Long Range Evolution-based Path Planning for UAVs through Realistic Weather Environments

Juan Carlos Rubio Torroella

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington

2004

Program Authorized to Offer Degree: Aeronautics and Astronautics

University of Washington

Graduate School

This is to certify that I have examined this copy of a master's thesis by

Juan Carlos Rubio Torroella

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

____

Rolf Rysdyk

____

Juris Vagners

Date: ________________________________

University of Washington

Abstract

Long Range Evolution-based Path Planning for UAVs through Realistic Weather Environments

Juan Carlos Rubio Torroella

Co-Chairs of Supervisory Committee:

Assistant Professor Rolf Rysdyk
Aeronautics and Astronautics

Professor Emeritus Juris Vagners
Aeronautics and Astronautics

The application of an evolution-based path planner for UAVs in long range flights is presented. The planner makes use of wind information from actual weather forecast databases and considers areas of potential icing hazard. Weather variability in both 3-D space and time is taken into account for the planning of the complete path. Examples show how the planning system makes use of favorable winds for fuel consumption benefit, avoids areas of potential icing, and is able to follow weather reconnaissance type of trajectories by visiting areas of interest as well as scanning the atmosphere with desired vertical maneuvers. A spherical Earth model for path generation has been implemented within the planner for long range applications. Vehicle performance and fuel consumption models are integrated into the planner with the use of performance tables, allowing the system to plan for any type of vehicle. The planner is capable of making in-flight modifications to further refine or adapt its path to updated weather information.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to Juris and Rolf for inviting me to participate in this project. It has been a great experience that has given me the satisfaction of working on an airplane-related application, which was my main motivation to pursue this Master's degree. And very important, it has triggered my interest in research areas directly involving UAV systems.

I also want to thank Anawat, Bryan, Dong, David and Sean with whom I have enjoyed working and sharing great times. Thanks to all friends, colleague students, professors and staff, who make the UW a great University.

Thanks to my family and friends in Mexico who have been in continuous contact during these couple of years.

And most of all, I am very grateful to my wife, Illiana, who has been incredibly supportive in my pursue of this goal.

This work is part of continuous research at the University of Washington Autonomous Flight Systems Laboratory. It is by no means a result of a single contribution, but of added contributions of several Graduate Students that have participated in the past. This research was funded by the NOAA National Sea Grant College Program under Grant NA16RG1044, project code R/0T-22. I also acknowledge the support received during my Master's degree studies from Consejo Nacional de Ciencia y Tecnología (CONACyT) and the US-Mexico Commission for Educational and Cultural Exchange (Fulbright Commission).

DEDICATION

To my wife, Illiana

Chapter 1

INTRODUCTION

Over recent years, interest in the use of Unmanned Aerial Vehicles (UAVs) has continuously increased together with the research and development on these systems. These vehicles offer potential benefits for a variety of applications where otherwise high cost and/or risky operations are involved. One of these applications is the use of UAVs in weather reconnaissance programs which motivated the research presented here.

The accuracy of numerical weather forecasts depends heavily on observation data available to the models. The sparsity of observational data over the oceans is a major cause for poor forecasting. At the present time, observations over these areas are obtained primarily from three different sources. Satellite data provide a good amount of information in the upper troposphere (such as water vapor and cloud track winds) and near the surface (scatterometer winds). However, they are limited in providing data within cloud systems. A second source of observations is measurement from commercial aviation aircraft flying across the oceans, but this information is in general limited to jet stream level. Thirdly, buoy and ship observations provide additional observation, but in a sparse capacity. This in general leaves the lower two-thirds of the troposphere lacking weather sampling over the oceans (0 to 6000m), compared to a much larger information database available over terrain surfaces where radiosondes lifted by balloons are launched twice a day. Thus, the lack of tropospheric sampling over the oceans creates a major area of potential for application of model sized UAVs, which offer continuous, long range, economical and multi-level observation of the atmosphere.

Among other systems, the Aerosonde and Seascan UAVs have shown promise on long range flights suitable for weather reconnaissance missions. The Aerosonde was the first unmanned aircraft (and smallest) to cross the Atlantic Ocean in 1998, taking off from Bell Island, Newfoundland and landing on South Uist in Scotland's Western Isles, a distance of 3270km [1]. In this successful flight, however, the vehicle flight computer was loaded by the research team with pre-defined way points based on the most recent available weather conditions and forecast, which the vehicle followed without further modifications. Autonomous planning and navigation capabilities are required for proactive future success of long range flights of small UAVs. Adaptation of the path to changing environment conditions while en-route is essential in future UAV systems. Autonomy will reduce the human operation work load which is still very high in unmanned vehicle flights.

The research presented here involves the integration of weather information from actual databases and forecast algorithms with evolutionary path planning techniques for long range autonomous flights. It addresses a 4-dimensional problem of space and time, requiring adaptation to account for unexpected changes in the environment. The objective is the planning of trajectories that balance fuel consumption, icing hazard avoidance and observation requirements, considering performance characteristics of the target vehicle. Preliminary results of this research were presented in [2]. Application of evolutionary planning techniques to multiple autonomous vehicles [3] is a significant part of the research at the University of Washington Autonomous Flight Systems Laboratory.

Chapter 2

EVOLUTION-BASED PATH PLANNER

This chapter presents the main concepts behind the path planner. It begins with some background of Evolutionary Computation methods and concludes with a description of the algorithm implemented in the Path Planner.

2.1 Evolutionary Computation

Evolutionary Computation (EC) is a class of global optimization techniques which mimic the evolution process witnessed in nature. Since the idea was introduced in the 1960s, several EC-based techniques such as genetic algorithms and evolutionary programming have been developed [4]. The techniques are slightly different in the actual implementations. They, however, use the same metaphor of mapping the problem to be solved onto a simple model of evolution. The mapping of terms used in evolutionary computation to the problem solution domain is shown in Figure 2.1.

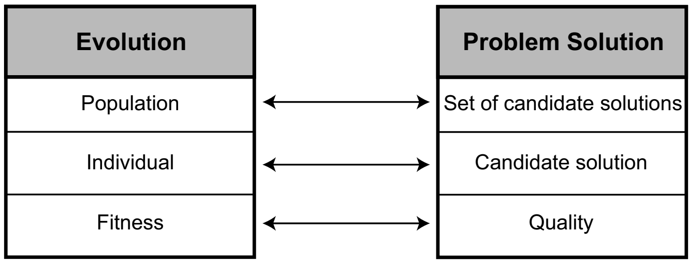

Figure 2.1: Mapping of problem solution onto a model of evolution.

Algorithms which are developed based on the concept of Evolutionary Computation are generally called Evolutionary Algorithms (EA). The problem solving process of an evolutionary algorithm is done by successively modifying a population of individuals each of which represents a candidate solution of the problem being solved in search of the optimal solution. The starting population is initialized randomly by an algorithm-dependent method. The trial solutions move their way through the search space by evolving the population toward the region with the highest *fitness* by means of randomized *mutation*, and *selection*. Each generation of the evolution consists of the following three phases: fitness evaluation, selection, and production of offspring. Fitness is a measure of the quality of a candidate solution represented by an individual in the population. Mutation is a process to alter an individual using a specific mechanism. The selection process is a procedure to reduce the size of the population by eliminating some individuals with lower fitness. The individuals that survive through the selection process are the *parents* which are used to create *offspring*, new individuals, for the next *generation*.

Regardless of their implementation, all evolutionary algorithms have the same fundamental procedure. The following are the primary steps in an evolutionary algorithm solving the generalized optimization problem:

1. Generate an initial population, $\mathbf{P}(n = 0)$, of $\mu + \lambda$ individuals. This initial population can be created completely at random or based on specific knowledge, or a combination of the two.

2. Evaluate the fitness of each individual using a fitness function $F(\vec{P}^j)$ where $\vec{P}^j$ is an individual in the population.

3. Select μ individuals out of the entire population based on their fitness to be parents used to generate offspring in the next step. The selection scheme can be either deterministic or probabilistic.

4. Generate λ offspring from the μ parents. Each of the offspring can be spawned by mutating a copy of one of the parents using a set of mutation mechanisms or by combining a pair of the parents.

5. Set $n = n + 1$ and go to step 2 until termination criterion is reached or the planning time expires.

The fitness function F used to evaluate fitness of each individual must be a combination of an objective function and a constraint function. The fitness function can also be in the form of a vector of functions $\vec{F}$. The detailed implementation of these steps largely depends on the problem.

Two popular EC-based techniques are Genetic Algorithms and Evolutionary Programming. Genetic Algorithms (GA) are the most well-known type of evolution-based optimization techniques developed by Holland [5], a computer scientist at the University of Michigan. The main idea behind them is that individuals in a population are encoded as ’chromosomes’ (in bit form) through which mutation and propagation occurs based on external fitness criteria. On the other hand, Evolutionary Programming (the method adopted for the path planner used here) models candidate solutions directly in the output space — the search space of the problem being considered; that is, each individual in the population is represented by the variables of interest for the problem, which are subject to direct modifications through the evolution process. The original Evolutionary Programming (EP) idea was introduced by L. J. Fogel, and was later extended by D. B. Fogel [4].

2.1.1 Evolution-based Path Planning

There are many advantages of using EC-based techniques in path planning problems. First, they are open architecture techniques. It is easy to solve optimization problems with linear or nonlinear performance functions and constraints. Second, EC-based planning approaches are continual adaptation techniques. They can run continuously as the vehicles execute the plans, and handle changes in the operating environment and vehicle capabilities. Furthermore, EC-based techniques respond to the changing environment quickly because they do not have to recompute the entire plans. The current plan is adapted in response to the changes.

EA based techniques have shown promise for solving optimization problems where complex and variable environmental characteristics are an important factor. Much interest has existed in the application of these techniques for path planning problems.

Fogel [6] applied Evolutionary Programming (EP) to an optimal routing problem of autonomous underwater vehicles (AUVs). This work shows that the planning algorithm can handle unexpected changes in dynamic environments. Fogel also considers a number of problems including multiple goal locations, detection avoidance, and cooperative goal observation for a pair of AUVs. Solution to these complex problems was achieved by only modifying the performance objective function.

Xiao [7] presented an adaptive evolutionary path planner for mobile robots. This approach combines off-line planning and on-line replanning in the same evolutionary algorithm. In this approach, a path is represented as a set of waypoints chosen at random connecting the initial and goal locations. The probability of selecting different mutation operators is adapted during the search to improve performance.

Capozzi [8] presented an evolutionary technique for path planning of an aerial vehicle in a simulated dynamic environment. The planning algorithm was tested in several complex scenarios: varying terrain, wind variations, dynamic obstacles, and moving targets. The simulation results show that the algorithm can efficiently search simultaneously in space and time to find feasible, near-optimal solutions.

Hacaoglu and Sanderson [9] developed an evolution-based planning algorithm using a multi-resolution path representation. The use of the multi-resolution path representation reduces the complexity of the planning problem and in turn reduces the computational time. They show that the planning system can be applied to mobile robots or manipulators with many degrees of freedom and provides effective results. In addition, they also proposed a multi-path planning algorithm which generates multiple alternative paths simultaneously.

Rathbun and Capozzi [10] developed an evolution-based path planner which explicitly accounts for uncertainties in the environment. In this work, the planning algorithm does not only use the estimate of the obstacle locations in the environment but also the uncertainty associated with the estimate. This work suggests an approximation method to compute the probability of intersection of the vehicle path with an obstacle. This probability of intersection is used in the fitness function to evaluate candidate paths. The proposed planning algorithm shows reasonably good results in comparison to graph-based search methods.

2.2 Evolution-based Path Planning Algorithm

The general concept of the EC-based planning algorithm used in this research is illustrated in Figure 2.2. The algorithm runs in a loop which has three phases. It starts by randomly generating a population of encoded plans. Then it evaluates the fitness value of each plan. The next step is to select the best plan to be the candidate solution for this generation and also select a portion of the plans in the population to be the parents of the next generation based on their fitness values using a selection scheme. The last step is to produce offspring from the parents selected in the previous phase. An offspring is generated by cloning a parent and applying a mutation mechanism to it or by crossover of two parents. This loop is run continuously to update the plan as the optimization process proceeds.

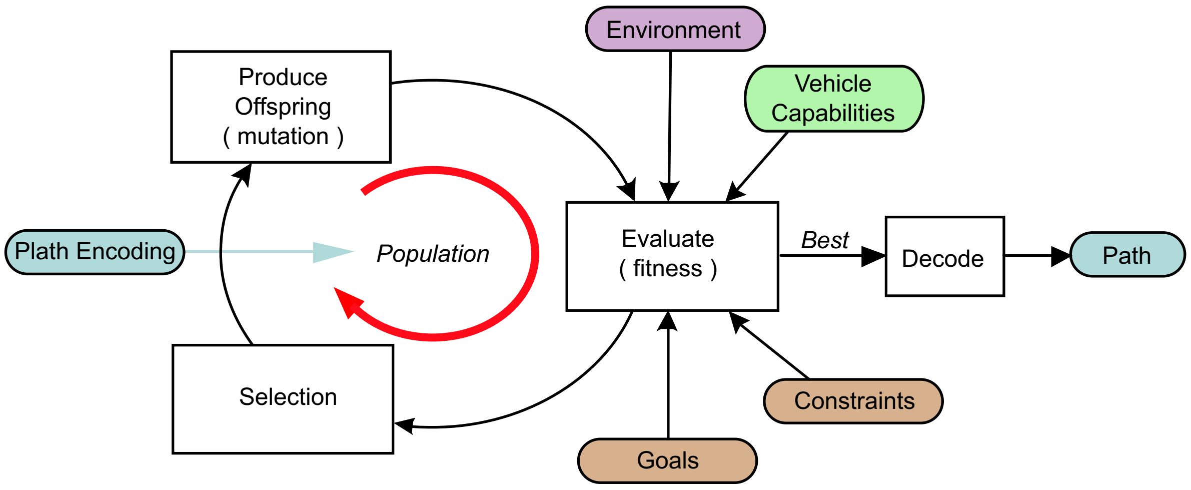

Figure 2.2: Overview of Evolutionary Computation based planning algorithm.

The EC-based planning algorithm used in this research is based on both Genetic Algorithms (GA) and Evolutionary Programming (EP) [11]. Work by Capozzi [12] suggests that the algorithm combining features of both paradigms can improve the performance of the optimization process. The design of an EC-based path planning algorithm involves the following issues: path encoding, fitness evaluation, mutation mechanisms, and selection scheme. The path planning algorithm for a single vehicle used in this research is based on the algorithm described in Rathbun and Capozzi [10].

2.2.1 Path Encoding

In the path planning algorithm used in this research, a path is encoded as a sequence of simple *segments* chained end-to-end, shown in Figure 2.3. For the long range application considered here, construction of segments was extended from a simple flat Earth model to a spherical Earth model. Chapter 3 presents the geometric details for such implementation. The segment parameters are limited to keep motion within the vehicle capabilities; the integration of vehicle performance into the path planner is considered in Chapter 4.

Continuity is maintained between the joining end of a segment and the starting point. Continuity in heading and speed at the join of two segments is not strictly enforced considering that the dimension of the problem is large enough to allow the vehicle's controller to adjust accordingly. We also enforce every path to end at the goal location by adding a necessary number of segments at the end of the last segment to extend the path to reach the goal location. The *go-to-goal* segments are added to a new path after it is created.

2.2.2 Fitness Evaluation

Fitness of a candidate path is the value that represents the performance measure of the path based on the objectives given by the problem. The fitness of individuals in the population must be determined during the evaluation process. It may be computed directly through a fitness function, or inversely as a cost function. Individuals with higher fitness (or lower

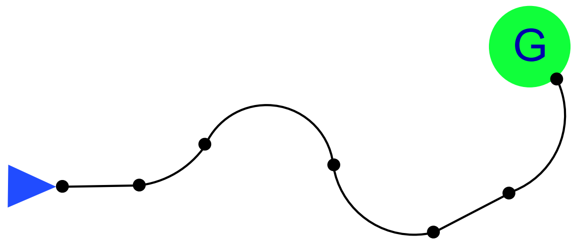

Figure 2.3: An encoded path which is composed of a chain of connected segments.

cost) have more chance to survive during the selection process. The quality of the resulting path depends heavily on this function.

The fitness (cost) function may be defined in several ways. A typical case consists of a weighted linear combination of parameterized terms which represent the mission specifications and constraints. Another approach consists of multi-objective optimization methods; rather than a single function, individual non-linear functions are designed for each parameter considered in the problem which allows one to define different competition methods between paths. The objective of any approach is to define fitness functions that may lead the solution to an expected performance. The planning examples presented in chapter 6 have primarily used the second approach by comparing the maximum value of all cost functions of each path. The parameters considered are:

- fuel requirement,
- icing encounters,
- target achievement.

A typical cost function relation between these objectives is represented in Figure 2.4.

2.2.3 Mutation Mechanisms

Mutation mechanisms are essential for the evolution process to improve the fitness of a candidate path generated in each generation and to eventually converge to the optimal solution. A mutation has the effect of randomly moving a candidate solution from one point

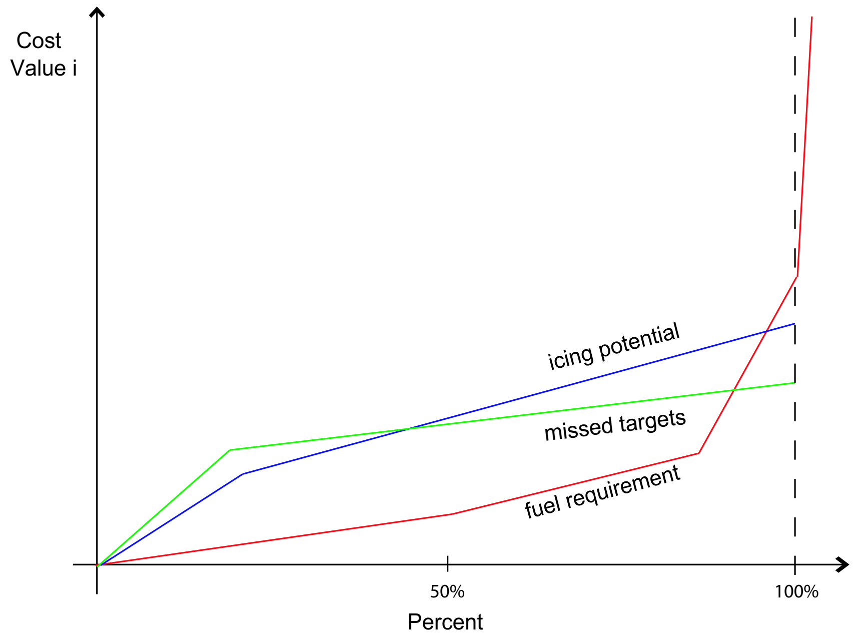

Figure 2.4: Example of Multiple Cost Functions for a Path.

in the search space to another. Therefore, an effective set of mutation mechanisms must be able to move a candidate solution from one point to any other point in the search space by applying a series of these mutation mechanisms to it.

In each generation, offsprings are created either by crossing over two randomly selected individuals in the population, or by copying an individual and mutating it using one of the mutation mechanisms chosen at random. We have five mutation mechanisms which are illustrated in Figure 2.5. The list below explains each of the mutation mechanisms.

- **Mutate 1-Point** - randomly changes the parameters of one or more segments, and then re-locates all the following segments. The first segment to be mutated and the number of segments to be mutated are selected at random.
- **Mutate 2-Point** - randomly changes the parameters of one or more segments, computes the new resultant end point for those segments (similar to Mutate 1-Point), and then connects back to the start of another segment of the path further along. The

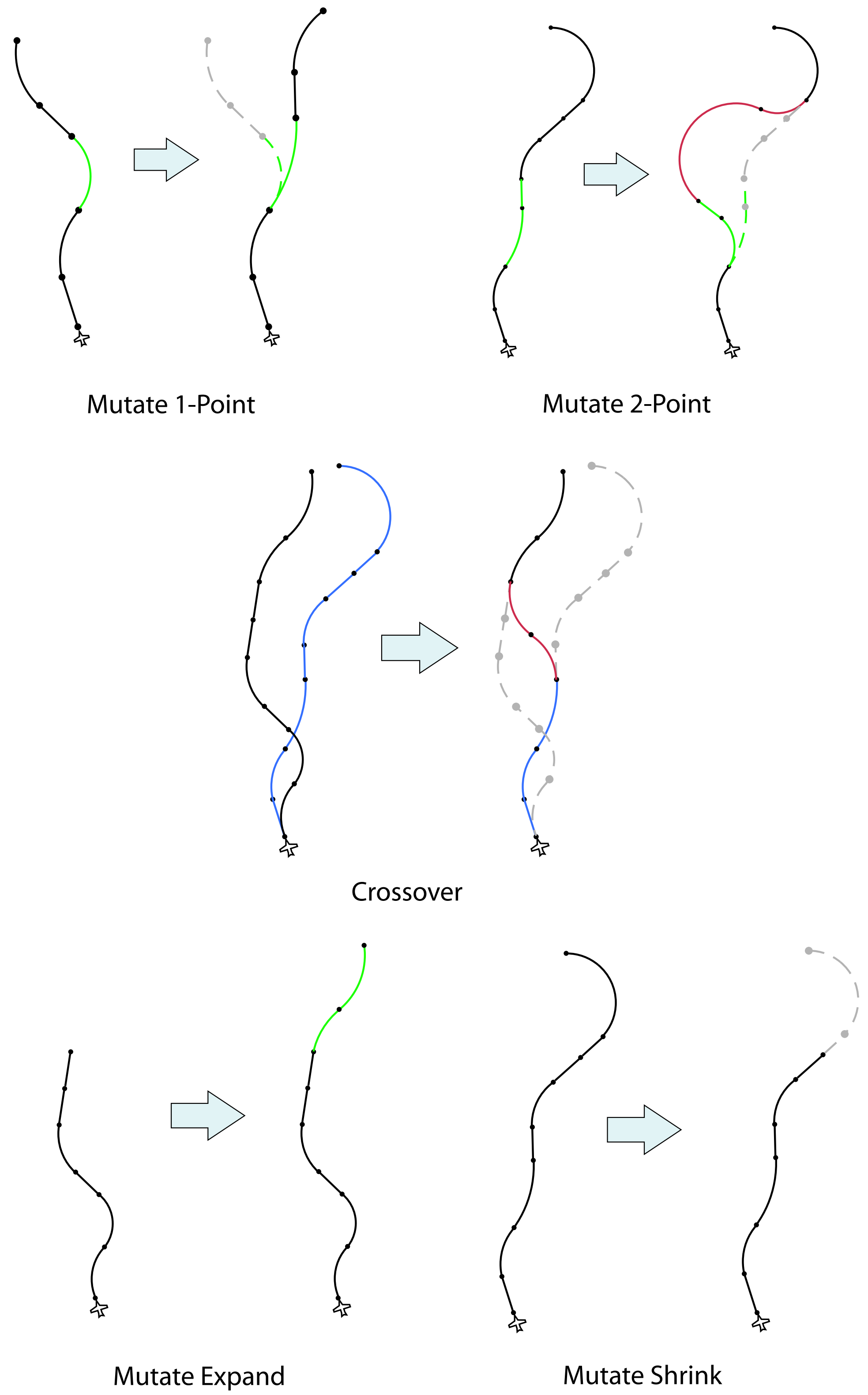

Figure 2.5: Path mutation mechanisms.

beginning segment and the segment joined to are both chosen at random.

- **Crossover** - takes the starting segments of one path and the ending segments of another and join the two sets of segments together.
- **Mutate Expand** - adds one or more randomly created segments onto the end of the path. All original parts of the path are left untouched.
- **Mutate Shrink** - removes one or more segments from the end of the path. The number of segments to be removed are selected at random.

Mutations of segments are not restricted only to their geometry, but may include the speed at which to fly them, thus taking advantage of different fuel consumption rates.

2.2.4 Selection Scheme

Given a population of the size $\mu + \lambda$, the selection scheme is a mechanism for selecting μ individuals to be parents at the n^{th} generation. These μ parents will be used to create λ offsprings later in the next mutation process. The selection scheme is independent of the instantiation of evolutionary computation chosen for a particular application. In this research, a q-fold binary tournament selection scheme is chosen. Figure 2.6 illustrates the procedures of the selection scheme which can be described as follows. For each individual $i \in \{1, 2, \ldots, \mu + \lambda\}$.

1. Draw $q \geq 2$ individuals randomly from the population (excluding individual i) with uniform probability $\frac{1}{\mu+\lambda-1}$. Denote these competitors by the indices $\{i_1, i_2, \ldots, i_q\}$.
2. Compare individual i's fitness against each of the competitors, i_j, $j \in 1, 2, \ldots, q$. Whenever the fitness of individual i is not worse than that of competitor i_j, individual i receives a point.

The score of each individual received during the tournament is an integer in the range [0,q]. After the scores of all individuals are determined, the top μ individuals with the best scores are selected as the parents for the next generation.

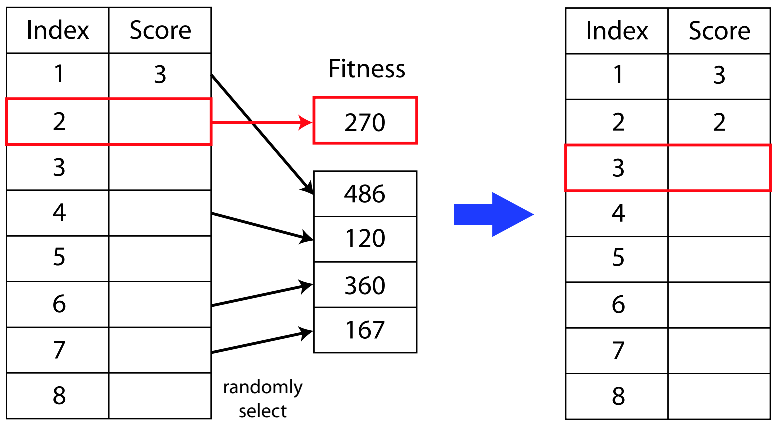

Figure 2.6: Illustration of the tournament selection scheme.

2.2.5 Dynamic Planning

Dynamic path planning is an iterative process. The planner of the vehicle continuously updates the path while the vehicle is moving in the field of operation. The steps in the dynamic path planning algorithm are described as follows:

1. Run the EC-based algorithm described above continuously to update the candidate path.
2. Send a portion *(trajectory)* of the current best path computed in step 1 to the vehicle controller once the vehicle reaches a new spawn point.
3. Update the estimates of the environment.
4. Update the paths in the population for a new vehicle location by removing a number of segments from the start of the paths and adding back segments to join the path to the new vehicle location.
5. Go to step 1

A diagram describing the concept of the dynamic path planning algorithm is shown in Figure 2.7. In dynamic path planning, the planning problem in each cycle is a nearby problem of that in the previous cycle. This approach attempts to preserve some information

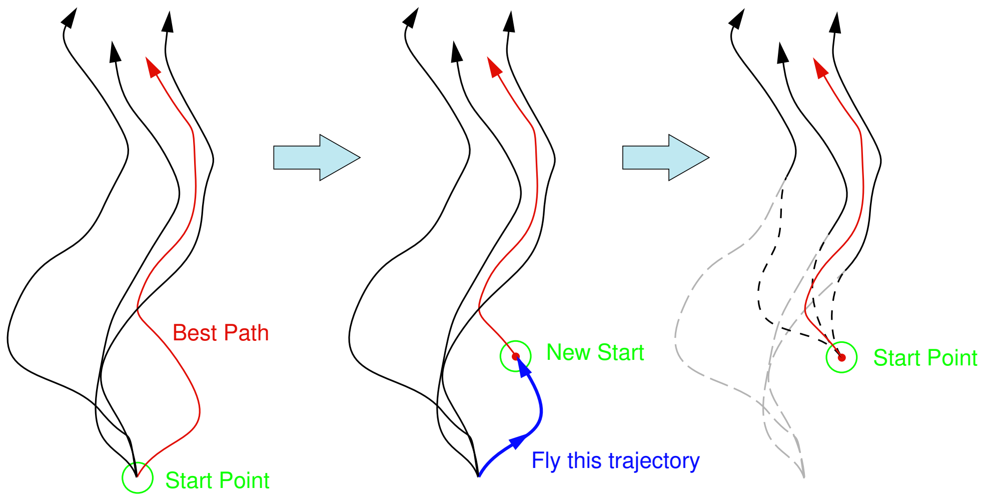

Figure 2.7: Concept of dynamic path planning algorithm which retains the knowledge gained from the previous planning cycle.

of the past solutions and uses it as the basis to compute new solutions even though the new problem is slightly different from the previous problem.

In highly dynamic environment situations, one would want the already-committed non-adapting sections of paths to be short to allow as much adaptability as possible. However, this comes with a trade-off on the time-available-to-plan requiring fast enough algorithms to compute new plans within a specified time limit. Figure 2.8 illustrates this concept. We define a *spawn point* as the point that specifies the starting point of a new trajectory which will be generated by the planner. In the figure, the current spawn point is located on the trajectory at time t_{s_p} which is the *execution time horizon* of the vehicle controller. This spawn point divides the committed section and the adapting section of the planned trajectory. The committed section, which will not be altered, is a portion of the path for the time period $t_k < t < t_{s_p}$. The adapting section of the trajectory is free to be further refined by the planner. The time t_N at the end point of this trajectory segment is the *planning time horizon*. A new committed trajectory is sent to the vehicle's controller for execution every time when the vehicle reaches a spawn point. The time difference between two adjacent spawn points ($\Delta T_s = (t_{s_p} - t_{s_{p-1}})$) specifies the maximum time-available-to-plan for the planner to update the adapting section. In this manner, the planner can account for new information about the environment that becomes available during execution.

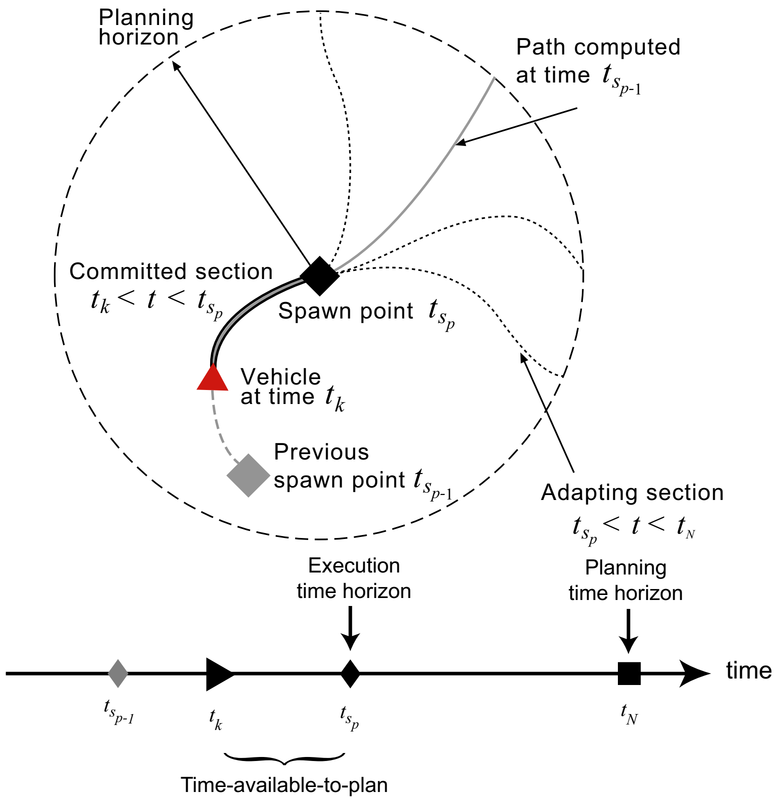

Figure 2.8: Dynamic Planning Concept. A vehicle (at time t_k) is moving along a trajectory previously computed at time t_{s_p-1}, shown as a gray line. The current spawn point at time t_{s_p} is shown as a black diamond.

2.2.6 Parallel Evolvers

The use of parallel evolution has a potential of addressing several issues for path planning: a) acceleration of the EC-based search for a feasible optimal solution, b) avoidance of local minima traps and c) integration of human operator inputs. Studies have shown that results computed using Evolutionary Algorithms can converge to global optimum solutions in simple optimization problems ([11], [13], [14], [15]). However, convergence properties rely heavily on the strength of the mutation mechanisms and selection scheme used in the algorithm to ensure global search space coverage. In addition, the optimization algorithm used in the planning system must converge ’fast enough’ relative to the speed of the environmental dynamics, which becomes essential for real-time applications. Addressing these issues, some researchers [16], [17] have shown that running several evolution processes in parallel can improve the speed of convergence and avoid local minima as well as tracking moving peaks in dynamic optimization problems [18].

Figure 2.9 illustrates the concept of the parallel evolution technique. In essence, multiple evolvers are coded to run independently based on the same fitness function. The key issue is to determine the best merging scheme. One does not wish to directly compete these parallel populations until comparable fitness levels are reached. Two possible strategies among many others for two parallel evolvers are: 1) Compare the two populations; if the secondary population is better, replace individuals in the principal population by those in the secondary population. 2) Replace individuals in the principal population by the top individuals from the two populations.

The use of parallel evolution may also allow the integration of inputs from various algorithms. Being independent, each evolver can be initialized in different ways. One possibility is the integration of human operator inputs, an important issue in automated algorithms. This scheme may allow human input to compete effectively with algorithm solutions. For example, if the operator has a high level of confidence that a particular path should be followed, then this information can be accommodated in the planner algorithm as hard constraints. If, however, it is not clear that the operator inputs are better than what the algorithm is deriving, the input paths are treated as initial conditions to a parallel evolver that is then merged at an appropriate time with the primary evolver. Other sources of initialization to each evolvers may include independent random seeds or deterministic method solutions (e.g. from A* [19] or D* [20] algorithms).

A preliminary application of the parallel evolution concept was used in the planning

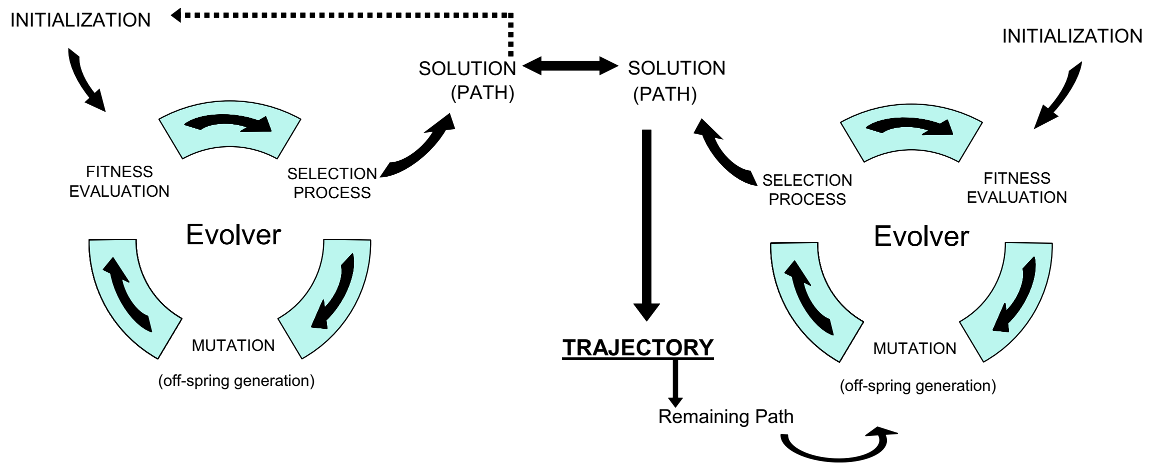

Figure 2.9: Parallel Evolution Concept.

examples presented in chapter 6. Two evolvers perform the same path search. They are randomly initialized and the best path results of each evolver are compared at every trajectory selection interval. The evolver containing the overall best path provides the portion of the path to commit and to continue evolving with its current population. The second evolver is re-initialized with random seeding to plan for the remaining distance, with a handicap number of generations.

Chapter 3

SPHERICAL GEOMETRIES BEHIND THE PATH PLANNER

Long range applications require consideration of the curvature of the Earth to account for factors such as:

- The distance of a degree of longitude varies with latitude,
- Heading azimuth (angle in horizontal plane relative to north) is not constant while moving directly from one point to another (i.e. in a 'straight' path),
- Shortest distance between two points on a sphere follows a great circle.

This chapter presents all the geometry-related computations and assumptions developed and implemented in the path planner. A spherical Earth has been assumed.

3.1 Coordinate System and Main Vectors

A 3-D Cartesian coordinate system was adopted for the development of the required computations. The main coordinate system is a right handed frame X-Y-Z with its origin at the Earth's center, the x axis passing through lat $0°$, long $0°$, and the z axis passing through the North Pole.

Five main parameters for path encoding are used throughout the planner: latitude θ, longitude ϕ, geometric altitude with respect to the Earth's surface Λ, heading azimuth ψ, and heading elevation γ (angle relative to a horizontal plane). Two main types of vectors are required for the development of the necessary computations: position vector $\vec{r}$ and azimuth vector $\vec{a}$ (Figure 3.1). They are constructed as unitary vectors based on θ, ϕ and ψ.

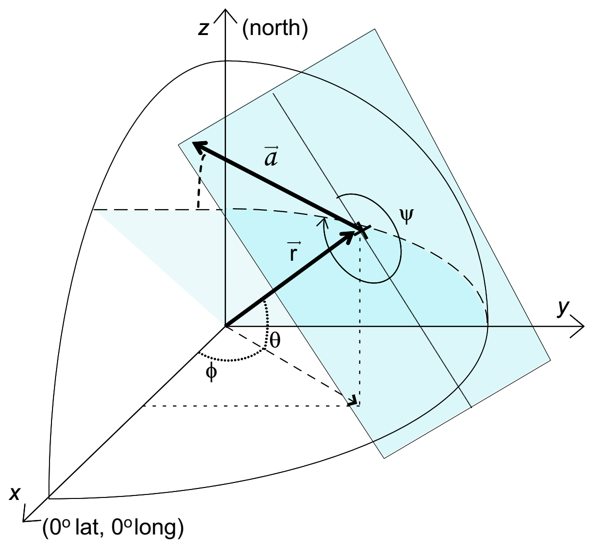

Figure 3.1: Position and Azimuth Vectors, in earth-centered frame $\mathcal{F}_{ec}$

The normalized position vector $\vec{r}$ is obtained by the known relationship:

$$\vec{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_{ec} = \begin{bmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ \sin\theta \end{bmatrix}_{ec} \tag{3.1}$$

In the case of the azimuth vector, it was necessary to find a relation between the cartesian components of the azimuth vector $\vec{a}$ and θ, ϕ and ψ. Figure 3.2 shows a closer look to a tangent plane similar to that appearing in Figure 3.1. It shows how the unit azimuth vector may be constructed as:

$$\vec{a} = \vec{a}_r + \vec{a}_p + \vec{a}_z \tag{3.2}$$

where $\vec{a}_z$ is in the direction of the z-axis, and $\vec{a}_r$ and $\vec{a}_p$ correspond to the *radial* and *perpendicular* vectors of the projection of the azimuth vector on the $x - y$ plane (Figure 3.3).

It is clear from Figure 3.2 that $\vec{a}_z$ corresponds to the projection of $\vec{a}$ on the z axis; thus, in the $\mathcal{F}_{ec}$ frame it corresponds to

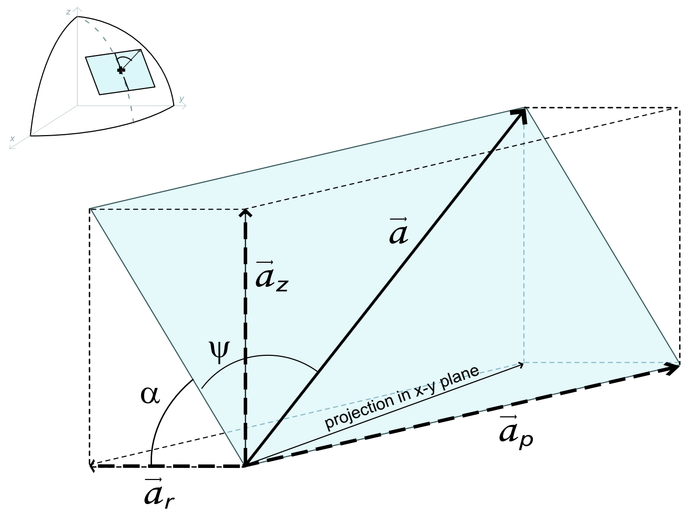

Figure 3.2: Azimuth Component Vectors.

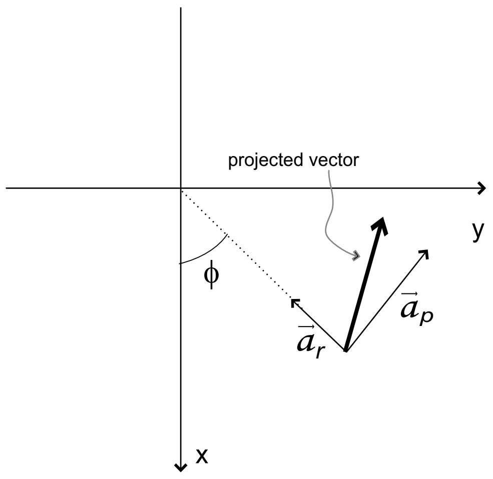

Figure 3.3: Azimuth Projection and Components on the $x - y$ Plane.

$$\vec{a}_z = a_z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{ec} \tag{3.3}$$

From Figure 3.3 we may relate vectors $\vec{a}_r$ and $\vec{a}_p$ to the $\mathcal{F}_{ec}$ frame by their x and y components:

$$\vec{a}_r = a_r \begin{bmatrix} -\cos\phi \\ -\sin\phi \\ 0 \end{bmatrix}_{ec} \tag{3.4}$$

$$\vec{a}_p = a_p \begin{bmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{bmatrix}_{ec} \tag{3.5}$$

Thus, substituting in 3.2 we have

$$\vec{a} = a_r \begin{bmatrix} -\cos\phi \\ -\sin\phi \\ 0 \end{bmatrix}_{ec} + a_p \begin{bmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{bmatrix}_{ec} + a_z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_{ec} \tag{3.6}$$

Referring back to Figure 3.2, and recalling that $\vec{a}$ is a unit vector, it is clear that a_r, a_p, and a_z are given by

$$a_r = \cos\psi\cos\alpha \tag{3.7}$$

$$a_p = \sin\psi \tag{3.8}$$

$$a_z = \sin\alpha\cos\psi \tag{3.9}$$

Substituting them in 3.6, we then have

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{ec} = \begin{bmatrix} -\cos\phi\cos\psi\cos\alpha - \sin\phi\sin\psi \\ -\sin\phi\cos\psi\cos\alpha + \cos\phi\sin\psi \\ \sin\alpha\cos\psi \end{bmatrix}_{ec} \tag{3.10}$$

And since $\alpha = 90^\circ - \theta$, the relation between the azimuth vector $\vec{a}$ and the parameters θ, ϕ and ψ in the $\mathcal{F}_{ec}$ frame is finally given by:

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{ec} = \begin{bmatrix} -\cos\phi\cos\psi\sin\theta - \sin\phi\sin\psi \\ -\sin\phi\cos\psi\sin\theta + \cos\phi\sin\psi \\ \cos\theta\cos\psi \end{bmatrix}_{ec} \tag{3.11}$$

Vectors $\vec{r}$ and $\vec{a}$ define vector $\vec{n}$ via the cross product $\vec{n} = \vec{r} \times \vec{a}$, which defines the great circle plane formed by the previous two vectors.

$$\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}_{ec} = \begin{bmatrix} \sin\phi\cos\psi - \sin\theta\cos\phi\sin\psi \\ -\cos\phi\cos\psi - \sin\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi \end{bmatrix}_{ec} \tag{3.12}$$

These three normalized vectors are of basic use for segment calculations.

3.2 Segment Construction

Two main types of geometries for the encoding of segments of a path have been implemented in the path planner: straight segments following a great circle path (which is a *projected* great circle path when changing altitude), and curve segments following a small circle path (Figure 3.4).

3.2.1 Straight Segments (Climbs, Descents and Lines)

A straight path segment is defined by the following parameters:

- Initial position $\theta_i, \phi_i, \Lambda_i$,
- Length S,
- Initial heading, ψ, γ, where $\gamma = 0$ for constant altitude segments referred to as *lines*
- Final position, $\theta_f, \phi_f, \Lambda_f$, which is either computed based on the parameters above, or defined by a pre-defined position to join to.

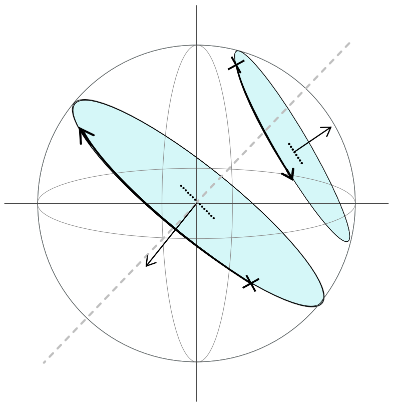

Figure 3.4: Great Circle and Small Circle Paths.

Figure 3.5 presents the vector structure to generate these type of segments. The method computes the final position after traveling a distance S from the initial position and heading. The assumption is that even for climbs and descents, the path follows a circular geometry with the same radius as the great circle that would be flown with no altitude change, given by the radius of the earth + initial altitude ($R_{E+\Lambda_i}$). The final position vector $\vec{p}$, normalized by $R_{E+\Lambda_i}$, is obtained by:

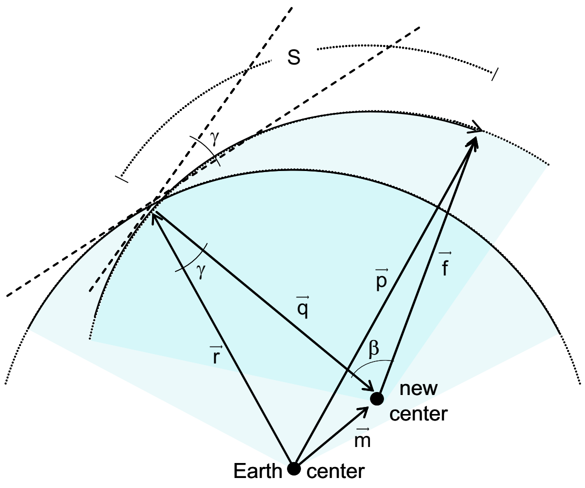

Figure 3.5: Great Circle Projection Method for Climb Segments (Normal View of Great Circle Plane).

$$\vec{p} = \vec{m} + \vec{f} \tag{3.13}$$

where $\vec{m}$ is the vector defining the translation of the Earth center to the *new center* for the projected great circle climb and $\vec{f}$ is the final position vector from the *new center* location. These two vectors are constructed with the aid of the two frames R-A-N, based on $\vec{r}$, $\vec{a}$, $\vec{n}$; and N-(-Q)-E, a right handed frame based on $\vec{n}$, $-\vec{q}$, $\vec{e}$ (see Figure 3.6):

$$\vec{m} = \vec{q} + \vec{r} \tag{3.14}$$

and

$$\vec{f} = \begin{bmatrix} 0 & \cos\beta & \sin\beta \end{bmatrix} \begin{bmatrix} \vec{n} \\ -\vec{q} \\ \vec{e} \end{bmatrix} \tag{3.15}$$

$$\beta = \frac{S}{R_{E+\Lambda_i}} \tag{3.16}$$

where $\vec{q}$ and $\vec{e}$ are defined by

$$\vec{q} = \begin{bmatrix} -\cos\gamma & \sin\gamma & 0 \end{bmatrix} \begin{bmatrix} \vec{r} \\ \vec{a} \\ \vec{n} \end{bmatrix} \tag{3.17}$$

$$\vec{e} = \vec{n} \times \vec{-q} \tag{3.18}$$

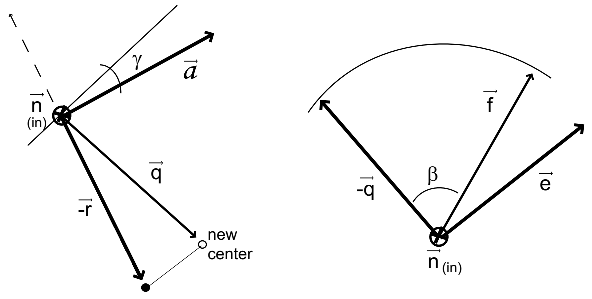

Figure 3.6: Local Auxiliary Frames for Straight Segment Computation.

Recall from Section 3.1 that $\vec{r}$, $\vec{a}$, $\vec{n}$ form an R-A-N system with transformation given by,

$$\begin{bmatrix} \vec{r} \\ \vec{a} \\ \vec{n} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & \sin\theta \\ -\cos\phi\cos\psi\sin\theta - \sin\phi\sin\psi & -\sin\phi\cos\psi\sin\theta + \cos\phi\sin\psi & \cos\theta\cos\psi \\ \sin\phi\cos\psi - \sin\theta\cos\phi\sin\psi & -\cos\phi\cos\psi - \sin\theta\sin\phi\sin\psi & \cos\theta\sin\psi \end{bmatrix} \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix} \tag{3.19}$$

From the ending position vector $\vec{p}$ we directly obtain the final latitude θ_f, longitude ϕ_f and altitude Λ_f (note that $\Lambda_f = \Lambda_i + R_{E+\Lambda_i}(|\vec{p}| - 1)$, as $\vec{p}$ was normalized by $R_{E+\Lambda_i}$). The final elevation angle γ_f is simply given by the angle between $\vec{f}$ and $\vec{p}$. The final heading azimuth is computed as

$$\psi_f = \psi_{f\to i} + 180^\circ \tag{3.20}$$

where $\psi_{f\to i}$ is the azimuth direction from the final position to the initial position, computed with the method presented in subsection 3.3.2.

Strictly, for this geometry to make sense (i.e. for climbs) we are subject to a geometric restriction such that

$$S_{max} = R_{E+\Lambda_i}\frac{\pi + \gamma}{2} \tag{3.21}$$

to avoid ”climbing back down to Earth” (see Figure 3.7). Typically the planner is restricted to generate segments of a much smaller scale for the path planning to be acceptable.

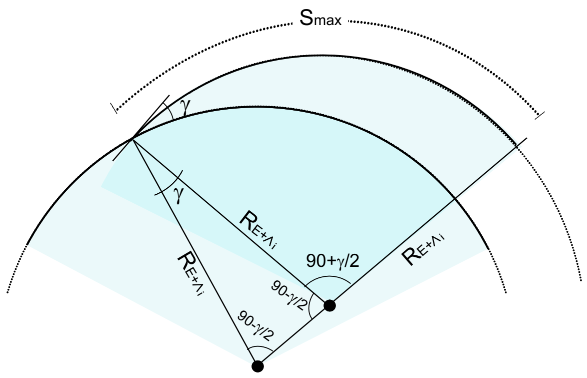

Figure 3.7: Geometric Limitation for Maximum Length of Straight Segments.

3.2.2 Curved Segments (Turns)

A curved path segment is defined by the following parameters:

- Initial position, $\theta_i, \phi_i, \Lambda_i$
- Length (flying distance) S,
- Initial heading, $\psi, \gamma = 0$, (no altitude change commanded for curved segments)
- Radius R_s,
- Direction of turn, either clockwise (CW) or counter-clockwise (CCW),
- Final position, $\theta_f, \phi_f, \Lambda_f$, which is computed based on the parameters above.

Figure 3.8 presents the geometric construction of this segment type. First, it is necessary to locate the center of turn, for which the method to construct straight segments is used. Based on the current heading azimuth, a *fake azimuth* is computed by adding $\pm 90°$ to the current azimuth, in order to point in the direction where the center of turn must be located (depending on if the turn is to be clockwise or counterclockwise). This being done, a fake azimuth vector $\vec{fa}$ may be constructed. Vectors $\vec{r}$ and $\vec{fa}$ form a set just as the one required to find the end position when moving on a great circle, with $\gamma = 0$. In this case, the desired radius of turn R_s (which is a *curved* radius following the sphere's shape) determines where the center is located, and its position vector $\vec{c}$ is obtained (see subsection 3.2.1). Other required vectors (Figure 3.8) are

$$\vec{rc} = Proy_c \vec{r} \tag{3.22}$$

and

$$\vec{t} = \vec{r} - \vec{rc} \tag{3.23}$$

The set $\vec{t}$, $\vec{n}$, $\vec{c}$ form a left-handed frame T-N-C, the curve segment of interest being constructed lying on the plane T-N. The final position vector $\vec{p}$, normalized by $R_{E+\Lambda_i}$ is given by

$$\vec{p} = \vec{t_f} + \vec{c} \tag{3.24}$$

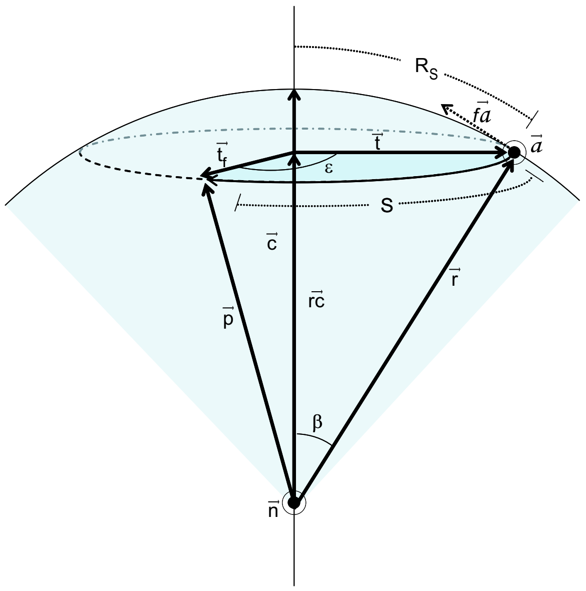

Figure 3.8: Curve Segment Geometry for a Clockwise Turn.

where

$$\vec{t_f} = \begin{bmatrix} -\cos\varepsilon & \sin\varepsilon & 0 \end{bmatrix} \begin{bmatrix} \vec{t} \\ \vec{n} \\ \vec{c} \end{bmatrix} \tag{3.25}$$

$$\varepsilon = \frac{S}{||\vec{t}||} \tag{3.26}$$

The final azimuth heading is obtained by

$$\psi_f = \psi_{f\to c} + \begin{cases} 180^\circ & \text{if turn is CCW} \\ -180^\circ & \text{if turn is CW} \end{cases} \tag{3.27}$$

where $\psi_{f\to i}$ is the azimuth direction from the final position to the center point, computed with the method presented in subsection 3.3.2.

3.3 Supporting Calculations

Several other computations are required to support the construction of paths, as in many situations (particularly during the mutation and joining mechanisms) the planner does not only need to generate a segment based on initial location and heading, but also between two predefined locations or must be constrained to meet certain parameters.

3.3.1 Distance and Range between Two Points

We define distance as the arclength between them subject to the geometric shape used for straight segments. Range is defined as the arclength between them not considering altitude change, and based on the altitude of the ’initial’ point, also subject to the circular shape.

Figure 3.9 shows the geometry structure for the distance calculation. Clearly, the distance between the two points is simply given by

$$D = \beta \; R_{E+\Lambda_1} \tag{3.28}$$

where the angle β is obtained by

$$\beta = 2 \; \sin^{-1} \frac{l}{2 \; R_{E+\Lambda_1}} \tag{3.29}$$

The distance l in equation 3.29 is computed using the cosine rule

$$l = \sqrt{R^2_{E+\Lambda_1} + R^2_{E+\Lambda_2} - 2 \; R^2_{E+\Lambda_1} \; R^2_{E+\Lambda_2} \; \cos\alpha} \tag{3.30}$$

where α (the angle between the two position vectors) is computed via the dot product, which in terms of θ and ϕ results in:

$$\alpha = \cos^{-1}(\cos\theta_1 \cos\theta_2 \cos(\phi_1 - \phi_2) + \sin\theta_1 \sin\theta_2) \tag{3.31}$$

The range, as previously defined, is simply computed as

$$Range = \alpha \; R_{E+\Lambda_1} \tag{3.32}$$

Strictly speaking, this method is valid only for $l < 2R_{E+\Lambda_1}$; otherwise point 2 is not reachable according to the circular shape of the straight segment definition.

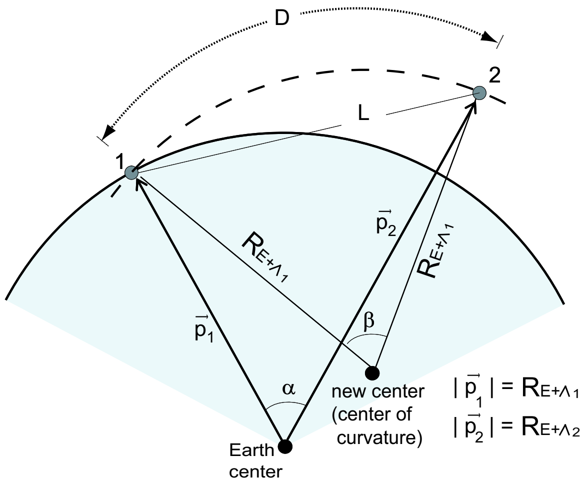

Figure 3.9: Distance Between Two Points.

3.3.2 Shortest Distance Azimuth Direction

Knowing the heading azimuth required to go straight from one point to another is an important calculation. Figure 3.10 presents the geometry involved. The method computes the azimuth angle that corresponds to the shortest distance to travel from point 1 to point 2.

Angle Ω is the inclination angle of the great circle plane with respect to the z axis obtained by

$$\cos\Omega = n_z/|\vec{n}| \tag{3.33}$$

The figure on the right shows a closer look to the spherical triangle formed. Being a right angled triangle, Napier's rules for spherical triangles may be applied (see Appendix A). Thus, we can directly relate angles Ω, θ_1 and ψ by

$$\sin\overline{\Omega} = \cos\theta_1 \cos\overline{\psi} \tag{3.34}$$

which is equivalent to

$$\cos\Omega = \cos\theta_1 \sin\psi \tag{3.35}$$

Solving for ψ we have

$$\psi = \sin^{-1}\left(\frac{\cos\Omega}{\cos\theta_1}\right) \tag{3.36}$$

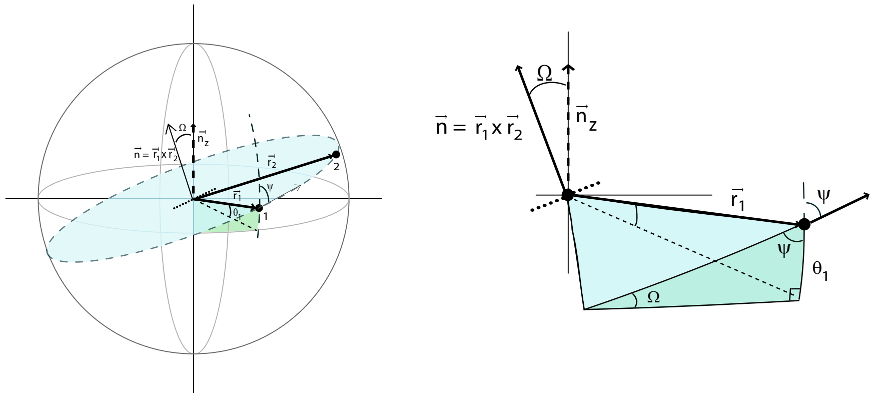

Figure 3.10: Heading Azimuth from Point 1 to Point 2.

By using 3.36, the returned angle ψ is in the range $[-\pi/2, \pi/2]$, corresponding to northerly headings. For this reason, a correction may be required in order to account for southerly headings. This is achieved by computing the tangent vector to the great circle $\vec{tan} = \vec{n} \times \vec{r_1}$ and computing it's z component. If $tan_z < 0$, the actual azimuth must be a southerly azimuth, and thus the correction $\psi_{real} = \pi - \psi$ is necessary, where ψ is the angle computed directly by equation 3.36.

As may be noticed, this method is only concerned with the azimuth angle for the direction from one point to another. If required, the elevation angle is computed through the method presented in 3.3.4.

3.3.3 Length based on Initial Elevation Angle and Altitude Change

In many cases it is necessary to compute the required segment's length, S, based on a desired altitude change $\Delta\Lambda$ and a specified initial elevation angle, γ_i (i.e. determined by maximum or minimum achievable climb angles by the vehicle). Again, computations were restricted by the circular geometry used for straight segments. Figure 3.11 presents the main geometric parameters involved.

The segment length is

$$S = \beta \; R_{E+\Lambda_i} \tag{3.37}$$

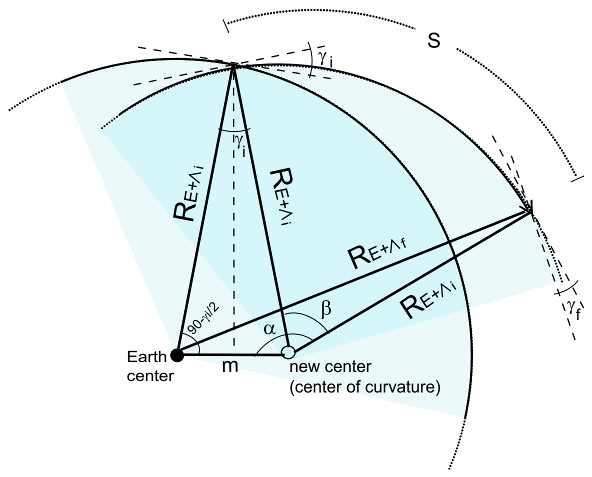

Figure 3.11: Geometry for Length Calculation based on Elevation Angle and Altitude Change.

From the figure, we note that

$$\beta = \alpha - (\frac{\pi}{2} - \frac{\gamma_i}{2}) \tag{3.38}$$

Also, from the cosine rule, we have

$$R^2_{E+\Lambda_f} = m^2 + R^2_{E+\Lambda_i} - 2(m)(R_{E+\Lambda_i}) \cos\alpha \tag{3.39}$$

Solving for α in equation 3.39 we have

$$\alpha = \cos^{-1} \frac{m^2 + R^2_{E+\Lambda_i} - R^2_{E+\Lambda_f}}{2(m)(R_{E+\Lambda_i})} \tag{3.40}$$

This can be substituted in equation 3.38, which itself is substituted into 3.37, giving the following equation for the segment length based on initial elevation angle and altitude change:

$$S = \left| \cos^{-1} \frac{m^2 + R^2_{E+\Lambda_i} - R^2_{E+\Lambda_f}}{2(m)(R_{E+\Lambda_i})} - \frac{\pi - \gamma_i}{2} \right| R_{E+\Lambda_i} \tag{3.41}$$

(the absolute value required in case $\Delta\Lambda < 0$), where

$$\begin{aligned} R_{E+\Lambda_f} &= \Delta\Lambda + R_{E+\Lambda_i} \\ m &= 2\, R_{E+\Lambda_i} \sin(\gamma_i/2) \end{aligned}$$

3.3.4 Initial Elevation Angle required between Two Points

By specifying the required initial and final locations for the segment, and computing the distance between them, one can calculate the required initial elevation γ_i by solving for it from equation 3.41. In this case, a simple solution is difficult to obtain; but considering that γ will normally be a small angle, we use the approximations

$$\sin(\gamma/2) \approx \gamma/2 \quad \cos(\gamma/2) \approx 1$$

Thus, we end up with a quadratic relation, from which we find

$$\begin{aligned}\gamma_i \;\approx\; & \frac{\sin(\frac{S}{R_{E+\Lambda_i}})}{\cos(\frac{S}{R_{E+\Lambda_i}})-1} - \\ & \frac{\sqrt{(R^2_{E+\Lambda_i}\ \sin^2(\frac{S}{R_{E+\Lambda_i}}) - (\cos(\frac{S}{R_{E+\Lambda_i}})-1)(R^2_{E+\Lambda_f} - R^2_{E+\Lambda_i}))}}{R_{E+\Lambda_i}(\cos(\frac{S}{R_{E+\Lambda_i}})-1)}\end{aligned} \tag{3.42}$$

3.3.5 Relating Final and Initial Elevation Angle

As part of the attempt to make a segment flyable by the vehicle capabilities, it is sometimes required to compute the initial elevation angle required to achieve certain final elevation angle (or viceversa). This relation may be obtained referring back to figure 3.11, where we notice that the triangle $R_{E+\Lambda_i} - m - R_{E+\Lambda_i}$ is isosceles, so

$$\sin(\gamma_i/2) = \frac{m/2}{R_{E+\Lambda_i}} \tag{3.43}$$

or

$$\gamma_i = 2\sin^{-1}\frac{m}{2\ R_{E+\Lambda_i}} \tag{3.44}$$

where, by the cosine rule, we see from the figure that

$$m = \sqrt{R^2_{E+\Lambda_f} + R^2_{E+\Lambda_i} - 2\ R^2_{E+\Lambda_f}\ R^2_{E+\Lambda_i}\cos\gamma_f} \tag{3.45}$$

3.4 Bounding Boxes

Path planning may involve interaction with other objects, which are referred to as *sites*. These may be obstacles (sites to avoid) or targets (sites to approach) depending on the objectives of the flight. They may be static or moving sites, with paths associated with them. The planner must then be able to consider intersection of the vehicle's planned path with those of the sites in the search space. Methods to determine probabilities of intersection in uncertain environments are explained by Rathbun, et.al. [10] and [21], and are not addressed here. We instead focus on algorithms developed to accelerate intersection computations that involve spherical geometry based methods for path planning in a spherical Earth representation.

Instead of checking intersection at every point in a path (which requires a much smaller discretization), intersection checks are made first on a *macro* level. Each segment in a path, and each site in the world and their estimated trajectory, is enclosed in a 3-D space by a *Bounding Box*. If two Bounding Boxes from different paths do not overlap, then no intersection is possible between those segments in the paths and the check is done for some other Bounding Boxes of interest. Only when Bounding Boxes overlap an intersection is possible and checks on a much smaller discretization level are made.

Bounding Boxes are intended to be simple shaped, and are determined by six basic parameters:

- maximum and minimum latitudes ($\theta_{\Xi}^{mx}, \theta_{\Xi}^{mn}$),
- east and west longitudes ($\phi_{\Xi}^{E}, \phi_{\Xi}^{W}$),
- maximum and minimum altitudes ($\Lambda_{\Xi}^{mx}, \Lambda_{\Xi}^{mn}$).

The following subsections present the calculation of Bounding Boxes in a spherical Earth model, as well as the overlapping check algorithm.

3.4.1 Straight Segments Bounding Box

Let P be the spherical Earth's perimeter. Bounding latitudes for a Straight Segment are given by

$$(\theta_\Xi^{mx}, \theta_\Xi^{mn}) = \begin{cases} (\Omega, \min\{\theta_i, \theta_f\}) & : \ \Psi^N \rightarrow \Psi^S \text{ and } S < P \\ (\max\{\theta_i, \theta_f\}, -\Omega) & : \ \Psi^S \rightarrow \Psi^N \text{ and } S < P \\ (\max\{\theta_i, \theta_f\}, \min\{\theta_i, \theta_f\}) & : \ \Psi^k \rightarrow \Psi^k \text{ and } S < P/2 \\ (\Omega, -\Omega) & : \ \begin{array}{l}\Psi^k \rightarrow \Psi^k \text{ and } S > P/2 \\ \text{or } S > P\end{array} \end{cases} \tag{3.46}$$

$$k = \{N, S\}$$

where Ω is the great circle inclination angle in equation 3.33 (from Figure 3.10 it is clear that the maximum and minimum possible latitudes for a straight segments are $\pm\Omega$). Ψ^N and Ψ^S represent *Northerly* and *Southerly* heading azimuths respectively, and $\Psi^j \rightarrow \Psi^j$ indicates a *"j"-erly* heading azimuth change from start to end of the segment.

Bounding longitudes are given by

$$(\phi_\Xi^W, \phi_\Xi^E) = \begin{cases} (\phi_i, \phi_f) & : \ \Psi^E \text{ and } S < P \\ (\phi_f, \phi_i) & : \ \Psi^W \text{ and } S < P \\ (0°, \rightarrow 360°) & : \ S > P \end{cases} \tag{3.47}$$

where Ψ^E and Ψ^W represent *Easterly* and *Westerly* heading azimuths respectively.

Finally, bounding altitudes are simply given by

$$(\Lambda_\Xi^{mx}, \Lambda_\Xi^{mn}) = (\max\{\Lambda_i, \Lambda_f\}, \min\{\Lambda_i, \Lambda_f\}) \tag{3.48}$$

(this assumes the S_{max} restriction in the case of climb segments).

Figure 3.12 shows two examples of resulting bounding boxes (constant altitude and climbing segments, respectively). They correspond to a case where $\Psi^{NE} \rightarrow \Psi^{SE}$ (for $S < P$).

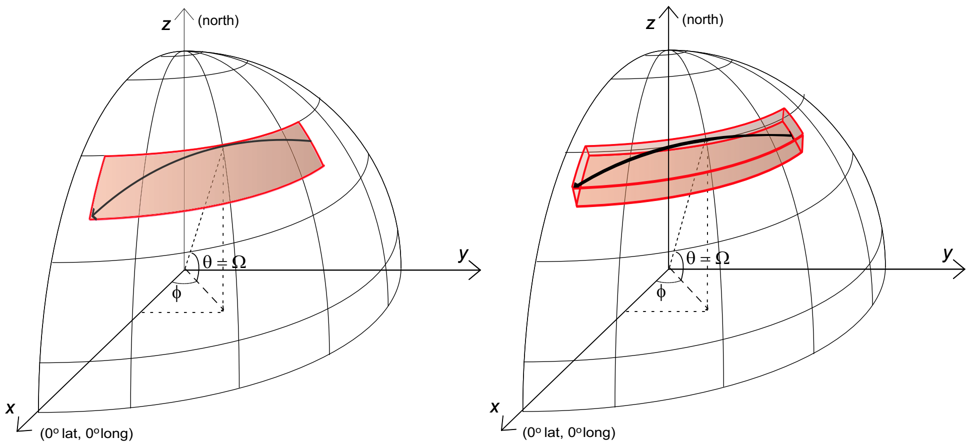

Figure 3.12: Bounding Box for Straight Segments.

3.4.2 *Curved Segments Bounding Box*

Let P be the perimeter of the small circle that defines the curved segment. Similar to equation 3.46, bounding latitudes are given by

$$(\theta_{\Xi}^{mx}, \theta_{\Xi}^{mn}) = \begin{cases} (\Theta^M, \min\{\theta_i, \theta_f\}) & : \ \Psi^N \rightarrow \Psi^S \text{ and } S < P \\ (\max\{\theta_i, \theta_f\}, \Theta^m) & : \ \Psi^S \rightarrow \Psi^N \text{ and } S < P \\ (\max\{\theta_i, \theta_f\}, \min\{\theta_i, \theta_f\}) & : \ \Psi^k \rightarrow \Psi^k \text{ and } S < P/2 \\ (\Theta^M, \Theta^m) & : \ \begin{array}{l}\Psi^k \rightarrow \Psi^k \text{ and } S > P/2 \\ \text{or } S > P\end{array} \end{cases} \tag{3.49}$$

$$k = \{N, S\}$$

where Θ^M and Θ^m are the tangent latitudes to the small circle, corresponding to maximum and minimum possible latitudes for the curve (Figure 3.13). They are obtained by

$$\Theta^M = \theta_c + \beta \tag{3.50}$$

$$\Theta^m = \theta_c - \beta \tag{3.51}$$

where θ_c is the small circle center's latitude, and $\beta = R_s/R_{E+\Lambda}$. When the center of

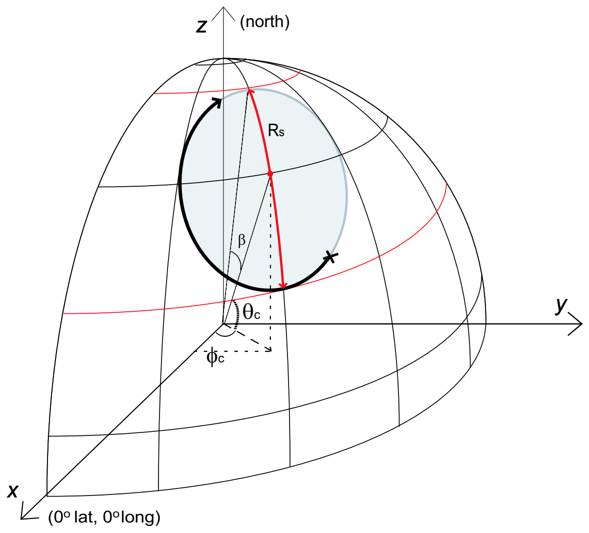

Figure 3.13: Maximum and Minimum Possible Latitudes for Curved Segment's Bounding Box.

the curve is close to either pole such that $|90° - \theta_c| < \beta$, then

$$\Theta^M = 180° - (\theta_c + \beta) \quad \text{if } \theta_c > 0 \tag{3.52}$$

$$\Theta^m = -180° + (\theta_c + \beta) \quad \text{if } \theta_c < 0 \tag{3.53}$$

Bounding longitudes, for a small circle far from the poles such that $|90° - \theta_c| > \beta$, are given by

$$(\phi^W_\Xi, \phi^E_\Xi) = \begin{cases} (\Phi^W, \varphi\{\phi_i, \phi_f\}) & : \Psi^E \rightarrow \Psi^W \text{ and } S < P \\ (\varphi\{\phi_i, \phi_f\}, \Phi^E) & : \Psi^W \rightarrow \Psi^E \text{ and } S < P \\ (\varphi\{\phi_i, \phi_f\}, \varphi\{\phi_i, \phi_f\}) & : \Psi^k \rightarrow \Psi^k \text{ and } S < C \\ (\Phi^W, \Phi^E) & : \begin{array}{l} \Psi^k \rightarrow \Psi^k \text{ and } S > C \\ \text{or } S > P \end{array} \end{cases} \tag{3.54}$$

$$k = \{E, W\}$$

where

$$\varphi\{\phi_i, \phi_f\} = \begin{cases} \phi_i & : 0 < \psi_{i \rightarrow f} < \pi \\ \phi_f & : \pi < \psi_{i \rightarrow f} < 2\pi \end{cases}$$

and

$$C = \begin{cases} 2R_s A^{\tau}(\pi - A)^{1-\tau} & : \quad k = E \\ 2R_s A^{1-\tau}(\pi - A)^{\tau} & : \quad k = W \end{cases} \qquad \tau = \begin{cases} 1 & : \quad \text{CW} \\ 0 & : \quad \text{CCW} \end{cases}$$

Φ^W and Φ^E are the maximum possible west and east longitudes, which are tangent to the small circle. The tangent points occur at angles A and $-A$ as shown in Figure 3.14. Angle A may be computed using one of Napier's Rules for spherical triangles (Appendix A), applied to the red triangle in the figure:

$$\sin \overline{A} = \tan b \tan(\overline{90^\circ - \theta_c}) \tag{3.55}$$

which is equivalent to

$$\cos A = \tan b \tan \theta_c \tag{3.56}$$

so

$$A = \cos^{-1}(\tan b \tan \theta_c) \tag{3.57}$$

With these angles A and $-A$, the longitudes that are tangent to the small circle can be found by using the method in section 3.2.1, by 'moving' from the small circle center a distance R_s with a heading azimuth $\psi = A$ to find the east longitude Φ^E, and with $\psi = -A$ to find the west longitude Φ^W.

When the small circle's center is close to a pole, such that $|90^\circ - \theta_c| < \beta$, there are no tangent longitudes, and A does not exist. Then the bounding longitudes are given by equation 3.47.

For constant altitude curved segments as used in the planner, altitude bounds are simply $(\Lambda_{\Xi}^{mx}, \Lambda_{\Xi}^{mn}) = (\Lambda, \Lambda)$.

Figure 3.15 (left) shows the resulting bounding box for the curve segment of Figures 3.13 and 3.14, which corresponds to a case where $\Psi^{SW} \rightarrow \Psi^{NE}$ and $S < P$. The figure on the right shows a case where the turn was close to a pole.

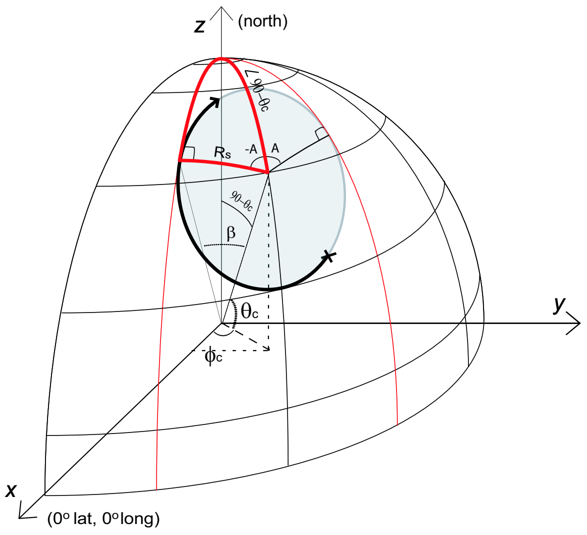

Figure 3.14: Maximum Possible West and East Longitudes for Curved Segment's Bounding Box.

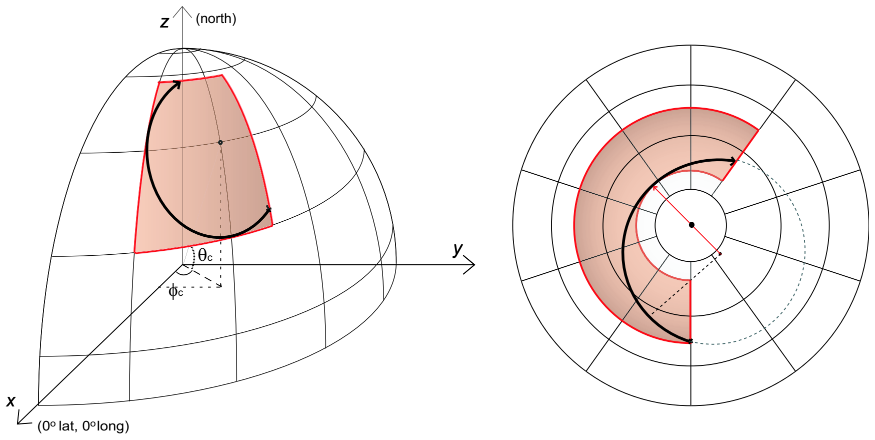

Figure 3.15: Bounding Box for Curved Segment.*Left*: far from the pole. *Right*: close to the pole.

3.4.3 Site Bounding Box

Sites are in general treated as uncertain objects of which only some information is available at certain times. Thus, their location at other times must be estimated based on uncertainty model methods [21]. At every time t_k a site location may be estimated with some uncertainty radius $\sigma(t_k)$ based on available information such as heading, speed, intentions, etc., at some earlier time t_0. And by adding the known *size* of the site μ (i.e. due to physical size, payload range, etc.), a site is considered having a radius $\rho(t_k) = \sigma(t_k) + \mu$. Certainly, if no updated information of the site's location is available, the uncertainty radius $\sigma(t_k)$, and thus $\rho(t_k)$, should increase as time advances (Figure 3.16).

The computation of a site's bounding box over some time range $[t_i, t_f]$ must then consider these factors (notice that bounding boxes for segments are effectively the bounding box of the vehicle also at some time intervals). As sites are modeled as a center location with a radius ρ, the site at a given time t_k has effectively the shape of a small circle (a full perimeter curve segment). And if the site's estimated trajectory (of length S) is considered as a straight path based on initially known heading and speed, then computation of the bounding box for a site in the time range $[t_i, t_f]$ is a combination of the methods for bounding boxes for straight segments and curved segments.

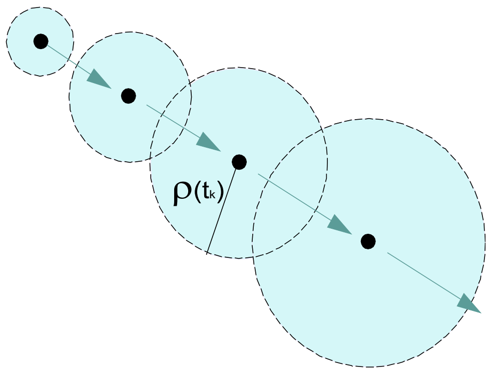

Figure 3.16: Site's radius ρ increasing over time due to uncertainty.

For cases in which the site location in the range of interest is far from the poles (i.e. $|90^\circ - \theta_{c,k}| > \rho_k / R_{E+\Lambda}$ for all t_k in the range $[t_i, t_f]$), the bounding box is computed as follows:

Let P be the spherical Earth's perimeter. West and East longitudes are given by

$$(\phi_\Xi^W, \phi_\Xi^E) = \begin{cases} (\Phi_{initial}^W, \Phi_{final}^E) & : \quad \Psi^E \rightarrow \Psi^W \text{ and } S < P \\ (\Phi_{final}^W, \Phi_{initial}^E) & : \quad \Psi^W \rightarrow \Psi^E \text{ and } S < P \\ (0^\circ, \rightarrow 360^\circ) & : \quad S > P \end{cases} \tag{3.58}$$

And maximum and minimum latitudes are given by

$$(\theta_\Xi^{mx}, \theta_\Xi^{mn}) = \begin{cases} (\max\{\Omega_r, \mu_M\}, \mu_m) & : \quad \Psi^N \rightarrow \Psi^S \\ (\mu_M, \min\{-\Omega_r, \mu_m) & : \quad \Psi^S \rightarrow \Psi^N \\ (\mu_M, \mu_m\}) & : \quad \Psi^k \rightarrow \Psi^k \text{ and } S < P/2 \\ (\max\{\Omega_r, \mu_M\}, \min\{-\Omega_r, \mu_m\}) & : \quad \Psi^k \rightarrow \Psi^k \text{ and } S > P/2 \end{cases} \tag{3.59}$$

where

$$\mu_M = \max(\Theta_{initial}^M, \Theta_{final}^M)$$

$$\mu_m = \min(\Theta_{initial}^m, \Theta_{final}^m)$$

and

$$\Omega_r = \Omega + \rho_\Omega$$

(ρ_Ω meaning the site's computed radius when hitting latitude Ω).

When the site location gets close to a pole in some time during the range of interest (i.e. $|90^\circ - \theta_{c,k}| < \rho_k / R_{E+\Lambda}$ for any t_k in the range $[t_i, t_f]$), then the bounding box would be defined within all longitudes

$$(\phi_\Xi^W, \phi_\Xi^E) = (0^\circ, \rightarrow 360^\circ) \tag{3.60}$$

and latitude bounds are given by

$$(\theta_{\Xi}^{mx}, \theta_{\Xi}^{mn}) = \begin{cases} (90^\circ, \mu_m) & : \text{pole involved is North Pole} \\ (\mu_M, -90^\circ) & : \text{pole involved is South Pole} \end{cases} \tag{3.61}$$

Finally altitude bounds would simply be given by

$$(\Lambda_{\Xi}^{mx}, \Lambda_{\Xi}^{mn}) = (\max(\Lambda_{\Xi initial}^{mx}, \Lambda_{\Xi final}^{mx}), \min(\Lambda_{\Xi initial}^{mn}, \Lambda_{\Xi final}^{mn})) \tag{3.62}$$

Figure 3.17 presents a case where the site is far from the pole, as well as a case where it is close to a pole. In both cases, $\Lambda_{\Xi}^{mx} = \Lambda_{\Xi}^{mn}$ was assumed.

3.4.4 Bounding Box Overlapping Check

Having defined the bounding box with the six basic parameters mentioned previously, box overlapping is quickly checked. Altitude and latitude bound intersection criteria are straight forward, but special consideration is required for criteria involving longitudes.

Let box 1 be the bounding box with the minimum left longitude value. Then, no intersection between paths can occur if at least one of the following criteria is true:

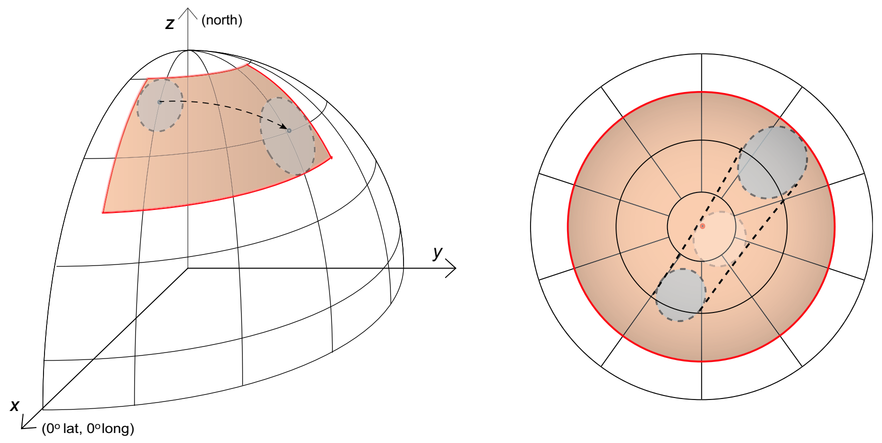

Figure 3.17: Bounding Box for a Site. *Left*: far from the pole. *Right*: close to the pole.

- $\Lambda_{\Xi1}^{mx} < \Lambda_{\Xi2}^{mn}$
- $\Lambda_{\Xi1}^{mn} > \Lambda_{\Xi2}^{mx}$
- $\theta_{\Xi1}^{mx} < \theta_{\Xi2}^{mn}$
- $\theta_{\Xi1}^{mn} > \theta_{\Xi2}^{mx}$

And, only if ($\phi_{\Xi1}^{W} > \phi_{\Xi1}^{E}$ and $\phi_{\Xi2}^{W} > \phi_{\Xi2}^{E}$) is false:

- $\{\phi_{\Xi2}^{W} > \phi_{\Xi1}^{E} \quad | \quad \phi_{\Xi1}^{W} < \phi_{\Xi1}^{E}$ and $\phi_{\Xi2}^{W} < \phi_{\Xi2}^{E}\}$
- $\{\phi_{\Xi1}^{W} > \phi_{\Xi2}^{E}$ and $\phi_{\Xi2}^{W} > \phi_{\Xi1}^{E} \quad | \quad \phi_{\Xi1}^{W} > \phi_{\Xi1}^{E}$ or $\phi_{\Xi2}^{W} > \phi_{\Xi2}^{E}\}$

Chapter 4

VEHICLE PERFORMANCE

This chapter presents how Vehicle Performance information is integrated into the planner to assure that the airplane is capable of traversing a path, as well as considering an appropriate fuel consumption model for the vehicle of interest. The planner does not deal with vehicle dynamics, and it assumes the vehicle maintains course.

4.1 Performance Constraints to the Path Planner

Every segment of a path generated by the planner is constrained by vehicle performance characteristics:

- Descent rates,
- Climb rates,
- Descent angles,
- Climb angles,
- Speeds (descents, climbs, cruise),
- Achievable turn radius,
- Altitudes for flight.

All these characteristics depend heavily on altitude and vehicle weight. Therefore, it has been necessary to constrain segment construction appropriately to assure that every path would be flyable.

Changes in any segment of a path during mutation processes has an effect on the subsequent unmodified segments. This effect may be strictly *geometric* (i.e. increasing the total length of a path by adding new segments at some middle portion of it) as well as effects due to interaction with the environment (i.e. similar segments passing through different wind fields will have different fuel requirements). Thus, the estimated weight of the vehicle at a certain position in a path (and thus its performance characteristics) will vary from the original estimate if the path undergoes a mutation process during evolution (Section 2.2). If each segment is constructed based on *current* estimates of vehicle characteristics (i.e. weight at arrival), then whenever the path is subject to mutation, a recomputation of all subsequent segments would be required to account for the *new* vehicle characteristics, and very likely would need modifications to every segment (which in fact would have a 'propagation' effect for every segment). Thus, a mutation which was intended only at some middle part of the path would effectively change the path almost entirely and would loose characteristics that where not to be mutated from the parent path, affecting the overall idea of how the Evolution Algorithm should work.

To avoid this problem, instead of constructing segments restricted by *current* vehicle weight, they are built based on the most restricting performance criteria that the vehicle could have throughout the planned path. That is, segments are constrained by performance characteristics based on maximum or minimum weight (depending on the performance parameter in question) that the vehicle could have during the planned path. Effectively then, performance characteristics constraints are based on one of the following:

$$\begin{aligned} \text{maximum weight } &= \text{weight at start point for the planned path} \\ &\quad \text{(i.e. weight at end of the trajectory commited to fly)}, \\ \text{minimum weight } &= \text{empty weight.} \end{aligned}$$

Besides the performance parameters mentioned earlier, the planner is designed to consider varying fuel consumption rates of the vehicle (i.e. the UAV consumes less fuel as it gets lighter -it gets lighter as it consumes fuel). For this reason, even though maximum and minimum weights are used to determine performance characteristics to limit segment parameters, weight changes throughout the flight of a planned path are considered for fuel consumption computations. Therefore, whenever a path is modified at any segment (through the mutation process), the vehicle's weight is recalculated throughout the remaining path.

4.2 Performance Tables

The use of Performance Tables was chosen as a standardized input interface to provide the planner with vehicle performance information. The planner can then utilize any vehicle model for path planning.

Table 4.1 shows the parameters to read and their required arguments. Tables are read through linear interpolations of the appropriate arguments.

Other required inputs are Fuel Capacity, Empty Weight, Initial Fuel and Initial Weight (all in kg); an estimate of the best achievable acceleration (currently a single value in m/s^2, although can be extended to be a function of altitude, mass, etc.); as well as Maximum and Minimum Altitudes (m) for the flight.

4.2.1 Seascan Performance Tables

The Insitu Group (www.insitugroup.net) has provided a flight performance library (C-code) with functions that can be accessed to compute performance characteristics of Seascan UAVs. These performance functions use the lift coefficient as the central parameter. A C++ code has been developed to use this library to create performance tables according to the format and parameters required by the planner. Therefore, performance tables can be generated for any Seascan model with the use of the appropriate aircraft parameter file provided by The Insitu Group. The planner can run this application ($SeascanPerformance.exe$) during the initialization process to generate the performance table (in the case of using a Seascan UAV), or it can be fed with a pre-constructed table.

The use of accurate aircraft performance data in this planning system adds an important layer of realism.

Table 4.1: Required Parameters and Dependencies in Vehicle Performance Tables.

Dependency	Units	Parameter	Units
Altitude	m	Maximum Descent Rate (-)	m/s
Mass	kg	Maximum Climb Rate (+)	m/s
		Minimum Level Speed	m/s
		Maximum Level Speed	m/s
		Maximum Descent Slope (-)	m/m
		Maximum Climb Slope (+)	m/m
		*Minimum Descent Rate (-)	m/s
		*Minimum Climb Rate (+)	m/s
		*Minimum Descent Slope (-)	m/m
		*Minimum Climb Slope (+)	m/m
		*Cruise Best Range Fuel Use	kg/s
		*Cruise Best Range Speed	m/s
Altitude	m	Maximum Descent Speed	m/s
Mass	kg	Minimum Descent Speed	m/s
Slope	m/m	Maximum Climb Speed	m/s
		Minimum Climb Speed	m/s
Altitude	m	Minimum Turn Radius	m
Mass	kg	Level Flight Fuel Use	kg/s
Speed	m/s		
Altitude	m	Descent Fuel Use	kg/s
Mass	kg	Climb Fuel Use	kg/s
Slope	m/m		
Speed	m/s		

**Optional parameters*

Chapter 5

INTEGRATING WEATHER INFORMATION INTO THE PATH PLANNER

Expected weather conditions must be considered for the successful planning of long range flights. This Chapter presents the use of weather forecast information in the planner. Two main weather factors are considered: Forecast Winds and Icing Potential.

5.1 Sources of Weather Information and Prediction

5.1.1 GRIB-format Databases

In 1985 the World Meteorological Organization (WMO) approved a general purpose data exchange format designated FM 92-VIII Ext GRIB (GRIdded Binary) [22]. It is a compact bit-oriented format for efficient transmitting of large volumes of gridded data allowing faster computer-to-computer transmissions compared to character-oriented bulletins. It also serves as a data storage format with similar benefits in terms of information storage and retrieval devices. Therefore, numerous weather products are available in GRIB format. For such reason, GRIB format products have been adopted as the main source of atmospheric information for the path planner.

This standard format has continuously gone through revisions, changes and modifications throughout the years. However, by the end of the 20th century it became apparent that GRIB1 could no longer satisfy the requirements for new data and information management, and more structural changes where required. This triggered the development of GRIB Edition 2 (GRIB2, which in fact now stands for General Regularly-distributed Information in Binary Form), approved for operational use starting November 7, 2001. Although, ICAO

(International Civil Aviation Organization) requested continuation of GRIB1 for the dissemination of its World Area Forecast System (WAFS) products, for which the United Kingdom and United States WAFS centers agreed to continue the production of data in GRIB1. Thus, both GRIB1 and GRIB2 are currently in use and it is expected to continue this way for some time.

As of September 2003, the National Centers for Environmental Prediction (NCEP) of the U.S. have been running pilot studies on the use of GRIB2, and have maintained the publication of products using GRIB1. Having adopted the NCEP as the main operational source of weather information for our planner (due to the availability of databases through the internet), GRIB1 has been maintained as the format implemented for the use in our algorithm. We will therefore focus here on GRIB Edition 1. As GRIB2 becomes the standard format of use, it may become necessary to work on modifications in our algorithms and use of decoding programs to adapt to the appropriate changes. Main differences between the two formats may be found in [22], [23].

5.1.2 GRIB Edition 1

Each GRIB record contains a single parameter with values located at an array of grid points. Records are logically divided in six different sections to control, identify and define the data contained:

- Section 0: Indicator Section (IS)
- Section 1: Product Definition Section (PDS)
- Section 2: Grid Description Section (GDS) - optional
- Section 3: Bit Map Section (BMS) - optional
- Section 4: Binary Data Section (BDS)
- Section 5: End section (ES)

Each section's information is classified and contained in specific octets (bytes). With the exception of the first four octets of the Indicator Section, and the End Section itself, all octets contain binary values. Each section’s first three octets indicate its length, and distinguish each section from another (except for the IS and ES, which are invariant in length). We revise here the main information contained in each one. A much more detailed description, as well as tables of corresponding values for each octet may be found in GRIB format code guides ([24], [25], [26]).

Indicator and End Sections

The Indicator Section is eight octets long. It identifies the start of the record with the first four octets indicating ’GRIB’ in the International Reference Alphabet (IRA) No.5 (international name for the U.S. version known as ASCII). Octets 5 to 7 indicate the total length, in octets, of the GRIB record (including Sections 0 and 5). Octet eight corresponds to the Edition number of GRIB used.

The End Section is only four octets long containing ’7777’ coded in IRA No.5. This serves as indication of the ending of the GRIB record.

Product Definition Section (PDS)

This section provides relevant information for the path planner. It defines the initial time of the data in Coordinated Universal Time (UTC) as well as the time intervals of each forecast and the total time range included. It specifies the weather parameters in the report, the altitude level or layer, the grid type used, and whether a Grid Description Section is included. The complete PDS section structure is presented in Table 5.1.

Grid Description Section (GDS)

Even though the GDS section is optional, it is highly recommended that it be included in any report as it provides a detailed description of the 2-D grid type used. Even if the record is based on ’standard’ grid types, the inclusion of this section eliminates any question about the ’correct’ geographical grid for a particular field. The GDS content is presented in Table

Table 5.1: Product Definition Section.

Octet no.	PDS Content	
1-3	Length in octets of the Product Definition Section	
4	Parameter Table Version number. Currently Version 3 for international exchange. Parameter table version numbers 128-254 are reserved for local use.	
5	Identification of center	
6	Generating process ID number	
7	Grid Identification (geographical location and area)	
8	Flag specifying the presence or absence of a GDS or a BMS	
9	Indicator of parameter and units	
10	Indicator of type of level or layer	
11-12	Height, pressure, etc. of the level or layer	
13	Year of century	Initial (or Reference) time of forecast - UTC or Start of time period for averaging or accumulation of analyses
14	Month of year	
15	Day of month	
16	Hour of day	
17	Minute of hour	
18	Forecast time unit	
19	P1 - Period of time (Number of time units) (0 for analysis or initialized analysis). Units of time given by content of octet 18.	
20	P2 - Period of time (Number of time units) or Time interval between successive analyses, successive initialized analyses, or forecasts, undergoing averaging or accumulation. Units given by octet 18.	
21	Time range indicator	
22-23	Number included in average, when octet 21 indicates an average or accumulation; otherwise set to zero.	
24	Number Missing from averages or accumulations.	
25	Century of Initial (Reference) time (=20 until Jan. 1, 2001)	
26	Identification of sub-center	
27-28	The decimal scale factor D. A negative value is indicated by setting the high order bit (bit No. 1) in octet 27 to 1 (on).	
29-40	Reserved (need not be present)	
41-...	Reserved for originating center use.	

5.2

Bit Map Section (BMS)

The purpose of this optional section is to provide a bit map consisting of contiguous bits with a bit-to-data-point correspondance as defined in the grid description. If a bit is set equal to 1, it implies the presence of a datum for that grid point in the Binary Data Section; a bit value of zero implies the absence of such. This is useful in shrinking fields where fair portions of the field are not defined. For example, in the case of global grids of sea surface temperature, the bit map may be used to suppress the ’data’ at grid points over land by setting the corresponding bits to zero. It is not recommended to use the BMS if only a small portion of the grid has un-defined data, as adding the bit map may add more bits than the removal of few data values.

Binary Data Section (BDS)

The BDS section contains the packed data for the altitude level specified in the PDS, as well as the binary scaling information required to reconstruct the original data (different methods for packaging may be used in GRIB records). The contents of this section are presented in Table 5.3.

5.1.3 Weather Information Products

There are numerous weather products that are distributed in GRIB format. The two main products that have been used to run path planning simulations are GFS and Reanalysis information.

GFS stands for the Global Forecast System, which became operational in 2002 [27]. It is an integrated version of the Medium Range Forecast (MRF) and Aviation (AVN) models, providing real-time numerical weather forecasts for the entire globe out to 16 days four times per day (00UTC, 06UTC, 12UTC, and 18UTC). This integration provides increased

Table 5.2: Grid Description Section.

Octet no.	GDS Content
1-3	Length in octets of the Grid Description Section
4	NV, the number of vertical coordinate parameters
5	PV, the location (octet number) of the list of vertical coordinate parameters, if present
	or
	PL, the location (octet number) of the list of numbers of points in each row (when no vertical parameters are present), if present
	or
	255 (all bits set to 1) if neither are present
6	Data representation type
7-32	Grid description, according to data representation type, except Lambert, Mercator or Space View
or	
7-42	Grid description for Lambert or Mercator grid
or	
7-44	Grid description for Space View perspective grid
PV	List of vertical coordinate parameters (length = NV x 4 octets); if present, then PL = 4 x NV + PV
PL	List of numbers of points in each row, used for quasi-regular grids (length = NROWS x 2 octets, where NROWS is the total number of rows defined within the grid description)

Table 5.3: Binary Data Section Section.

Octet no.	BMS Content
1-3	Length in octets of binary data section
4	Bits 1 through 4: Flag
	Bits 5 through 8: Number of unused bits at end of Section 4.
5-6	The binary scale factor (E). A negative value is indicated by setting the high order bit (bit No. 1) in octet 5 to 1 (on).
7-10	Reference value (minimum value); floating point representation of the number.
11	Number of bits into which a datum point is packed
12-nnn	Variable, depending on octet 4; zero filled to an even number of octets.
14	Optionally, may contain an extension of the flags in octet 4.

resolution and additional satellite data allowing improved forecasts.

Reanalysis (historical) products are the result of a joint effort put in place by the NCEP and the National Center for Atmospheric Research (NCAR) called the Reanalysis Project. Its goal is to produce new atmospheric analyses using historical data (from 1948 onwards) as well as to produce analyses of the current atmospheric state.

These products provide, among many others, the parameters of interest for the current state of the path planner. Their processing and use is described in section 5.2.

5.1.4 Icing Potential Algorithms

Icing is a major issue in aerial vehicles, particularly affecting small aircraft. Small-sized UAVs are thus highly vulnerable to icing conditions. Continuous research has been taking place on several aspects of icing hazards: from ice accretion prediction models [28], [29], [30], to performance degradation [31],[32], to control of vehicles in icing encounters [33].

Certainly, identification and prediction of regions of potential icing have also been of much interest. In the past several years, a number of icing diagnostic and forecast algorithms have been under development. They have used different sources of information. A general observation is that algorithms based purely on model output capture most icing pilot reports (PIREPs) but tend to over forecast icing by indicating it in areas where clouds do not exist (a necessary condition for encountering icing). On the other hand, algorithms that are based primarily on data from instruments (such as radars, satellites, or surface observations) are quite accurate in the locations where they indicate icing, but tend to under forecast it because these instruments are unable to identify all icing locations.

In 2002 the Aviation Weather Center began the routine production of the Current Icing Potential (CIP) product developed by the NCAR In-flight Icing Product Development Team [34], [35]. The CIP is the operational version of the Integrated Icing Diagnosis Algorithm (IIDA) which has been verified over several years. It is an hourly 3-D *diagnosis* of icing likelihood which integrates information from five data sources: Rapid Update Cycle (RUC) model, satellite and radar data, surface observations (METARs) and PIREPs. With this integration of sources, it attempts to capture the maximum number of PIREPs while trying to avoid over indication of icing presence in a 3-D space. It outputs a *likelihood* of encountering icing, also referred to as *icing potential*.

The interest in determining icing conditions has certainly not only been restricted to diagnosis (analysis of current conditions), but to generate icing forecasts. Among other efforts and algorithms developed in recent years, the NCAR has also worked in developing the Integrated Icing Forecast Algorithm (IIFA), now the Forecast Icing Potential (FIP) algorithm, that became operational in January 2004. The FIP algorithm uses the CIP algorithm as a template. It creates forecasts every hour out to 3 hours, as well as 6, 9 and 12 hour forecasts every three hours. However, because of observations not being available in a forecast mode, the FIP needs to create surrogates from the model for the observations available to the CIP to build its forecasts. The FIP algorithm is explained in [36].

The performance of both algorithms has been studied and monitored continuously. In general, the method to evaluate their performance is based on computing three basic statistics: PODy = probability of detection of *Yes* PIREPs (proportion of positive icing PIREPs that were correctly forecast to be in locations with icing conditions), PODn = probability of detection of *No* PIREPs (proportion of negative icing PIREPs that were correctly forecast to be in locations with *no* icing conditions) and % Volume = the percentage of the airspace volume that has a Yes forecast icing. Their overall results have been quite good, considered as *skillful* as determined by the evaluation methods. Details on the methodology and statistical performance results of past years can be obtained from [37], [38], [39]. Results for the past 12 months (updated quarterly) for the CIP algorithm may also be found through the internet in the Aviation Weather Center's web-site ($http://adds.aviationweather.gov$).

For our purpose, an adapted version of the IIFA is used to provide the planner with information regarding areas of potential icing to avoid (section 5.2.2).

5.2 *Processing Weather Information for use in the Path Planner*

A vast amount of weather parameters are stored in GRIB formatted files. Since the parameters contained may vary from source to source, a pre-processing step has been developed in order to standardize the input data used by the path planner. Once this step has taken place, the decoding and processing of weather information occurs during the initialization stages of the Planner, followed by the computation of icing potential. This section provides a description of these steps (Figure 5.1).

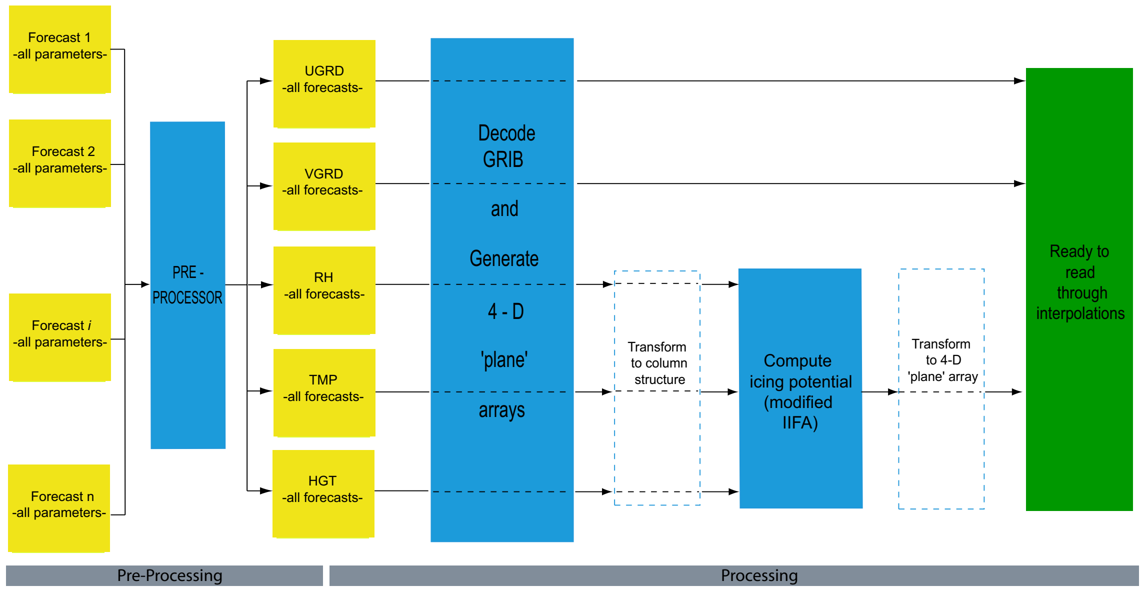
Forecast 1 -all parameters-
Forecast 2 -all parameters-
Forecast *i* -all parameters-
Forecast n -all parameters-
PRE - PROCESSOR
UGRD -all forecasts-
VGRD -all forecasts-
RH -all forecasts-
TMP -all forecasts-
HGT -all forecasts-
Decode GRIB and Generate 4 - D 'plane' arrays
Transform to column structure
Compute icing potential (modified IIFA)
Transform to 4-D 'plane' array
Ready to read through interpolations
Pre-Processing
Processing

Figure 5.1: Processing of Weather Information for the Planner.

5.2.1 *GRIB-format Weather Data Processing*

Pre-processor

One weather data file may contain several GRIB-format weather records (recall each record corresponds to one parameter; see section 5.1.2). The objective of the pre-processing step is to generate one binary input file per weather parameter (i.e. UGRD). It is expected that the raw GRIB sources contain records covering the complete flight time of interest (otherwise the last record is used to determine weather conditions for further times). The generated files have the following format:

- Each file contains all records for a given parameter at each pressure level and forecast times (records are sorted in order of increasing forecast time).
- Each record consists of a header and a data section.
- The header section has information from the Product Definition Section (PDS) and the Grid Description Section (GDS) from the original GRIB message.
- The data portion contains weather variable data decoded into native binary floating point format.

Also, as not all parameters from the weather sources may be required, this pre-processing step may serve as a filter. Only files for the weather parameters of interest may be generated. Similarly, it is possible to filter and retain only specific pressure levels records. For our purpose, the parameters of interest are:

- horizontal wind vectors UGRD and VGRD (m/s),
- temperature TMP (K),
- relative humidity RH (%),
- geopotential height HGT (gpm).

which are stored for the following pressure levels:

- 1000mb (occurs near 0m),
- 925mb,
- 850mb (occurs near 1500m),
- 700mb (occurs near 3000m),
- 600mb,

- 500mb (occurs near 5500m),
- 400mb

The pre-processing has been designed to accommodate both input files from GFS or Reanalysis products. The data extraction and separation process itself is made with the use of a freely distributed utility for GRIB data extraction called wgrib developed at the National Weather Service (refer to the wgrib home page $http : //wesley.wwb.noaa.gov/wgrib.html$ for details).

Processing

Having generated the files in the 'standard' format for the planner to read, the planner extracts the data at its initialization stages. The algorithm used to read the data simply follows the rules and structure for the PDS, GDS and BDS contents of stored records in the binary files for each parameter.

The 2-dimensional data grids contained in the BDS (and based on projection-type information from the GDS) are extracted as 2-D arrays of latitude-longitude data. The extraction from three of the most used projection-types are currently implemented: latitude-longitude grids, gaussian latitude-longitude grids, and polar stereographic grids (refer to PDS and GDS descriptions in GRIB format code guides for details); the first one being the commonly used for GFS and Global Reanalysis products.

With the time of forecast and pressure level information found on the PDS content for each 2-D BDS grid, complete 4-D arrays are constructed for each weather parameter (i.e. one 4-D array per binary file obtained from the pre-processing stage). These arrays (UGRD, VGRD, RH, TMP, HGT) constitute the source for direct reading of data for the planner during the planning process.

5.2.2 Computing Icing Potential

The CIP and FIP products (which are distributed in GRIB-format) currently only cover the continental United States (CONUS). However, we are interested in providing the planner with areas of potential icing that should be avoided in any region (i.e. over the Pacific Ocean). For this reason, the algorithms behind these products (particularly the IIFA) have been integrated with our path planner with appropriate and necessary modifications.

Integrating the IIFA with the planner must deal with the fact that the sources of input information used for the operational FIP are unavailable. One of the main sources is the Rapid-Update-Cycle 2 (RUC-2) weather model (refer to [40] for details). The RUC-2 provides high frequency updates of current conditions and short-range (0 to 12-hr) forecasts of high resolution by assimilating continuous observations and reporting every 1 hour (this model is in fact useful for confirming, or questioning, short-term predictions of longer time frame models such as the GFS). The RUC-2 is used in the IIFA to generate equivalent information that observations would provide to the CIP.

An alternative for the use of the IIFA in the planner is found from weather parameters included in the GFS and Reanalysis products. The RH (relative humidity) and TMP (temperature) from the weather products may substitute some of the outputs of the RUC model. These two parameters, aided by the HGT (geopotential height), serve as the inputs for the cloud forecast scheme in the IIFA providing an estimate of cloud presence and icing potential. And having extended forecast weather products, areas of potential icing may be computed for all records in which these parameters are available.

The computation of icing potential takes place after the RH, TMP and HGT arrays are generated. From these, a similar 4-D ICING array is constructed filled with icing potential data. Because the IIFA scans the atmosphere vertically (i.e. columnwise), the RH, TMP and HGT *plain* grids are rearranged into *column* grids prior to feeding the IIFA. The icing potential grid obtained from the algorithm is re-arranged into plain type grids to be used the same way the UGRD and VGRD are.

The use of this adapted IIFA is definitely not expected to provide exactly the same results as those that the FIP may generate, as other inputs such as initial data from satellites are unavailable. Nevertheless, the results are quite close and it is certainly advantageous to provide the planner with 1) estimates of icing potential areas for any region and 2) for extended forecasts, not available from regular FIP reports. Figure 5.2 shows a comparison example between the icing potential as computed with our implemented algorithm versus actual reports from the CIP (current conditions). Figure 5.3 shows a similar comparison, comparing an FIP report for a 12hrs forecast based on the same initial report as in Figure 5.2.

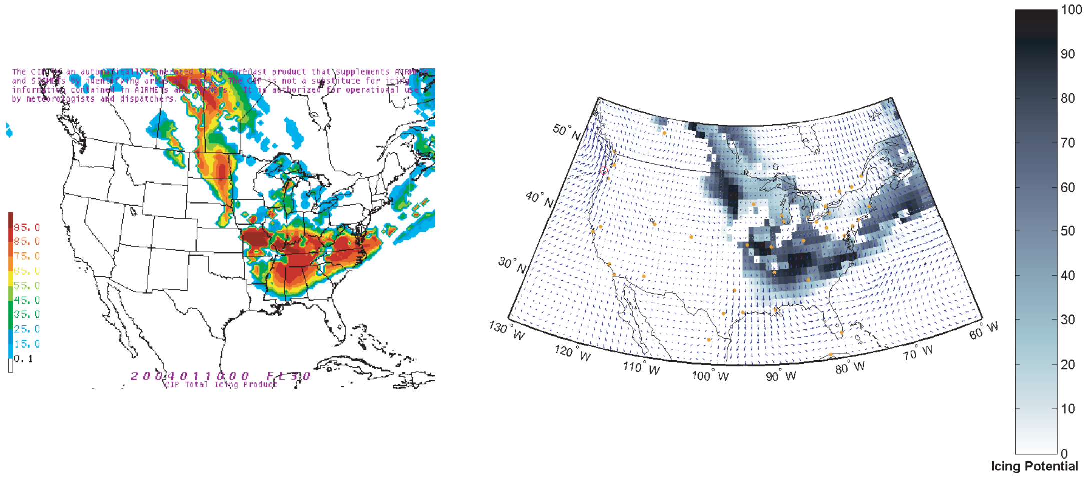

Figure 5.2: Current Icing Potential Comparisons. *Left*: CIP report on Jan10,2004 at 00UTC. *Right*: Computed Icing Potential for path planner using GFS report of Jan10,2004 at 00UTC for 00UTC forecast. Altitude: FL030 (915m).

5.3 Reading Data and Planning through Wind Fields and Icing Potential

The UGRD, VGRD and ICING 4-D grid arrays are used through linear interpolations to determine wind field and potential icing encountered according to the vehicle's position

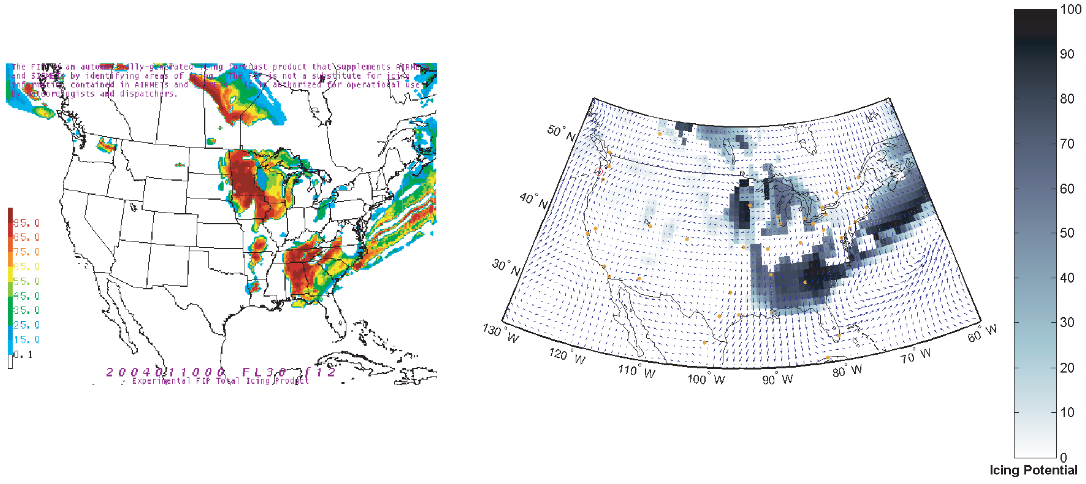

Figure 5.3: Forecast Icing Potential Comparisons. *Left*: FIP report on Jan10,2004 at 00UTC for 12UTC forecast. *Right*: Computed Icing Potential for path planner using GFS report of Jan10,2004 at 00UTC for 12UTC forecast. Altitude: FL030 (915m).

and time, as explained below. A standard atmosphere is assumed for the geometric-altitude to pressure-altitude relationship.

5.3.1 Use of Wind Data

Wind field data from the UGRD and VGRD array is used in a double sense (although related one to the other). They are used to determine the time of arrival to any point of a path, and from this information two other data are obtained: the fuel consumed to get there, and the weather state present at that time, for that location. This information is computed as follows.

At the evolver initialization stage, launch time and position are used to determine the initial state of the wind field by linearly interpolating data from the UGRD and VGRD. Once the initial weather state has been determined, the path planner starts generating the first population of paths. Every time that a new segment is created, the planner estimates the time it will take to traverse it and stores the end clock time as part of the segment's information. This time estimation is computed at a sub-segment level based on length, airspeed at the sub-segment's start point, a random or pre-defined end airspeed, and the wind field at the initial state of the sub-segment to estimate the groundspeed when traversing the sub-segment's length. This calculation provides the approximate time at the end of the sub-segment, which is then used to determine the wind field at that state (latitude, longitude, altitude) for the next sub-segment. In addition, after a mutation process (which can modify a path at any point along its length), the planner re-computes the estimated end times for all follow-on segments in order to determine new wind fields corresponding to their new 4-D locations. This time-variable consideration allows the planner to compute fuel consumption by taking into account the wind fields that the aircraft will encounter at the moment it estimates it will arrive at each location. Thus, when going through the fitness evaluation and selection process, it is able to generate better solutions for the complete flight.

5.3.2 Icing

Icing potential has a more passive role than the use of wind fields. Its information is effectively used only during fitness evaluation of paths in the population. Again, on a subsegment discretization level, icing potential values are obtained through interpolations from the ICING array according to position and time. A *single icing potential* value Υ for the complete path used for fitness comparisons is computed as:

$$\Upsilon = 1 - \prod_i (1 - \kappa_i) \tag{5.1}$$

where κ_i is the icing potential value at position and time i.

5.3.3 Forecast Updating

In an actual flight, the weather data (winds, icing) for future segments of the flight path cannot be precisely known, as they constitute data from numerical model outputs. As mentioned previously, the planner linearly interpolates this information to provide itself with wind field estimates at any given location and time. Moreover, the planner may be injected with updated forecast data (as it typically occurs every six hours) and incorporate new information into its continuous in-flight planning process. In this way, updated (and presumably more accurate) forecasts can begin to influence ”future” segments of the flight path-those segments not yet converted into trajectories. This feature allows the planner to adapt to possible changes in the expected environment. To exploit this feature and allow for path refinement even if no important changes occur to the expected environment, planning is not confined to a pre-launch exercise. Rather, continuous planning and re-planning is performed during the entire flight. At a predetermined time interval, a ”present” path section is selected as the next actual trajectory to be flown, but the remaining ”future” path segments continually evolve until the next time interval. This system allows modifications to the planned path if better solutions based on updated information are found along the way.

Chapter 6

PATH PLANNING SUBJECT TO WIND FIELDS, ICING POTENTIAL AND TARGETED FLIGHT

This chapter presents a series of planning results showing several characteristics of the algorithm and its behavior on different scenarios. Each section discusses one case of interest:

- Section 6.1: Planning through Wind Fields
- Section 6.2: Re-planning for Updated Wind Field Forecast
- Section 6.3: Icing Avoidance
- Section 6.4: Site Observation
- Section 6.5: Altitude Scan
- Section 6.6: Local vs Global Optima: Potential for Parallel Evolvers

The first three sections are specifically intended to show the behavior of the path planning algorithm through realistic weather environments. The next two sections present additional characteristics in which the planning involves some targeted flight. The last section has been included as a clear example of the potential benefits of using parallel evolvers.

Every case presented here includes maps of the flight and a corresponding table specifying the data and parameters used for the path planning. Each map presents the planned paths (black) as well as the trajectory that the vehicle has followed after specified time intervals (red). The estimated consumed fuel, elapsed time and current altitude are indicated at the left of each map. Weather conditions correspond to the altitude at which the aircraft is currently positioned. As a reference, the maximum and minimum wind speeds and their location are also indicated. In the cases in which icing potential is considered, a colorbar to the right of each map is included.

The planning examples presented here used performance data from Insitu’s vehicles. The fuel consumption model uses: full throttle for climbs, idle engine for descents, best range (km/kg) for cruise. In some cases the fuel consumption as computed by the planner exceeds the fuel on board of the vehicle; however, the relevance of the results is the behavior of the planning algorithm. The results from the planner can thus be used to define the range of capabilities required from a specific vehicle to achieve the given mission objective. Further improvements to the planner may add the ability to deviate to alternate ’landing sites’ if necessary.

Dynamic planning (see Section 2.2.5) is used in all cases. This allows the planner to continuously search for other solutions as well as to refine planned paths even in cases where no new forecast information is received during flight. Two independent evolvers perform the same vehicle path search. At scheduled intervals, just before the planner selects the next trajectory to traverse, the best path results of both evolvers are compared. The evolver whose population contains the overall best path (based on fitness value) is selected to continue evolving the future path segments (the normal process for a single evolver). The second evolver is ’reset’ and begins replanning from scratch for the remaining distance, with a handicap of a specified number of generations. Currently no information sharing occurs between the evolvers which could eventually result in converging more rapidly to better solutions.

6.1 Planning through Wind Fields

The objective of this case is to show the ability of the planner to find routes that reduce the fuel consumption by taking advantage of favorable tailwinds based on future wind information. Several planning examples are presented as a matter of comparison. Flights are planned from Honolulu, HI to Long Beach, WA using weather information of November

11, 1998.

For the first example the flight was bounded between 300m and 5000m. Figure 6.1 shows several snapshots of the flight. After the pre-planning stage, each evolver suggested two different paths; the planner chose the best of the two. It can be seen from the figures how the actual flight followed a path that certainly resembles one of the initially planned paths. This shows how the planner was able to consider future winds that it estimated it would encounter during the flight. Furthermore, en-route re-planning continued with both evolvers achieving refinements to the planned path. It is clear from the figures that the aircraft is well-positioned based on the winds at each time. While the available wind information does not contain vertical wind field components, the variations in the horizontal (dominant) wind field at different altitudes are taken into account by the planner. Figure 6.1 presents the altitude history of this flight showing that the planner is able to exploit all three dimensions.

To compare this solution against a shorter flight (in distance), the simulation was repeated with the aircraft forced to fly along a straight path, though still allowed to alter its altitude. Figure 6.3 shows that the total fuel consumption for this path was higher than the path freely generated by the planner in Figure 6.1. Figure 6.4 shows that the planner decided once again to fly at various altitudes for this second case even when laterally restricted to fly straight to the goal.

In order to compare what the planner could achieve while restricting the straight path to a constant altitude, Figure 6.5 is presented. In this case, the planner was restricted to fly straight at 300m, the initial altitude of the previous examples. The estimated fuel consumption for this case is clearly higher. Table 6.1 presents the specific parameters used for these three planning examples. Table 6.2 compares the results for the three cases.

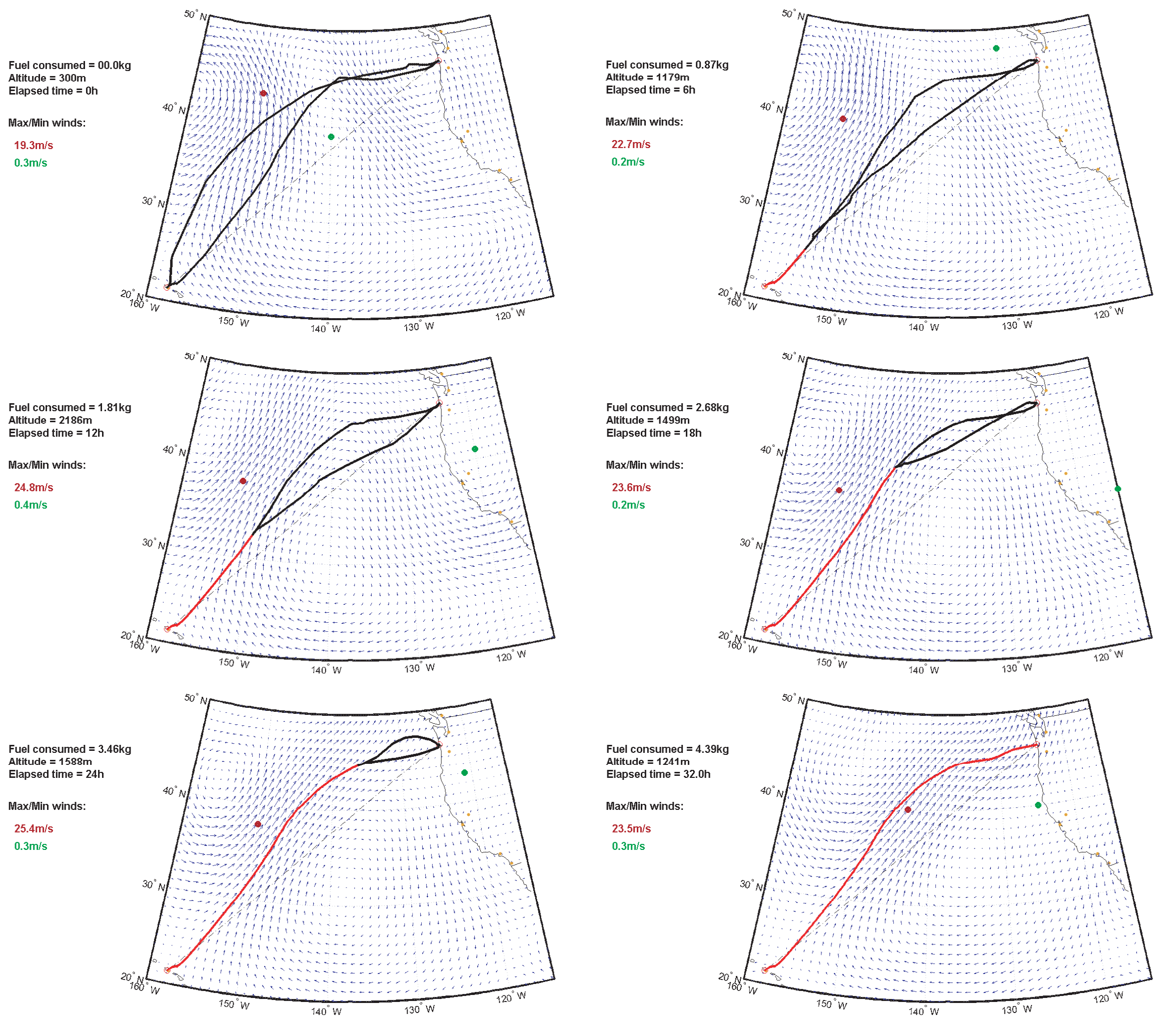

Figure 6.1: Free Planning through Winds, November 11, 1998.

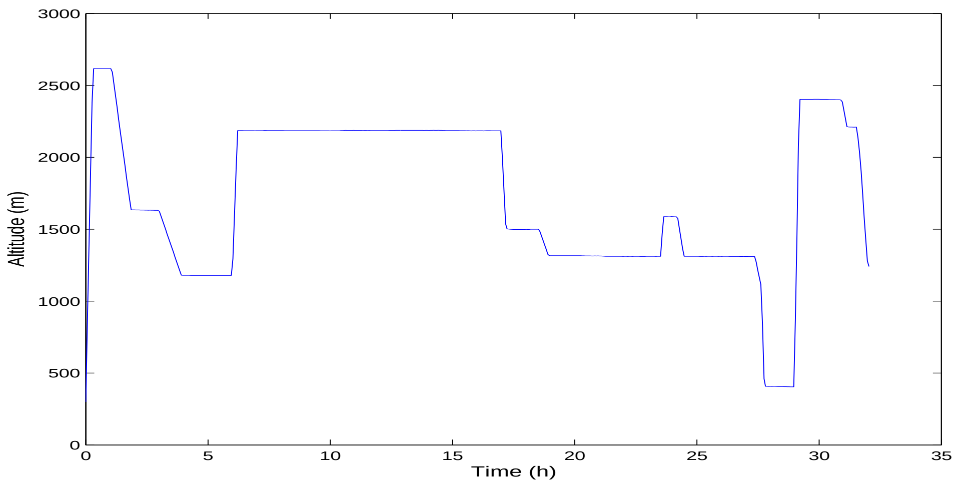

Figure 6.2: Altitude Time History of the Flight in Figure 6.1.

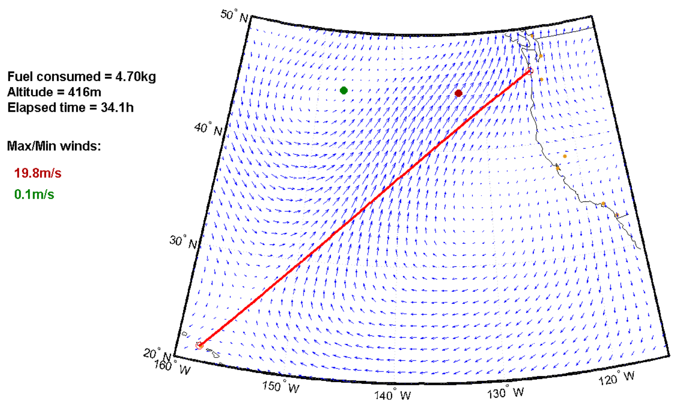

Figure 6.3: Direct Heading Planning through Winds, November 11, 1998.

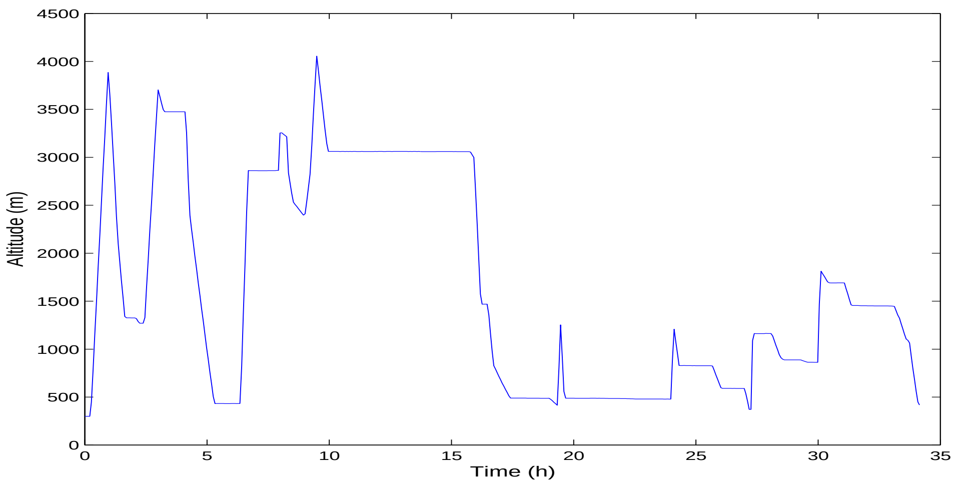

Figure 6.4: Altitude Time History of the Flight in Figure 6.3.

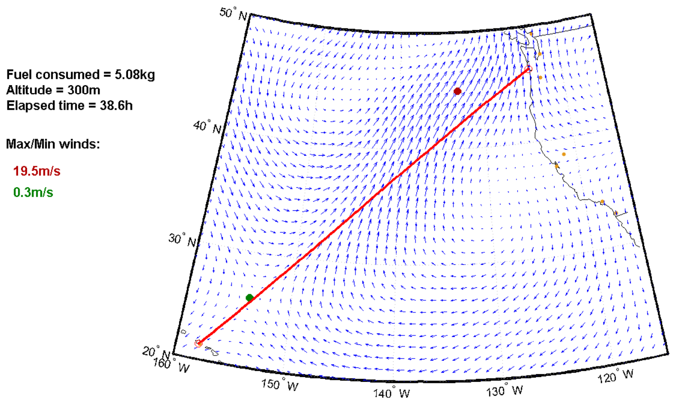

Figure 6.5: Straight Path at Constant Altitude, November 11, 1998.

Table 6.1: Settings for Planning through Winds Case.

Flight:	Honolulu, HI - LongBeach, WA
Launch Time (UTC):	November 11, 1998 (00:00)
Initial Altitude:	300m
Altitude Bounds:	300m-5000m / 300m-5000m / 300m
Weather Source:	Reanalysis
Trajectory commitment & replanning:	Every 3 hrs
Updated weather information during flight:	No
Vehicle:	Seascan (acp00003.002)
Fuel Capacity:	5.22kg
Planner settings:	
Parents, Population:	8,16
Pre-planning generations:	100
En-route planning generations:	100 (x2 for the loosing evolver)
Constraints in Planner:	
Fuel:	Yes
Icing:	No
Target:	No
Altitude Scan:	No
Other:	Free / Direct (free Altitude) / Straight

Table 6.2: Results Comparison for the Planning through Wind Fields Cases.

Example Flight	Fuel Consumed	Elapsed Time
Free Planning	4.39 kg	32.0 h
Direct Heading, Free Altitude	4.70 kg	34.1 h
Straight Path at Constant Altitude	5.08 kg	38.6 h

6.2 Re-Planning for Updated Wind Field Forecast

The planning examples presented in the previous section demonstrate how the planner can plan for a complete flight based on future wind information. In the interest of demonstrating the adaptability of the algorithm, a more realistic case will be demonstrated next. In this case, even when the planner has initial pre-flight forecast information on future wind fields, updated forecasts will become available during flight.

For the following simulation, historical weather data was again used as the source of wind information. As using this data would give the planner access to wind field information at any given location and time, the following results were obtained by simulating ’new wind field forecasts’ at predetermined intervals. This was accomplished by advancing the airplane’s clock at each interval so that it considered wind data corresponding to future times. In effect, this forced the planner to readapt the path during flight to a new ’forecast’ different from one it had previously considered. This is similar to what would occur if updated forecast information were provided to the planner.

Figure 6.6 presents several snapshots of the planned flight sequence, corresponding to a launch time on February 5, 1997 at 00:00UTC. Updated forecast data was simulated by advancing the weather data by twelve hours at every trajectory selection interval, which was set to occur every six hours. These figures show how the initial plan suggested a nearly straight path to the goal. It is in general maintained for the first 18 hours of flight, even though wind time had already been advanced twice in 12hr intervals (thus suggesting that even the ’new forecasts’ were not affecting the plan much). However, once the first 18 hours were flown, the figure shows one of the evolvers suggesting a plan that deviates from the originally selected path due to more intense forecasted wind changes. The planner decides to take this alternate route. The remainder of the path did not require a second major change even when the wind time was again advanced twice. Once more, note how the aircraft is well-positioned based on the wind fields present at each time. Figure 6.2 shows the altitude history of this flight. Table 6.3 presents the parameters used for this example.

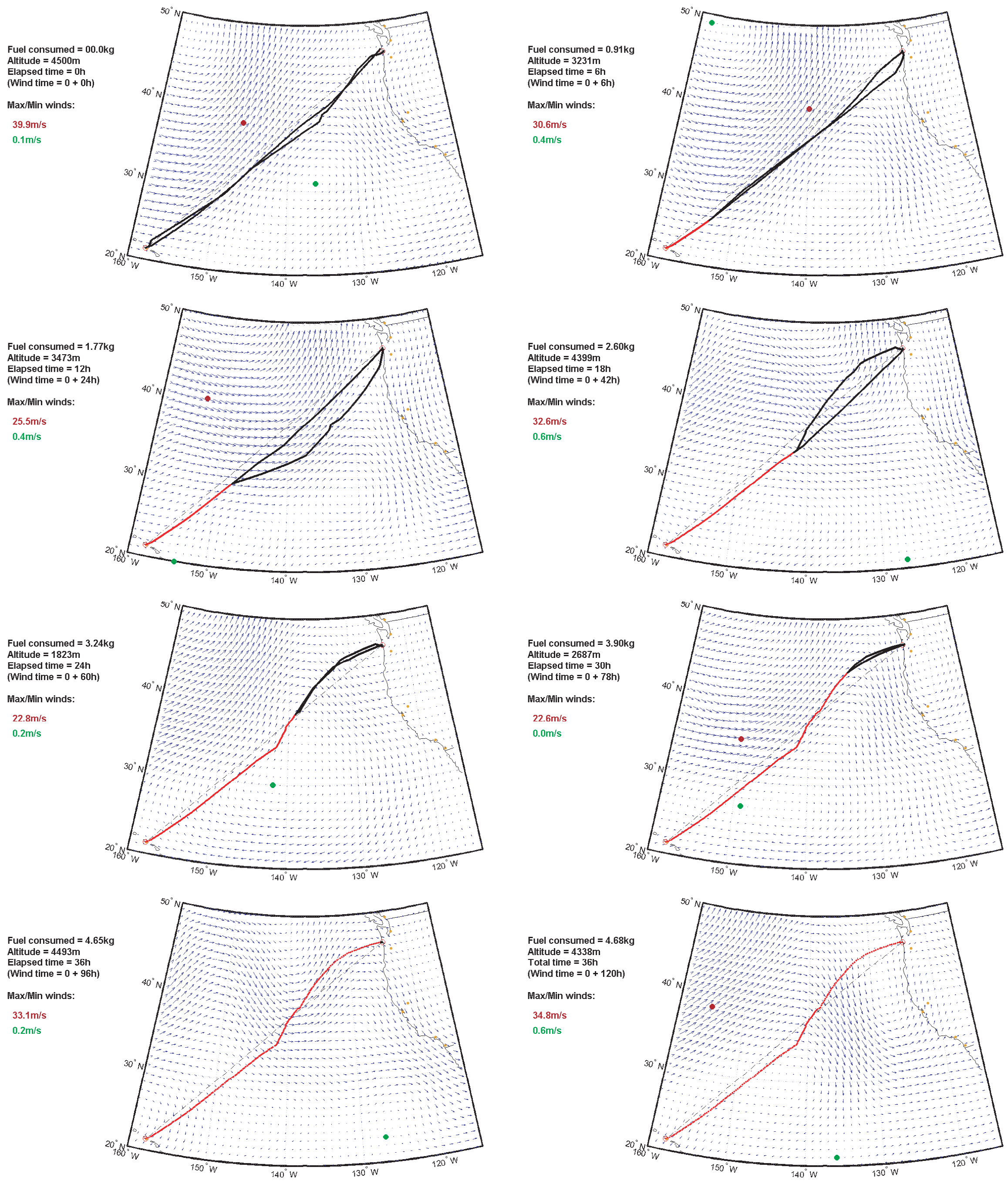

Figure 6.6: Forced Forecast Jumps, February 5, 1997.

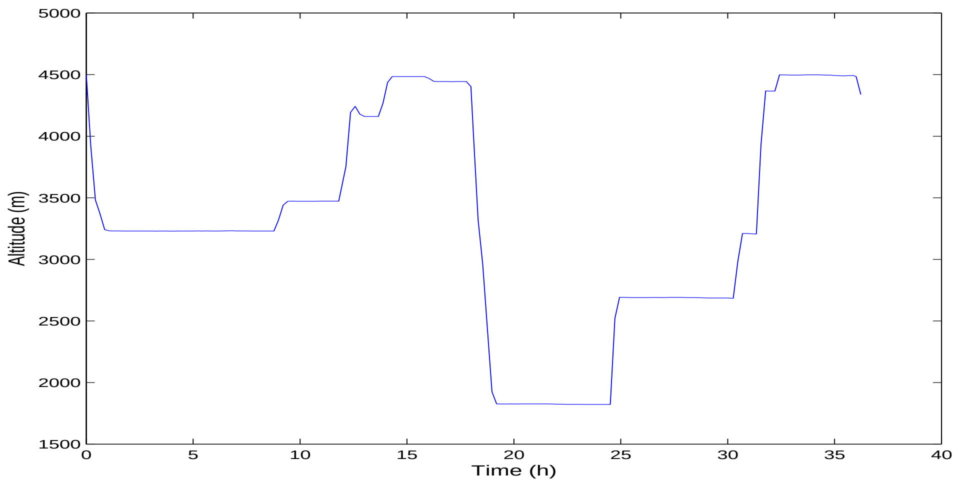

Figure 6.7: Altitude Time History of the Flight in Figure 6.6.

Table 6.3: Settings for Forced Forecast Jumps Case.

Flight:	Honolulu, HI - LongBeach, WA
Launch Time (UTC):	February 5, 1997 (00:00)
Initial Altitude:	4500m
Altitude Bounds:	300m / 4500m
Weather Source:	Reanalysis
Trajectory commitment & replanning:	Every 6 hrs
Updated weather information during flight:	Yes (+12hrs winds every cycle)
Vehicle:	Aerosonde
Fuel Capacity:	4.95kg
Planner settings:	
Parents, Population:	8,16
Pre-planning generations:	200
En-route planning generations:	200 (x2 for the loosing evolver)
Constraints in Planner:	
Fuel:	Yes
Icing:	No
Target:	No
Altitude Scan:	No
Other:	None

6.3 Icing Avoidance

The examples presented next show some comparison results when constraining the planner to completely avoid any region of potential icing (not considered in previous examples). They correspond to a launch time set for February 2, 1999 at (02:02UTC). Figure 6.8 presents the snapshots for a constant altitude flight at 5000m. By following the sequence carefully, it may be noticed how the planner made the vehicle 'wait' for the first several hours before a favorable clearing was available. From the snapshot corresponding to 24 hours of elapsed time it may be noticed how the planner had already developed a plan anticipating the *opening*, with the correct timing to get through.

By allowing a free altitude plan, rather than constraining it to a constant altitude, Figure 6.9 shows how the planner rapidly descends in order to avoid icing in high altitudes. The resulting path is certainly more fuel efficient than in the previous constant altitude case.

Table 6.4: Settings for Icing Avoidance Case.

Flight:	Honolulu, HI - LongBeach, WA
Launch Time (UTC):	February 2, 1999 (02:02)
Initial Altitude:	5000m
Altitude Bounds:	5000m / 300m - 5000m
Weather Source:	Reanalysis
Trajectory commitment & replanning:	Every 6 hrs
Updated weather information during flight:	No
Vehicle:	Seascan (acp00007.004)
Fuel Capacity:	3.9kg
Planner settings:	
Parents, Population:	8,16
Pre-planning generations:	400
En-route planning generations:	100 (x2 for the loosing evolver)
Constraints in Planner:	
Fuel:	Low constraint / Yes
Icing:	Yes
Target:	No
Altitude Scan:	No
Other:	None

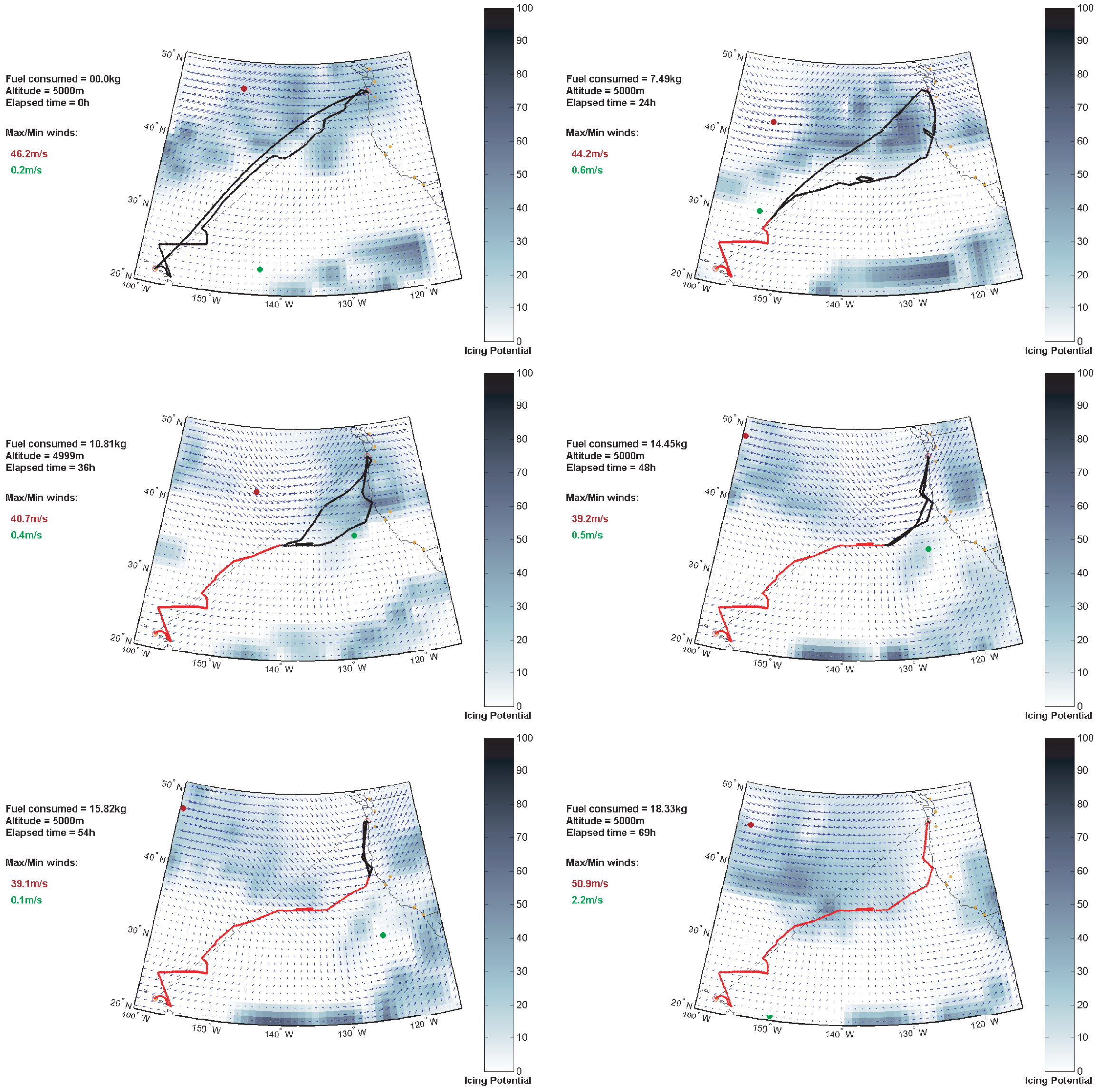

Figure 6.8: Icing Avoidance at Constant Altitude, February 2, 1999.

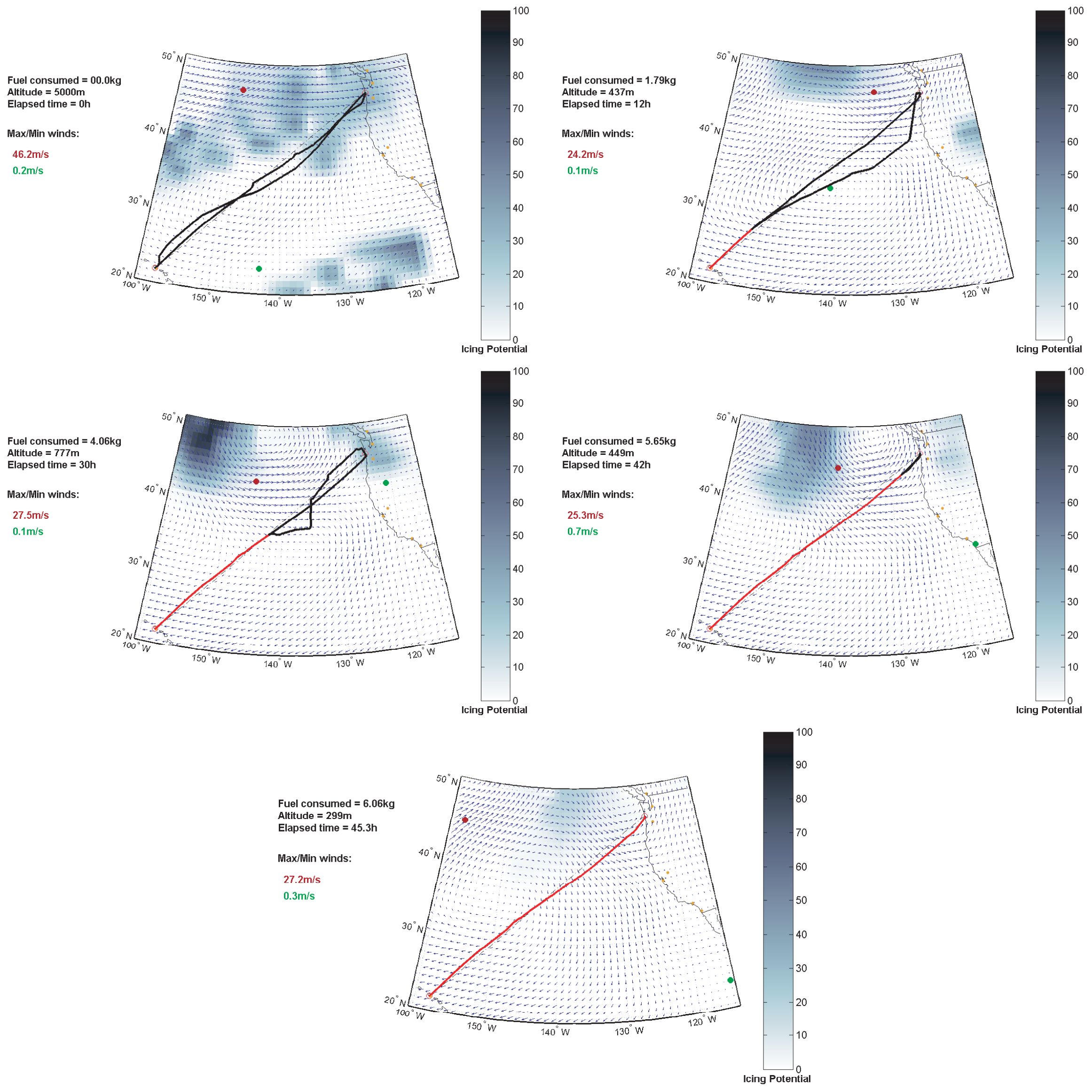

Figure 6.9: Icing Avoidance, Free Altitude, February 2, 1999.

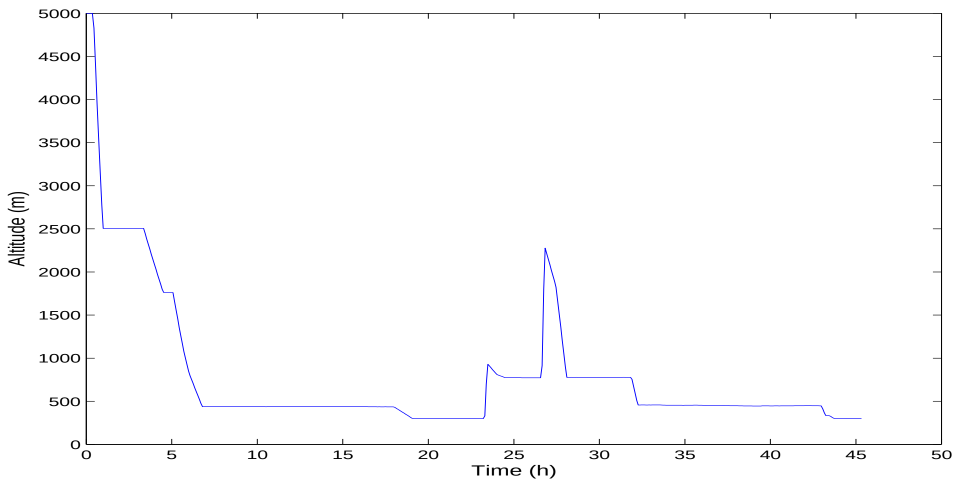

Figure 6.10: Altitude Time History of the Flight in Figure 6.9.

6.4 Site Observation

The following planning example simply shows the ability of the planner to consider not only flying between two points, but to respond when instructed to visit a specified observation area. This is achieved by adding a static site into the planner's environment. Figure 6.11 shows the resulting flight from Long Beach, WA to San Diego, CA requiring a visit to a specified location in the Pacific. Table 6.5 presents the parameters used in the planner.

6.5 Altitude Scan

This last example shows the implementation of an added feature in the planner involving full use of the 4-dimensional capability. The planner is not only instructed to fly to the goal, but it is required to follow a path that *scans* the atmosphere at specified intervals. This scanning flight path must, however, not be a hard constraint but part of all the soft constraints within the planner's cost function, as it has to allow flexibility in the planner to decide whether it is best for the mission to follow such scanning trajectory versus, for example, making sure fuel consumption is maintained within capabilities, or assuring that the aircraft not enter into icing conditions at some part of the instructed scanning path. The

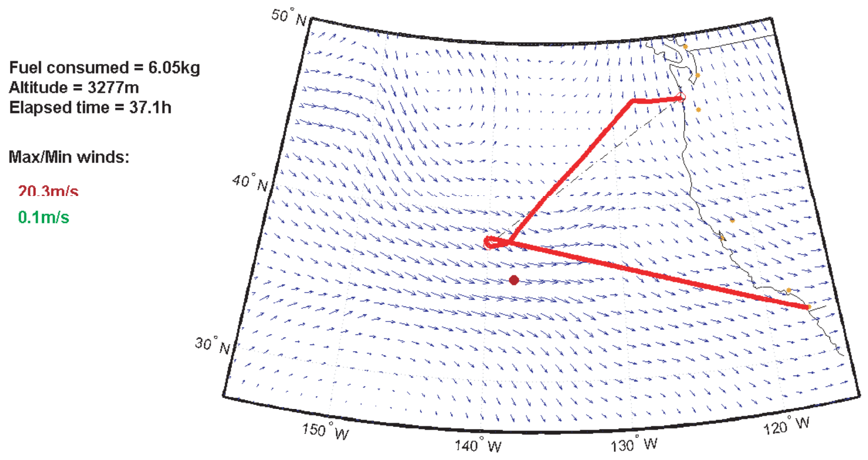

Figure 6.11: Site Observation, November 4, 2003.

Table 6.5: Settings for Site Observation Case.

Flight:	Long Beach, WA - San Diego, CA
Launch Time (UTC):	November 4, 2003 (00:00)
Initial Altitude:	5000m
Altitude Bounds:	300m - 5000m
Weather Source:	GFS
Trajectory commitment & replanning:	No
Updated weather information during flight:	No
Vehicle:	Seascan (acp00003.002)
Fuel Capacity:	5.22kg
Planner settings:	
Parents, Population:	15,30
Pre-planning generations:	400
En-route planning generations:	-
Constraints in Planner:	
Fuel:	Yes
Icing:	No
Target:	Yes (lat: 40°, lon: 228°)
Altitude Scan:	No
Other:	None

way this feature was implemented involves in part the use of the task concept presented by Pongpunwattana [41]. It is not a task planning process, but it uses the concept of tasks and time windows, where each of them has a score that is a directly related to the time the task is executed.

For the application presented here, tasks are set to match altitudes at specified intervals and time windows. This task achievement must be done efficiently *during* the trajectory from one point to another, and a path must not be rewarded if tasks are not spread in space (i.e., we would not want the vehicle to fly directly to the goal at some constant altitude, and then start scanning the atmosphere just in a region close to the goal until it runs out of fuel). But as the planner must still be able to freely determine the path to follow, altitude tasks must at the same time not be associated to a specific region in the horizontal plane (i.e. latitude, longitude) as an attempt to spread the scanning path in space. The approach is then to use an 'adaptable' method of task assignment where the number of tasks to reward are not fixed, but depend on each particular path in the population. As a remark, consider that the best trajectory will very likely have a different 2-D projection (i.e. as seen from above) if a scanning type path is requested rather than letting the planner to freely plan in all dimensions.

Figures 6.12 and 6.13 present an example result of the implementation of this feature considering the above mentioned conditions. In particular, Figure 6.13 presents the altitude history of the flight together with the requested scan time windows. Potential icing conditions were ignored for this example to appreciate the results of the scanning path and the desired scan without possible altitude limitations due to icing regions that would have made the planner not execute the desired scan (certainly a flight from Alaska to Washington State is very likely to present icing conditions).

With this approach, any type of scanning profile (frequency, altitudes, time windows) may be instructed to the planner while maintaining flexibility rather than being a hard constraint.

Table 6.6: Settings for Altitude Scan Case.

Flight:	Alaskan Peninsula - Long Beach, WA
Launch Time (UTC):	November 10, 2003 (00:00)
Initial Altitude:	4000m
Altitude Bounds:	300m - 5000m
Weather Source:	GFS
Trajectory commitment & replanning:	Every 6 hours
Updated weather information during flight:	No
Vehicle:	Seascan (acp00003.002)
Fuel Capacity:	5.22kg
Planner settings:	
Parents, Population:	30,60
Pre-planning generations:	1000
En-route planning generations:	100 (x2 for the loosing evolver)
Constraints in Planner:	
Fuel:	Yes
Icing:	No
Target:	No
Altitude Scan:	Yes (Scan Down)
Other:	None

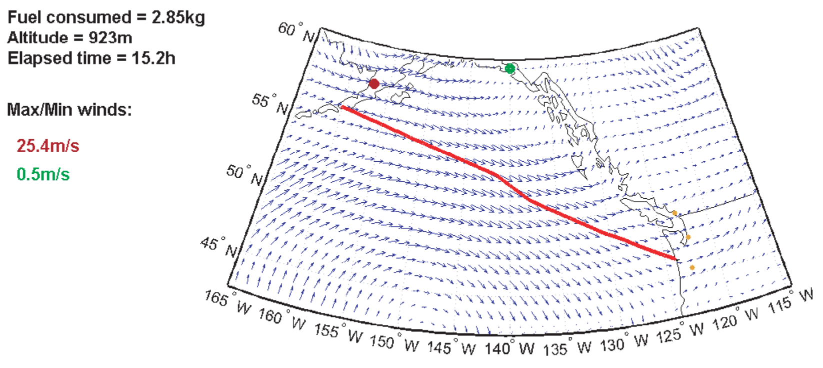

Figure 6.12: Altitude Scanning Flight, November 10, 2003.

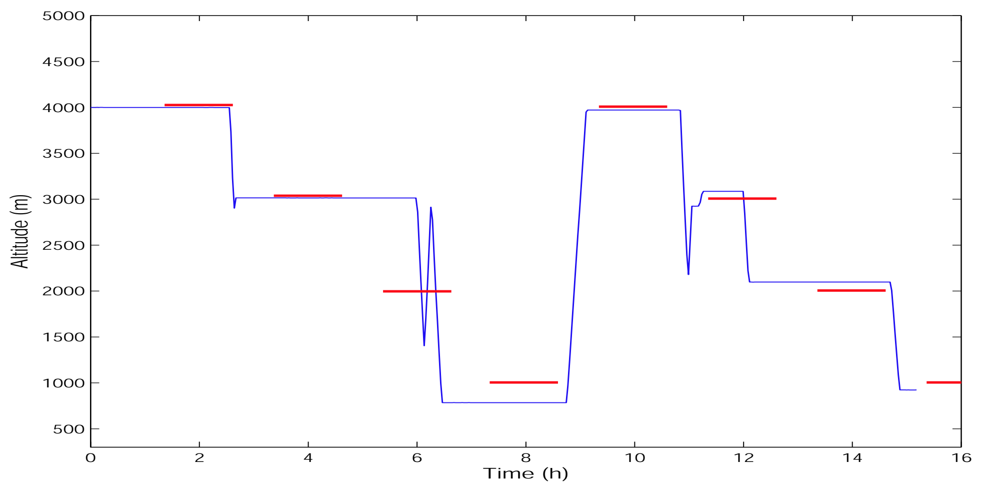

Figure 6.13: Desired Altitude Scan and Altitude Time History of the Flight in Figure 6.12.

6.6 Local vs Global Optima: Potential for Parallel Evolvers

Dealing with weather information in a 4-D environment makes it difficult in most cases to determine *visually* if the planner has achieved a local or a near-global optimum; a global optimum may not be as clear as in other types of path planning problems involving fixed targets and 2 dimensional search spaces. However, an interesting case was found where a local vs. a global optimum is quite clear in our application, which is worth presenting. It serves as a good scenario where the use of parallel evolvers may have a potential in improving the solution obtained by the algorithm (section 2.2.6).

The scenario is as follows. Using forecast information corresponding to November 10, 2003 at 00:00hrs, a flight from Honolulu, HI to San Diego, CA is planned. It was found that at high altitudes ranging from about 4000m to 5000m, strong winds were forecast south of the great circle from Honolulu to San Diego. Below these altitudes winds are not as strong, and in fact they are forecast as crosswinds and headwinds for a flight between the two cities. On the other hand, forecast winds north and near the great circle path are also strong crosswinds and headwinds at mainly all altitudes due to a high pressure system located north of the route. Only by following the clockwise direction of the high pressure system, for which it is necessary to travel first a long distance to the north, tail winds may be found for almost the entire flight at basically any altitude. The following examples and figures will make this situation much clearer. A flight south of the great circle at high altitudes would achieve the best fuel savings; although the region is small compared to the entire search space, not only due to the narrow flow of the favorable winds but due to the presence of icing. Flying around the high pressure system will also allow for good savings compared to a simple direct flight; this is a local optimum which is much easier to find as it includes a much larger altitude range and spread area in comparison to the ’global’ optimum, and where lower altitudes are free of icing conditions.

Figure 6.14 presents a case in which one of the evolvers was able to find the global optimum, while the other was *stuck* in a local optimum. The planner selected to follow the best path achieving the best fuel consumption. As a comparison, Figure 6.17 presents a case where the path followed corresponds to a local optimum. The total fuel consumption is higher than the global, but is certainly much lower than what the vehicle would require if flying just straight to the goal (as an interesting point, notice how the planned path at the beginning perfectly considered the movement and timing of the high pressure system, as shown on the snapshot corresponding to 36 hours). Note that even having two evolvers in this case, both pre-planned away from the global optimum. Chances are certainly improved with the use of two evolvers, but the highest potential of this concept involves continuous comparison and population mixing between evolvers; the examples presented here are only a preliminary use of two evolvers, where no information sharing occurs. A deep evaluation of the parallel evolution concept would involve assessment of Evolution-based Algorithms, which is beyond the scope of this work. Table 6.7 presents the data and planner settings for the examples presented here.

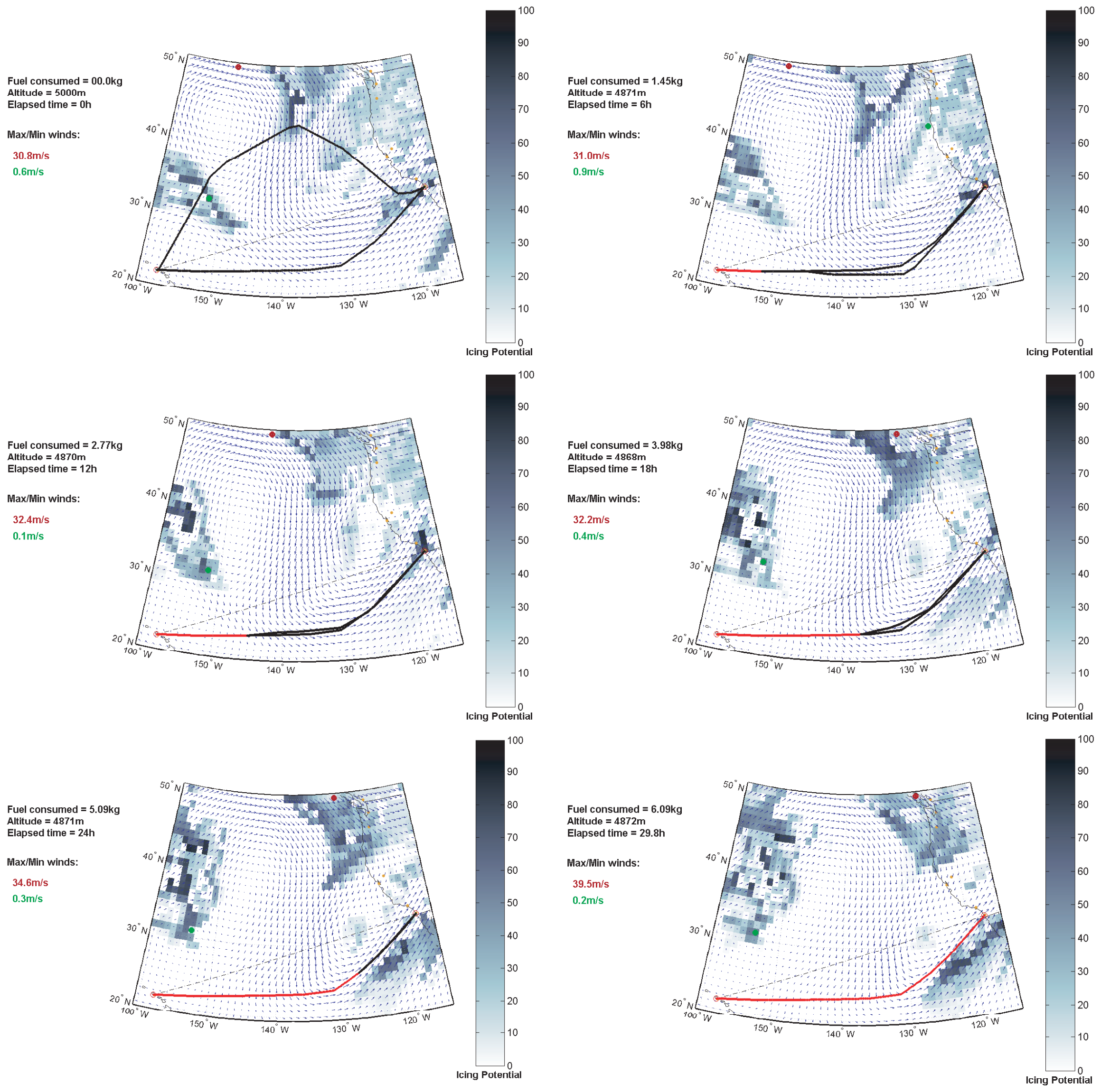

Figure 6.14: Global Optimum, November 10, 2003.

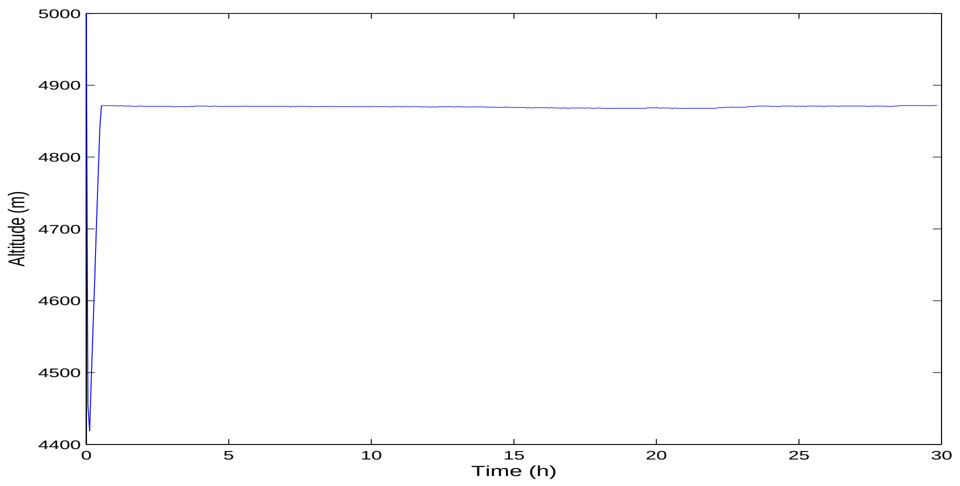

Figure 6.15: Altitude Time History of the Flight in Figure 6.14.

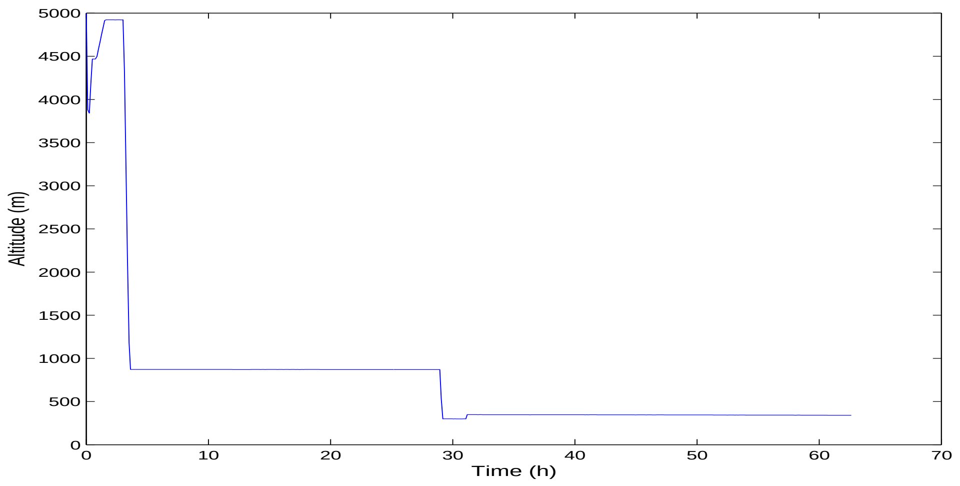

Figure 6.16: Altitude Time History of the Flight in Figure 6.17.

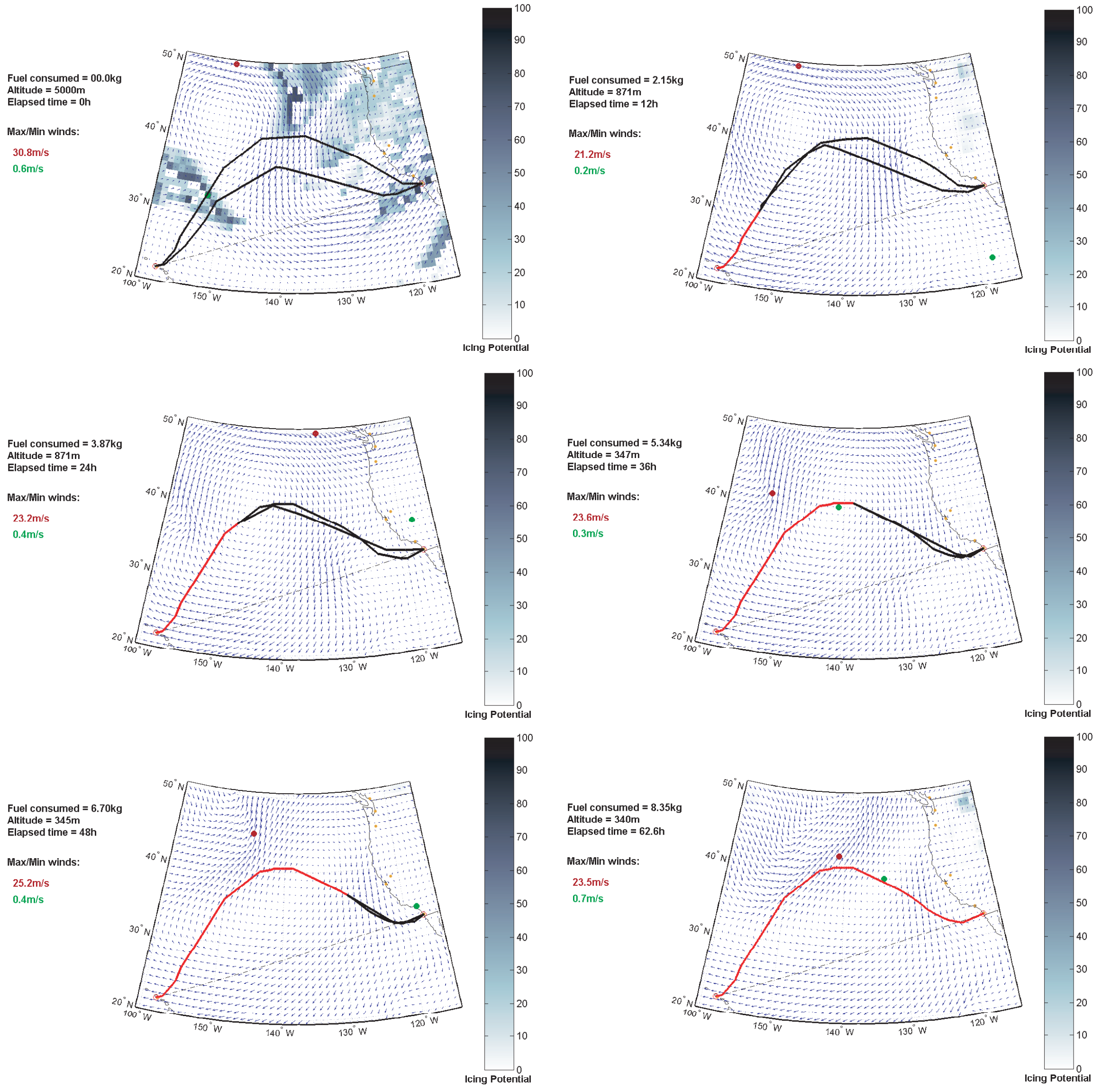

Figure 6.17: Local Optimum, November 10, 2003.

Table 6.7: Settings for Global vs. Local Optimums Case.

Flight:	Honolulu, HI - San Diego, CA
Launch Time (UTC):	November 10, 2003 (00:00)
Initial Altitude:	5000m
Altitude Bounds:	300m - 5000m
Weather Source:	GFS
Trajectory commitment & replanning:	Every 6 hrs
Updated weather information during flight:	No
Vehicle:	Seascan (acp00003.002)
Fuel Capacity:	5.22kg
Planner settings:	
Parents, Population:	25,50
Pre-planning generations:	200
En-route planning generations:	50 (x2 for the loosing evolver)
Constraints in Planner:	
Fuel:	Yes
Icing:	Yes
Target:	No
Altitude Scan:	No
Other:	None

Chapter 7

SUMMARY

This chapter serves as a summary of the research presented in this work, as well as proposed ideas of further research.

7.1 Summary of Work

The following features of the research have been presented:

- Evolutionary Computation techniques have been used for path planning of autonomous vehicles.
- The planning algorithm considers 4-dimensionality (3-D space and time). Both static (pre-plan) and dynamic (en-route) planning are performed.
- A spherical Earth model has been implemented in the path planner for long range applications.
- Vehicle maneuverability is considered to assure that path generation is limited to the vehicle's capabilities. This information, as well as the fuel consumption model, is provided to the planner with the use of performance tables which allows path planning for any type of vehicle.
- Weather information from standard GRIB format databases is fed into the planning system to consider realistic weather scenarios. Actual weather forecasts may be updated into the planner as they become available.
- Regions of potential icing conditions are computed by providing information from the weather forecasts to an Integrated Icing Forecast (based) Algorithm.

- Objectives for path generation include: optimizing fuel requirement, icing avoidance, site observation, altitude scanning.
- A preliminary implementation of the parallel evolution concept has been used.

7.2 Further Research

Areas of further work and research include:

- Methods to improve performance of the planning algorithms such as a more refined use of the parallel evolution concept.
- Research on fitness function models and user-friendly methods for problem objectives definition.
- Research on the integration of vehicle performance degradation models with the prediction of potential icing.
- Integration of path planning and guidance algorithms.
- Implementation of feedback from the vehicle's actual trajectory compared to the desired track for re-planning purposes.
- Hardware-in-the-loop and on-board UAV implementation of planning algorithms.

BIBLIOGRAPHY

[1] McGeer,T. and Vagners,J., “Historic crossing: an unmanned aircraft’s Atlantic flight,” *GPS World 10(2):24-30*, February 1999.

[2] Rubio,J.C. and Kragelund,S., “The Trans-Pacific crossing: Long range adaptive path planning for UAVs through variable wind fields,” in *Proceedings of the AIAA 22nd Digital Avionics Systems Conference*, (Indiana, IN), October 2003.

[3] Pongpunwattana,A., Rysdyk,R., Vagners,J., Rathbun,D., “Market-based co-evolution planning for multiple autonomous vehicles,” in *Proceedings of the AIAA Unmanned Unlimited Systems Conference*, September 2003.

[4] Fogel,D.B., *Evolutionary Computation: The Fossil Record*. New York: IEEE Press, 1998.

[5] Holland,J., *Adaptation in Natural and Artificial Systems*. PhD thesis, University of Michigan, 1998.

[6] Fogel,D. and Fogel,L., “Optimal routing of multiple autonomous underwater vehicles through evolutionary programming,” in *Symposium on Autonomous Underwater Vehicle Technology*, (Washington, D.C.), 1990.

[7] Xiao,J. et.al., “Adaptive evolutionary planner/navigator for mobile robots,” in *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 18–28, April 1997.

[8] Capozzi,B. and Vagners,J., “Evolving (semi)-autonomous vehicles,” in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, (Montreal, QC), 2001.

[9] Hocaoğlu,C. and Sanderson,A.C., “Planning multiple paths with evolutionary speciation,” in *IEEE Transactions on Evolutionary Computation*, pp. 169–191.

[10] Rathbun,D., and Capozzi,B.J., “Evolutionary approaches to path planning through uncertain environments,” in *Proceedings of the AIAA 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles, Systems, Technologies and Operations*, (Portsmouth, VA), May 2002.

[11] Fogel,D., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway: IEEE Press, 2nd ed., 2000.

[12] Capozzi,B.J., *Evolution-Based Path Planning and Management for Autonomous Vehicles*. PhD thesis, University of Washington, 2001.

[13] Rudolph,G., “Convergence of evolutionary algorithms in general search spaces,” in *Proceedings of the IEEE 3rd Conference on Evolutionary Computation*, (Piscataway, NJ), 1996.

[14] Rudolph,G., *Convergence Properties of Evolutionary Algorithms*. PhD thesis, Universitat Dortmund, 1996.

[15] Băck,T., *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[16] Alba,E. and Troya,J., “A survey of parallel distributed genetic algorithms,” in *Complexity*, vol. 4, pp. 31–52, 1999.

[17] Rudolph,G., “Parallel approaches to stochastic global optimization,” in *Parallel Computing: From Theory to Sound Practice* (W.Joosen and e. E. Milgrom, eds.), pp. 256–267, IOS Press, 1992.

[18] Branke,J. and Schmeck,H., “Designing evolutionary algorithms for dynamic optimization problems,” in *Advances in Evolutionary Computation* (A. Ghosh and e. S. Tsutsui, eds.), pp. 256–267, Springer, 2003.

[19] Nilsson, N., *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga Publishing Company, 1980.

[20] Stentz, A., “Optimal and efficient path planning for partially-known environments,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1994.

[21] Rathbun,D., Kragelund,S., Pongpunwattana,A., and Capozzi,B., “An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments,” in *Proceedings of the AIAA 21st Digital Avionics Systems Conference*, (Irvine, CA), October 2002.

[22] World Meteorological Organization, Geneva, *Guide to the WMO Table Driven Code Form Used for the Representation and Exchange of Regularly Spaced Data in Binary Form: FM 92 GRIB Edition 2*, January 2003.

[23] World Meteorological Organization, *FM 92 GRIB, Edition 2 - Version 2*, May 11 2003.

[24] Gordon,B., *The WMO Format for the Storage of Weather Product Information and the Exchange of Weather Product Messages in Gridded Binary Form as used by NCEP Central Oparations*. NCEP Central Operations, May 2002.

[25] Stackpole,J., *A Guide to GRIB: The WMO Format for the Storage of Weather Information and the Exchange of Weather Product Messages in Gridded Binary Form*. National Weather Service, February 1994.

[26] World Meteorological Organization, *(Part II) A Guide to the Code Form FM 92-IX Ext. GRIB, Edition 1*.

[27] “Year end review.” National Centers for Environmental Prediction, 2002.

[28] Ok,H., and Eberhardt,S., “Aircraft icing predictions using an efficient, incompressible navier-stokes solver,” in *Proceedings of the AIAA 32nd Aerospace Sciences Meeting*, (Reno, NV), January 1994.

[29] Wright,W., “A summary of validation results for LEWICE 2.0,” in *Proceedings of the AIAA 37th Aerospace Sciences Meeting and Exhibit*, (Reno, NV), January 1999.

[30] Habashi,W. et.al., “FENSAP-ICE: A second generation CFD-based, fully 3d in-flight icing simulation system,” in *FAA In-Flight/Ground De-icing International Conference*, (Chicago, IL), June 2003.

[31] McCann,D., “Percent power increase-a simple way to quantify an icing hazard,” in *Proceedings of the 9th Conference on Aviation, Range, and Aerospace Meteorology*, September 2000.

[32] Ratvasky,T., “Icing effects on aircraft performance, stability and control, and handling qualities,” in *FAA In-Flight/Ground De-icing International Conference*, (Chicago, IL), June 2003.

[33] Hossain,K., Sharma,V., Bragg,M. and Voulgaris,P., “Envelope protection and control adaptation in icing encounters,” in *Proceedings of the AIAA 41st Aerospace Sciences Meeting and Exhibit*, (Reno, NV), January 2003.

[34] “About current icing potential (CIP).” Aviation Weather Center.

[35] Politovich,M. and Bernstein,B., “CIP: What it is and where it’s going,” in *Proceedings of the FAA International In-flight Icing/Ground De-Icing Conference*, (Chicago, IL), June 2003.

[36] McDonough,F., Bernstein,B.C., Politovich,M.K., and Wolff,C.A., “The forecast icing potential (FIP) algorithm,” in *Proceedings of the AMS 20th International Conference on Interactive Information and Porcessing Systems (IIPS) for Meteorology, Oceanography and Hydrology*, (Seattle, WA), January 2004.

[37] Brown,B.G., Mahoney,J.L., Bullock,R., Fowler,T.L., Henderson,J., and Loughe,A., “Quality assessment report: Integrated icing diagnostic algorithm (IIDA),” tech. rep., Quality Assessment Group, Aviation Forecast and Quality Assessment Product Development Team, July 2001.

[38] Brown,B.G., Fowler,T.L, Mahoney,J.L., “Verification results for the integrated icing forecast algorithm (IIFA),” tech. rep., Quality Assessment Group, Aviation Forecast and Quality Assessment Product Development Team, July 2001.

[39] Brown,B.G., Fowler,T.L, Mahoney,J.L., Chapman,M., Bullock,R., Henderson,J., "Forecast icing potential (FIP): Quality assessment report," tech. rep., Quality Assessment Product Development Team, Aviation Weather Research Program, April 2003.

[40] Benjamin,S. et.al., "RUC-2 - the rapid update cycle version 2." NWS Technical Procedures Bulletin 448.

[41] Pongpunwattana,A., "Distributed real-time mission planning for autonomous vehicles using evolutionary computation." PhD Research Proposal, University of Washington, 2003.

[42] Lyman,E. and Goddard, E., *Plane and Spherical Trigonometry*. Boston: Allyn and Bacon, 1900.

[43] Kells,L., Kern,W. and Bland,J., *Plane and Spherical Trigonometry*. New York: McGraw-Hill Book Co., 3rd ed., 1951.

Appendix A

NAPIER'S RULES FOR SPHERICAL RECTANGULAR TRIANGLES

Ref:[42], [43].

Consider the right angle spherical triangle of figure A.1. Angles A, B and C are the angles formed by crossing sides, and angles a, b, and c are the angles formed by the sides of the triangle, at the center of the sphere. To understand Napier's rules, each angle of the triangle is arranged in the same sequence (excluding the known right angle C) as shown in the circle on the right of the figure, and taking the complement of those angles not in touch with angle C in the triangle. These five angles are called the *circular parts* of the triangle. Looking at the *parts* of the arranged circle, the two parts that are to each side of any selected part (called *middle part* are referred to as *adjacent parts*; the two parts in front of a middle part are called *opposite parts*. With this being said, Napier's rules are stated as follows:

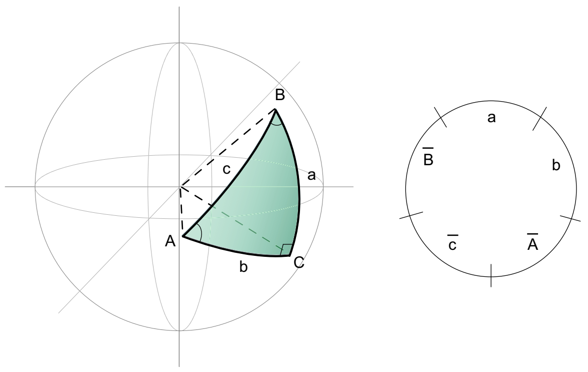

Figure A.1: Rectangular Spherical Triangle.

Rule 1. The sine of a part is equal to the product of the cosines of the two opposite parts. Example:

$$\sin \overline{c} = \cos a \ \cos b$$

Rule 2. The sine of a part is equal to the product of the tangents of the two adjacent parts. Example

$$\sin b = \tan a \ \tan \overline{A}$$