad \text{(A.5)}$$
$$c_{EQ_S} = \arccos(\sin(0) * \sin(-h) + \cos(0) * \cos(-h) * \cos(0)) = h \quad \text{(A.6)}$$
$$c_{EQ_E} = \arccos(\sin(0) * \sin(0) + \cos(0) * \cos(0) * \cos(h)) = h \quad \text{(A.7)}$$
$$c_{EQ_W} = \arccos(\sin(0) * \sin(0) + \cos(0) * \cos(0) * \cos(-h)) = h \quad \text{(A.8)}$$

$$c_{EQ_{NW}} = \arccos(\sin(0) * \sin(h) + \cos(0) * \cos(h) * \cos(-h)) = \arccos(\cos(h)^2) \quad \text{(A.9)}$$
$$c_{EQ_{NE}} = \arccos(\sin(0) * \sin(h) + \cos(0) * \cos(h) * \cos(h)) = \arccos(\cos(h)^2) \quad \text{(A.10)}$$
$$c_{EQ_{SW}} = \arccos(\sin(0) * \sin(-h) + \cos(0) * \cos(-h) * \cos(-h)) = \arccos(\cos(h)^2) \quad \text{(A.11)}$$
$$c_{EQ_{SE}} = \arccos(\sin(0) * \sin(-h) + \cos(0) * \cos(-h) * \cos(h)) = \arccos(\cos(h)^2) \quad \text{(A.12)}$$
$$\text{(A.13)}$$

Some remarks can be done:

- $c_{EQ_N} = c_{EQ_S} = c_{EQ_E} = c_{EQ_W}$
- $c_{EQ_{NW}} = c_{EQ_{NE}} = c_{EQ_{SW}} = c_{EQ_{SE}}$

Let us now detail the different factors for a point $A(lat_A, lon_A)$ from these constants.

$$m_N = \frac{c_{EQ_N}}{\arccos(\sin(lat_A) * \sin(lat_A + h) + \cos(lat_A) * \cos(lat_A + h))} \quad \text{(A.14)}$$
$$m_S = \frac{c_{EQ_S}}{\arccos(\sin(lat_A) * \sin(lat_A - h) + \cos(lat_A) * \cos(lat_A - h))} \quad \text{(A.15)}$$
$$m_E = \frac{c_{EQ_E}}{\arccos(\sin(lat_A)^2 + \cos(lat_A)^2 * \cos(h))} \quad \text{(A.16)}$$
$$m_W = \frac{c_{EQ_W}}{\arccos(\sin(lat_A)^2 + \cos(lat_A)^2 * \cos(h))} \quad \text{(A.17)}$$

$$m_{NE} = \frac{c_{EQ_{NE}}}{\arccos(\sin(lat_A) * \sin(lat_A + h) + \cos(lat_A) * \cos(lat_A + h) * \cos(h)} \quad \text{(A.18)}$$
$$m_{NW} = \frac{c_{EQ_{NW}}}{\arccos(\sin(lat_A) * \sin(lat_A + h) + \cos(lat_A) * \cos(lat_A + h) * \cos(h)} \quad \text{(A.19)}$$
$$m_{SE} = \frac{c_{EQ_{SE}}}{\arccos(\sin(lat_A) * \sin(lat_A - h) + \cos(lat_A) * \cos(lat_A - h) * \cos(h)} \quad \text{(A.20)}$$
$$m_{SW} = \frac{c_{EQ_{SW}}}{\arccos(\sin(lat_A) * \sin(lat_A - h) + \cos(lat_A) * \cos(lat_A - h) * \cos(h)} \quad \text{(A.21)}$$
$$\text{(A.22)}$$

Once again, some factors are equal:

- $m_E = m_W$
- $m_{NE} = m_{NW}$
- $m_{SE} = m_{SW}$

Appendix B

Zermelo's problem

B.1 Problem presentation

Zermelo's problem has been proposed by Ernst Zermelo [6] in 1931. It consists in finding the optimal trajectory for a boat from a point A to a point B taking into account currents. This problem can be transposed to an aircraft in presence of wind.

Problem statement A boat must travel through an area with strong ocean currents. The speed of the current is assumed to be known, it is a function of the position of the boat by its Cartesian coordinates (x, y). The components of the current speed are noted:

$$W_x = W_x(x, y) \tag{B.1}$$
$$W_y = W_y(x, y) \tag{B.2}$$

where x, y are the coordinates and W_x, W_y the components of the current in the x and y direction respectively. The norm of the relative speed of the boat with respect to the water is constant and given by V.

The problem is to steer the boat in such a way as to minimize the travel time necessary to go from point A to point B. The equations of motion of the boat are given by :

$$\dot{x} = V * \cos(\theta) + W_x(x, y) \tag{B.3}$$
$$\dot{y} = V * \sin(\theta) + W_y(x, y) \tag{B.4}$$

where θ is the angle between the boat speed vector and the horizontal axis.

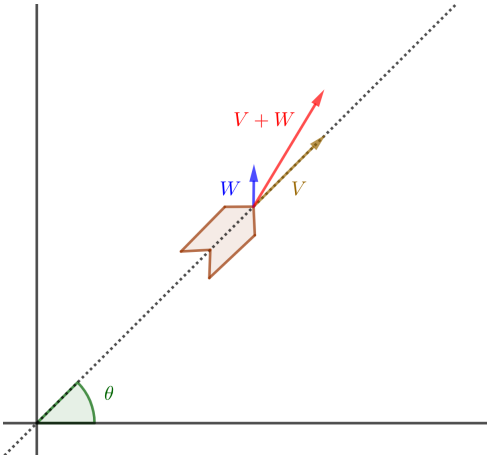


Figure B.1: Zermelo's problem setup.

Mathematical problem statement Mathematically, the problem is expressed as follows:

$$\min_{\theta} \; t_f = \int_0^{t_f} dt \tag{B.5a}$$

$$\text{under} \qquad \dot{x} = V * \cos(\theta) + W_x(x, y) \tag{B.5b}$$

$$\dot{y} = V * \sin(\theta) + W_y(x, y) \tag{B.5c}$$

$$(x(0), y(0)) = (x_0, y_0) = A \tag{B.5d}$$

$$(x(t_f), y(t_f)) = (x_f, y_f) = B \tag{B.5e}$$

B.2 Problem resolution

This problem has been addressed and solved in some particular cases of wind. Two cases will be detailed hereafter: constant wind and linear wind. These resolutions are done in the PhD thesis [23] and in the paper [28] for instance.

B.2.1 Constant wind case

Assume a constant wind.

$$\forall (x, y), W_x(x, y) = W_x, W_y(x, y) = W_y \tag{B.6}$$

It has been proved in [23] that the optimal trajectory in this case is the straight line between the starting point and the end point, whatever the value of the constant wind. Indeed, it is proved that at the optimal, the derivate of theta according to the time is zero.

$$\dot{\theta} = 0 \tag{B.7}$$

B.2.2 Linear wind case

Assume a linear wind on x-axis according to y.

$$W_x(x, y) = W * y \tag{B.8}$$
$$W_y(x, y) = 0 \tag{B.9}$$

It has been proven in [23] that the optimal trajectory follows these two equations:

$$x(\theta) = \frac{V}{2W}\left(\frac{1}{\cos\theta_f}(\tan\theta - \tan\theta_f) + \tan\theta\left(\frac{1}{\cos\theta_f} - \frac{1}{\cos\theta}\right) - \ln\left(\frac{\tan\theta_f + \frac{1}{\cos\theta_f}}{\tan\theta + \frac{1}{\cos\theta}}\right)\right) \tag{B.10}$$

$$y(\theta) = \frac{V}{W}\left(\frac{1}{\cos\theta} - \frac{1}{\cos\theta_f}\right) \tag{B.11}$$

where θ_f is the value of the angle θ at t_f.

Here are some examples of optimal trajectories for Zermelo problem for different wind value:

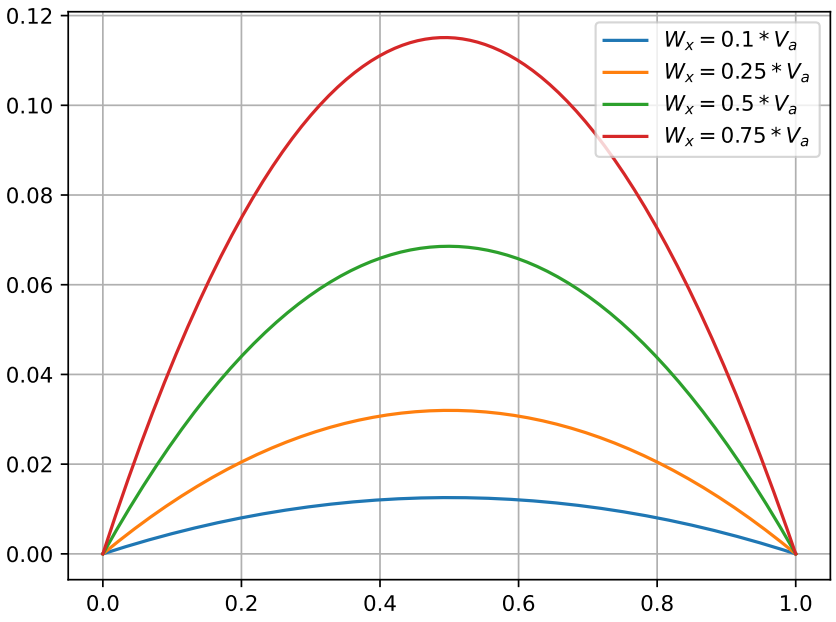


Figure B.2: Examples of optimal trajectories for Zermelo problem in the case of linear wind.

Appendix C

Computation of tangent circle to two segments

The problem here is to find a circle of radius r tangent to the two segments in the Figure C.1.

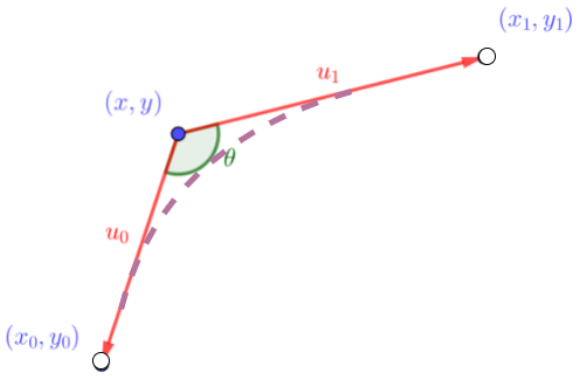


Figure C.1: Tangent circle to crossing segments - Problem statement.

Let:

$$\overrightarrow{u_0} \begin{pmatrix} x_0 - x \\ y_0 - y \end{pmatrix} \tag{C.1}$$

$$\overrightarrow{u_1} \begin{pmatrix} x_1 - x \\ y_1 - y \end{pmatrix} \tag{C.2}$$

Let θ the angle between u_1 and u_0.

The center of the circle is on the theta bisector.

Let $C(x_C, y_C)$ the center of the circle. Let α be one of the angles of the right-angled triangle where one of the angles is $\frac{\theta}{2}$. Since the sum of the angles of the triangle is π:

$$\alpha = \frac{\pi}{2} - \frac{\theta}{2} \tag{C.3}$$

The Figure C.2 summarizes these different elements.

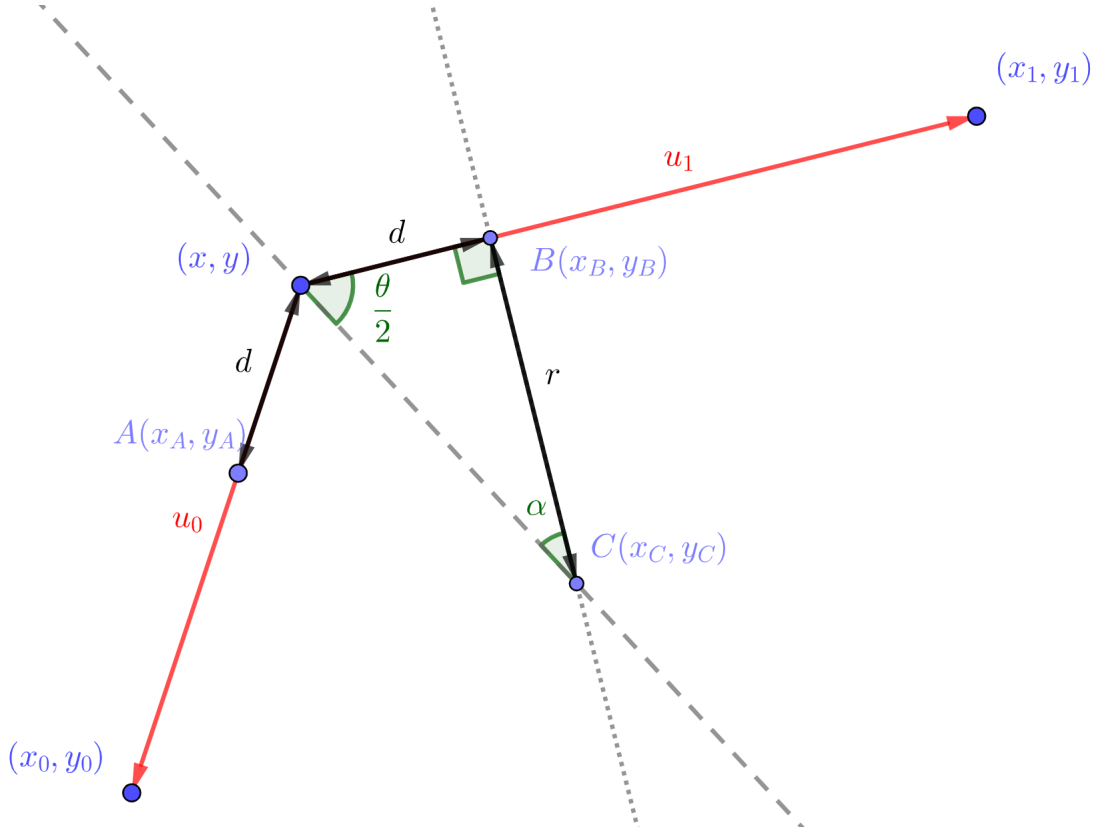


Figure C.2: Angles definition.

The distance d can be computed thanks to the following equation.

$$d = r * \tan(\alpha) \tag{C.4}$$

Then,

$$x_A = x + d * \frac{u_{0x}}{||u_0||} \tag{C.5}$$

$$x_B = x + d * \frac{u_{1x}}{||u_1||} \tag{C.6}$$

$$y_A = y + d * \frac{u_{0y}}{||u_0||} \tag{C.7}$$

$$y_B = y + d * \frac{u_{1y}}{||u_1||} \tag{C.8}$$

Point C can be defined thanks to the following linear system.

$$(x_A - x)x_C + (y_A - y)y_C = x_A(x_A - x) + y_A(y_A - y) \tag{C.9}$$
$$(x_B - x)x_C + (y_B - y)y_C = x_B(x_B - x) + y_B(y_B - y) \tag{C.10}$$

And so the result is obtained as shown by Figure C.3.

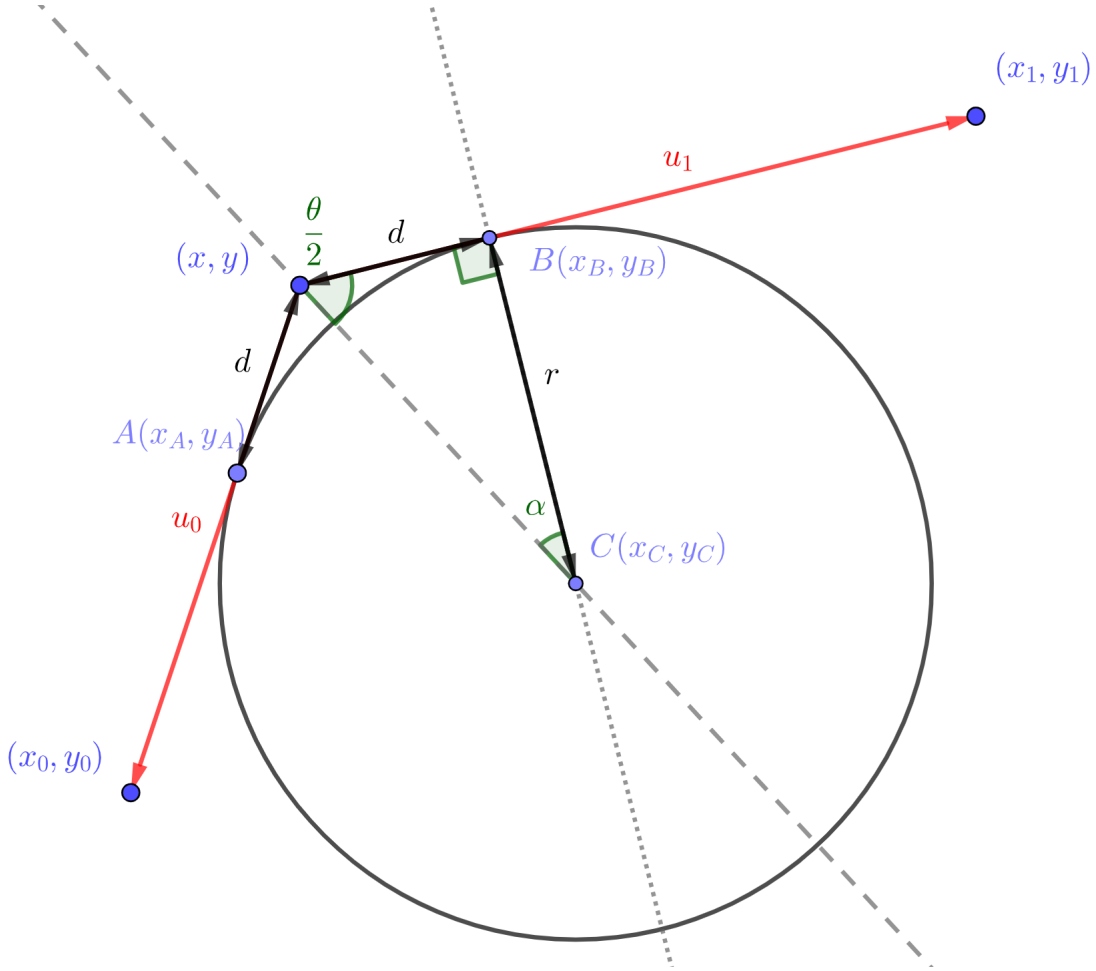


Figure C.3: Result of tangent circle to crossing segments.

Appendix D

Transition test for T-RRT

Algorithm 5: T-RRT transition test.

Input: c_i, c_j, d_{ij}
Output: *accepted*

$accepted \leftarrow false$
if $c_j > c_{max}$ **then**
  $accepted \leftarrow false$
else
  if $c_j < c_i$ **then**
    $accepted \leftarrow true$
  else
    $p \leftarrow \exp(\frac{-(c_j - c_i)/d_{ij}}{K * T})$ **if** $Rand(0,1) < p$ **then**
      $T \leftarrow \frac{T}{\alpha}$
      $n_{fail} \leftarrow 0$
      $accepted \leftarrow true$
    else
      if $n_{fail} > n_{fail_{max}}$ **then** ; `// Auto-tuning`
        $T * \alpha$
        $n_{fail} \leftarrow 0$
      else
        $n_{fail} \leftarrow n_{fail} + 1$
      end
      $accepted \leftarrow false$
    end
  end
end

Page intentionally left blank.

Name or the Intern: Céline Demouge

Course: IENAC18 SITA IA

Company: ENAC

Internship subject: Finding geodesic trajectories thanks to propagation methods

CONFIDENTIALITY QUESTIONNAIRE

The internship will include the writing of a thesis and its defence before a board of examiners. Some confidential information on your Company might be included in it.

- Please let us know of your requirements regarding the circulation of the thesis and the organisation of the oral presentation. In the absence of a reply from your company, the thesis will be considered non confidential and the oral presentation will be public.
- We would like to draw your attention to the fact that a thesis is a precious source of information on companies and employment opportunities for our students. It is also a mean to enhance communication on your company towards future students.
- A confidential thesis is referenced but not accessible despite its technical and scientific value. We encourage our students to include confidential information in an annex to enable access to the thesis without disseminating confidential data.

1- Thesis Confidentiality

☒ This thesis is not confidential

☐ This thesis is confidential for a period of 5 years after the oral presentation

☐ This thesis is confidential indefinitely

2- Oral Presentation

☒ All professionals interested by the subject matter can attend

☐ Only members of the jury and participants vetted by the company can attend

For the company,
Date : 04/08/2021

The student,
Date : 04/08/2021