NASA/TM–2014-218662

Towards a Formal Semantics of Flight Plans and Trajectories

George E. Hagen and Ricky W. Butler
Langley Research Center, Hampton, Virginia

December 2014

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–2014-218662

Towards a Formal Semantics of Flight Plans and Trajectories

George E. Hagen and Ricky W. Butler
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

December 2014

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

In the National Airspace System, flight plans are often used only as a planning tool by air traffic controllers and aircraft operators. These plans are implicitly translated into trajectories by the pilot or by the flight management system, and subsequently flown by the aircraft. This translation process inevitably introduces differences between the plan and the trajectory. However, given the current intended usage, exact correspondence between the plan and the trajectory is not needed. To achieve greater capacity and efficiency, future air traffic management concepts are being designed around the use of trajectories where predictability is extremely important. In this paper, a mathematical relationship between flight plans and trajectories is explored with the goal of making feasible, highly accurate predictions of future positions and velocities of aircraft. The goal here is to describe, in mathematically precise detail, a formal language of trajectories, whereby all receivers of the trajectory information will be able to arrive at precisely the same trajectory predication and to do this without having aircraft broadcast a large amount of data. Although even a four-dimensional flight plan is simple in structure, this paper will show that it is inherently ambiguous and will explore these issues in detail. In effect, we propose that a rigorous semantics for flight plans can be developed and this will serve as an important stepping stone towards trajectory-based operations in the National Airspace System.

Contents

1 Introduction

A number of advanced air traffic management concepts, often referred to as Trajectory Based Operations, require not only knowing the current position of aircraft is to a high degree of accuracy, but also where it will be in the future [4,8]. This information may be used directly, such as with de-conflicting aircraft trajectories, or it may be used indirectly as part of a procedural concept such as aircraft spacing in the terminal area.

In any event, there are many uses for knowing an aircraft's intended future trajectory, and no matter what party generates or uses this information, it will often be necessary to transfer this information to other parties. If an aircraft is intended to determine its own future trajectory, it will be necessary to let other aircraft and controllers know what this trajectory will be. If future trajectories are calculated by remote authorities, it will be necessary to at least let an aircraft know what it is expected to do. This transfer of data is especially important if there is some unexpected change partway through an aircraft's flight, due to an emergency situation, unexpected weather, or routine traffic management. In all of these situations it is necessary that the parties involved use this data to achieve consistent results.

It is not obvious exactly what information should be transmitted from an aircraft. At first, it might appear that the broadcast of the flight plan filed with the Federal Aviation Administration (FAA) would be sufficient. A flight plan, however, is not a precise description of where the aircraft will be at a particular time, but rather a set of constraints on the behavior of an aircraft. Typically these constraints are fairly loose: a set of horizontal waypoints that should be passed over or closely encountered in a given sequence. These are often augmented with certain altitude and speed restrictions (even as vague as "above 15000 feet"), and estimated takeoff times and arrival times.

Unfortunately, this only provides a rough approximation of where an aircraft will be in the future. This is good enough for basic planning, when it is anticipated that there will be little traffic nearby, or when ad hoc corrections will be subsequently allowed to ensure proper management of aircraft, but this approach severely limits potential performance and safety gains from advanced automation that will require highly accurate and consistent data.

There are deeper issues than a lack of precise information. Even if all horizontal points in a flight plan include both altitudes and times (i.e., a four dimensional flight plan), such a plan may not be flyable if it contains significant instantaneous velocity changes. Basic inertia makes it impossible to precisely follow such a flight plan, and there are inherent ambiguities as to how these plans are to be transformed into actual maneuvers. Should turns occur before waypoints or after they have been crossed over? What radius should those turns be? Are times in these plans Estimated Times of Arrival (ETA) or

Required Times of Arrival (RTA)? What speeds should be achieved between points, especially in the presence of turns that necessarily alter the distances involved?

As others have recognized, a more direct solution to this problem is to transmit trajectories instead of flight plans. Although the concept of a trajectory is simple, i.e., a function that assigns a position and velocity to a given time, the estimation of a trajectory is not an easy task. Delahaye, et. al. provide a comprehensive study of mathematical approaches to calculating trajectories [2]. A trajectory that perfectly represents the flight path of an aircraft is a function of the dynamics of the particular aircraft and also of a large number of variables that can only be approximated. What is the air pressure and wind behavior throughout a maneuver? How much fuel will have been consumed at any given point? What is the precise mass distribution within the aircraft, considering fuel burn and possible fuel transfer? These hard-to-determine parameters are among the reasons why predicting a simple climb profile can be extremely difficult.

Most control systems rely on feedback loops to adjust their behaviors. Therefore most papers that discuss trajectory generation present a method based on the dynamics of an aircraft and a feedback control system. But once the trajectory is computed, the result is a large sequence of positions as a function of times (usually with a time step of about 1 second). Broadcasting this entire trajectory would be inefficient so alternative approaches are of interest. The RTCA DO-242A document [7] describes the communication of trajectories using Trajectory Change Points (TCPs). The TCPs have been defined to accommodate everything from high precision trajectories generated by a FMS, to a very poorly defined turn-before waypoint used for an aircraft with no automation. The work by FAA/Eurocontrol Cooperative Research Action Plan 16 provides future directions for trajectory prediction [5, 9, 10].

There is another key problem that must be addressed. If future concepts of operation rely on knowing where an aircraft will be to a moderate degree of accuracy, how can this information be generated and transmitted? This paper presents one possible approach to this problem, emphasizing a means of generating approximations of trajectories that are general, flyable, simple to transmit, and simple enough to allow the mathematical proof of important safety properties. We propose the use of an approximation of a trajectory that has a well-defined semantics and relies on simple kinematic formulas involving constant accelerations. The resulting kinematic trajectories consist of constant velocity segments and constant acceleration segments where the resulting position and velocity functions of time are continuous. In the process of developing a method from translating from a 4D plan (i.e. a sequence of 4D waypoints) to a kinematic trajectory we have identified many ambiguities inherent in such a flight plan. In this paper we will present these ambiguities

and offer resolutions to these ambiguities.

We note the control system approach avoids this problem by delivering the full trajectory that a particular control system flies in the presence of the flight plan [3]. A key disadvantage of the control system approach is that different systems can produce different trajectories. In an airspace concept where the trajectory is only a loose approximation of the flight plan, differences in trajectories are unimportant. However, in concepts where air and ground systems rely on accurate trajectories, it is important that the receiver is able to precisely reconstruct the intended trajectory using only the information broadcast. If the receiver's trajectory prediction is different from the sender's, then potential safety issues may result. This is analogous to the language semantics problem in computer science where programs compiled by different compilers produce different outputs for the same program. This can result in unpredictable code that cannot be easily ported from one environment to another. In the air transportation case, this means that trajectories produced on one set of equipment may not necessarily match those produced on another set of equipment, even given identical source data. Perhaps this problem can be mitigated by a tighter *semantics* for flight plans that prescribe a mathematically precise trajectory.

We believe that this approach will also greatly facilitate simulation studies of the National Airspace System. We hypothesize that simulation software need only generate the range of trajectories produced by a suite of aircraft and that the exact reproduction of the trajectory of any particular aircraft is not necessary. If this hypothesis can be shown to be true (in future work), then effective simulation can be accomplished parametrically without having to model the dynamics of individual aircraft and without computing control laws for them. The advantage of such an approach is obvious – the computation time is greatly reduced, and thus fast-time simulation is enabled. This was the approach used in the Flexible Airspace Modeling Environment (FLAME) simulator which "simulates the movements of aircraft under the assumption that all flights exactly follow their flight plans" [1]. The FLAME simulator has been used on at least 5 major research projects. Directly following a flight plan is advantageous because it is very efficiently computed, but it has the disadvantage that this produces a discontinuous velocity vector over time (e.g. the instantaneous change of heading at a vertex in the flight plan graph).

In this paper we present the concept of a kinematic trajectory and an algorithm to generate a kinematic trajectory from a flight plan. This algorithm has been broken into six incremental steps (passes) that, taken together, will generate a well-defined trajectory, one that can be efficiently computed and still produce a continuous velocity vector over time, a property that we believe is central to a flyable trajectory. It is surprisingly subtle and there were many failed attempts before an adequate algorithm was discovered.

2 Kinematic Trajectories

There are many different notions of "flight plan":

- The flight plan that is officially filed with the FAA
- The flight plan that is loaded into a Flight Management System
- The flight plan used by controllers in an en-route sector

Consequently, a flight plan can come in many different varieties:

1. A sequence of 2D waypoints (horizontal positions only)
2. A sequence of 2D waypoints augmented with constraints on altitude and cruise speed
3. A sequence of 3D waypoints + a nominal ground speed
4. A sequence of 4D waypoints (fourth dimension is time)

The National Airspace System today is largely driven by the second item, though only the last one has enough information to enable conflict detection with traffic aircraft. Even the last option is not ideal because the sequence of 4D waypoints results in discontinuous velocity vectors, which airplanes cannot follow precisely. Also the velocity profile between the 4D points is not specified in the last option. If we assume that the velocity is constant between the 4D-waypoints, we have greater precision. We refer to such 4D-plans as *linear plans* because there is no acceleration within the segments. In a linear plan, all velocity changes occur instantaneously at the points between two segments. However, we can construct a slightly more realistic plan that includes acceleration zones in addition to the linear segments of a traditional flight plan. This is illustrated in Figure 1. The orange waypoints represent turn Trajectory Change Points (TCPs) and define the turn acceleration[1] The magenta waypoints are ground speed TCPs and define a ground speed acceleration. Interestingly, this trajectory can be completely defined by listing this sequence of 4D waypoints. To reconstruct a kinematic plan from the TCPs, they generally need only include two additional pieces of information: the type of acceleration (i.e., turn, ground speed, or vertical speed) and the acceleration rate. Therefore, they can be very efficiently broadcast and they can be generated from a detailed 4D linear plan (i.e. a sequence of 4D waypoints). However, this process is non-trivial and involves several compromises and trade-offs. In this paper we discuss this process of generating a kinematic plan from a 4D-plan. If this process can be formally defined one has, in effect, created a formal semantics for 4D linear "flight plans".

[1] We note that DO242A defines trajectory change points differently.

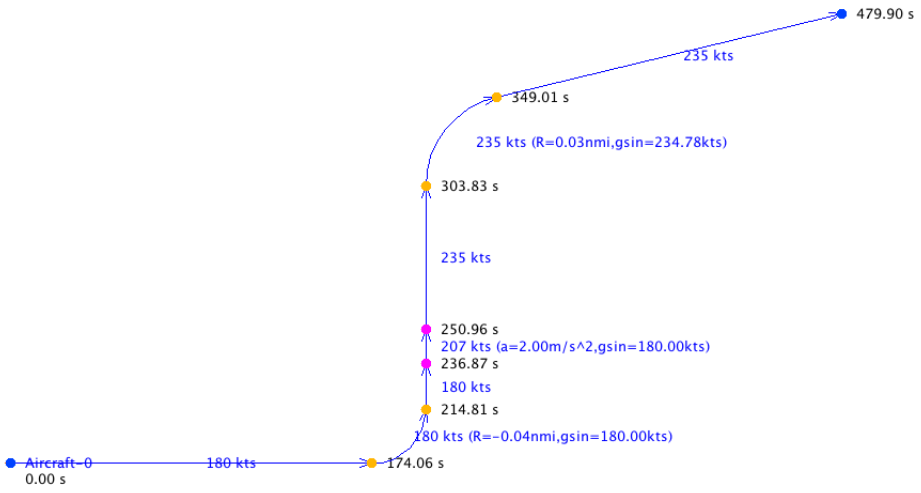

Figure 1: Example Kinematic Plan

2.1 Well-Formed Kinematic Plans

A kinematic plan is a time sequence of 3D points where some of these points are labeled as Trajectory Change Points (TCPs). We introduce a simple notation:

- $\mathbf{s_i}$ is the 3-dimensional position of the ith waypoint in the plan
- t_i is the time of the ith waypoint in the plan
- plan *segment* i is interval between the ith and ith+1 waypoints in the plan
- $\mathbf{v_i}$ is the 3-dimensional velocity associated with the ith waypoint in the plan (and, if not in an acceleration zone, the ith segment)

In a kinematic plan we introduce the following types of TCPs:

`BOT`	beginning of turn (of less than 180 degrees)
`EOT`	end of turn
`BGS`	beginning of ground speed acceleration
`EGS`	end of ground speed acceleration
`BVS`	beginning of vertical speed acceleration
`EVS`	end of vertical speed acceleration

Each acceleration zone is delimited by a "beginning" and "end" TCP of the appropriate acceleration type. The "beginning" point provides the initial state of the acceleration zone, including the exact acceleration value. The "end"

point provides the duration of acceleration, as well as a position reference for checking the acceleration calculations. Additionally, optional interpolated "mid point" TCPs may be included. These can be especially useful in representing turns, and a "middle of turn" (`MOT`) point appears in several diagrams in this paper.

A kinematic plan is *well-formed* if every beginning TCP has a corresponding ending TCP. For example, a beginning of turn TCP (`BOT`) is followed by a corresponding end of turn TCP (`EOT`) and turns are not overlapping. This must also be true of beginning and end of ground speed changes (`BGS/EGS`) and beginning and end of vertical speed changes (`BVS/EVS`). A vertical acceleration segment can overlap a turn segment or a ground speed segment, but a turn segment cannot overlap a ground speed acceleration segment. We also do not allow segments of the same type to overlap. For example, a vertical acceleration segment must end before another one starts. These restrictions greatly simplify the calculations involved.

2.2 Position and Velocity in a Kinematic Plan

The position and velocity vectors are easily computed for all segments that are not acceleration zones. This includes any segments that begin with a non-TCP point or segments that are not between a TCP beginning point and its corresponding end point. In this case, the velocity of this segment is computed as follows

$$\mathbf{v}_{i-1} = \frac{\mathbf{s}_i - \mathbf{s}_{i-1}}{t_i - t_{i-1}}$$

The position $s_i(t)$ at absolute time t within segment i is given as follows

$$\mathbf{s}_i(t) = \mathbf{s}_i + \mathbf{v}_i\,(T_i - t)$$

where T_i is the time of waypoint i.

Horizontal (turn and ground speed) and vertical acceleration calculations are orthogonal to each other.

2.2.1 Turn Segments

In a segment i that begins with a `BOT`, the velocity vector is not constant, but the turn rate ω (i.e. the acceleration) is assumed to be constant and the magnitude of the velocity v is constant. Therefore, the horizontal components of the velocity vector are defined as follows:

$$v_x(t) = v\sin(\theta + \omega t)$$
$$v_y(t) = v\cos(\theta + \omega t)$$

where θ is the initial track[2] of the aircraft and t is the relative time since the beginning of the segment. Consequently the horizontal components of the position throughout the turn are given by

$$s_x(t) = s_x(0) + \frac{v}{\omega}[\cos\theta - \cos(\theta + \omega t)]$$
$$s_y(t) = s_y(0) - \frac{v}{\omega}[\sin\theta - \sin(\theta + \omega t)]$$

where $\mathbf{s} = (s_x(0), s_y(0))$ is the initial position and θ is the track of the velocity vector at the beginning of the segment. The radius of the turn is given by $R = \frac{v}{\omega}$. We can define a function `turn` as follows:

$$\texttt{turn}(\mathbf{s}(0), \mathbf{v}(0), \omega)(t) = (s_x(t), s_y(t))$$

Note that velocity at the beginning of the segment, $\mathbf{v}(0)$, has track θ and magnitude v.

2.2.2 Ground Speed Segments

We want to compute the position and velocity within a segment that begins with a `BGS`. We let s_0 represent the position at the beginning of the segment and v_0 the horizontal velocity at the end of the previous segment. The direction of the velocity vector is not going to change, so we use the unit vector $\hat{\mathbf{v}}_0$. The horizontal component of the velocity, $\mathbf{v}$, at time t is:

$$\mathbf{v}(t) = (|v_0| + at)\hat{\mathbf{v}}_0$$

where t is the relative time since the beginning of the segment and a is the ground speed acceleration. The position at relative time t within the acceleration segment is:

$$\mathbf{s}(\mathbf{t}) = \mathbf{s}(0) + (|v_0|t + \frac{1}{2}at^2)\hat{\mathbf{v}}_0$$

2.2.3 Vertical Speed Segments

The horizontal components during a vertical acceleration can be computed using the simple linear formulas above. The vertical component of the velocity, $\mathbf{v}_z$, is calculated as follows:

$$v_z(t) = v_{0z} + at$$

where t is the relative time since the beginning of the segment, a is the vertical speed acceleration, and v_{0z} is the vertical component of the velocity at the end of the previous segment. The position at relative time t within the acceleration segment is:

$$s_z(t) = s_{0z} + v_{0z}t + \frac{1}{2}at^2$$

[2]measured clockwise from due north.

2.3 Consistent Kinematic Plans

There is a precise relationship between the locations and times of `BOT/EOT` pairs and the associated acceleration (i.e. turn rate or angular velocity). This is also true of the `BGS/EGS` and `BVS/EVS` pairs. If t is the duration of acceleration segment, then the position at relative time t must exactly correspond to the above formulas at time t. For example, if s_{BOT} is the position of the `BOT`, then the location of the `EOT`, s_{EOT}, must satisfy

$$s_{EOT} = \texttt{turn}(s_{BOT}, v_{BOT}, \omega)(t)$$

Also, the velocity at the end of the turn should be equal to the velocity at the beginning of the next segment. In other words, the velocity is continuous. We call this correspondence *consistency*.

For a ground speed acceleration segment, the total distance covered must be equal to $|v_0|t + \frac{1}{2}at^2$ and the final speed at the end of the segment must be $|v_0| + at$ and match the initial speed at the next segment. For a vertical speed acceleration segment, the total vertical distance covered must be $v_{0z} + \frac{1}{2}at^2$ and the final speed must be $v_{0z} + at$ and match the initial vertical speed of the next segment.

2.4 Prescriptive versus Descriptive Trajectories

As we discussed earlier, a trajectory is usually defined as a time-sequence of 3D positions that are generated using a dynamics model and a control system designed to follow the flight plan. Such a trajectory is *descriptive* of a particular aircraft and involves a large amount of data that may not be conveniently broadcast. The kinematic plan we presented in the above subsections is a *prescriptive* trajectory. It does not seek to define the behavior of any particular aircraft. It seeks to define an ideal trajectory that an airplane could closely follow. It is also very convenient to broadcast.

Strategic airborne conflict detection and resolution (CD&R) seeks to resolve future conflicts based on intent information that is broadcast by traffic aircraft. This intent information must be communicated in a way that enables a reasonably accurate reconstruction of the trajectory that the traffic aircraft will actually fly. There is a tradeoff between using simpler specification mechanisms for the intent and constraining aircraft to follow these specifications (prescriptive) versus using more complex mechanisms that give aircraft more flexibility in how they fly but come at the cost of more information that must be broadcast (descriptive). Today, an aircraft roughly flies in accordance with a flight plan, which can be altered by new vectors issued by a controller. Unfortunately, as discussed earlier in this paper, even an unaltered flight plan does not completely determine the trajectory of an aircraft. It merely places constraints on this trajectory. We hypothesize that the real trajectories flown by

aircraft can be approximated using a number of linear segments and constant acceleration segments (or, if bandwidth is not an issue, a sufficient number of purely linear segments). We note that if an aircraft flies a trajectory that deviates significantly from this constant acceleration assumption, then the accuracy can be increased by adding more segments with gradually changing accelerations (i.e. with less time between them). A key research question is how prescriptive should the communication mechanism be? Although we do not seek to answer that question in this paper, we suspect that the best approach will be more prescriptive than DO-242. This is in the spirit of Required Navigation Performance (RNP). A good approach may be to use some mechanism to bound the error between the communicated trajectory and the actual flown trajectory.

We also choose to utilize ground speed instead of airspeed to have a common frame of reference between aircraft that are possibly large distances away. These trajectories are intended to be shared with other parties as useful common knowledge. It is assumed that a modern (or future), GPS-enabled flight control system would be able to compensate for most nominal local wind conditions[3]. If this is not the case, revised trajectories may be broadcast that more closely represent an aircraft's intended actions.

3 Generation of Kinematic Plans from Linear Plans

3.1 Notation and Data Structures

We will work with the following basic data structures. This presentation will use Euclidean space, possibly projected from geodetic coordinates.

`Position` : A `Position` is a triple (x,y,z) representing a spatial location

`Velocity` : A `Velocity` is a triple representing a velocity vector. Velocities can be interpreted as having track, ground speed, and vertical speed components. They can also be interpreted as Euclidean 3-space vectors.

`NavPoint` : A `NavPoint` represents a 4-D point in space and time. It includes at least a `Position` and a time, possibly with additional data.

`Trajectory Change Point (TCP)` : A `TCP` is a `NavPoint` that can include some supplementary data, generally used to indicate the beginning or end of an acceleration zone. This will generally include a TCP type field

[3]If the originator of the trajectory in question has sufficient knowledge of wind conditions, ground speeds may be calculated so to result in the desired air speed for a given segment.

and an acceleration or angular velocity field. The TCP type fields may be `NONE, BOT, EOT, BGS, EGS, BVS`, and `EVS`.

`Plan` : A `Plan` is an abstraction of a 4-flight plan or a trajectory.

We say that a plan is *linear* if it does not contain any TCPs, and *kinematic* if it does. A kinematic plan is expected to be both well-formed and consistent. These properties are necessary requirements for a trajectory to be *flyable*, but not sufficient in themselves.

3.2 Tradeoffs

There are many inherent ambiguities in a flight plan, even a highly constrained 4-D flight plan, and not all aspects of the original linear plan can be preserved. This is not obvious until one tries to generate a trajectory from a flight plan—then it becomes painfully clear. Our algorithm prioritizes preserving the ground speeds of the original linear plan over the times of individual points (including the end point). Therefore it interprets the linear plan as a series of positions and implicit ground speeds (times are ETAs) rather than positions and absolute times (RTAs). In the vertical dimension, it prioritizes the altitudes of points where the vertical speed changes significantly, as opposed to specific vertical speeds. This allows one to more easily create a plan consistent with parameters such as "climb to 20,000 feet and then level off and proceed to waypoint at 300 knots."

Variations on this algorithm, possibly involving pre-processing, can prioritize other aspects of the original linear plan, such as RTAs.

Please note that while air speed is generally used in current flight systems, it is an inherently relative measurement based on local conditions, and as such not terribly useful for precise, long-term—or possibly even short-term—predictions, especially for a nonlocal third party. In these algorithms we instead choose to prioritize ground speeds, which at least have a consistent frame of reference. In short: air speed may be very useful for the act of flying the aircraft, but it is considerably less useful when describing the aircraft's predicted positions to someone else. We anticipate that future control systems will be able to match ground speeds under most reasonable conditions. If local conditions do make achieving a planned ground speed infeasible, the trajectory can always be updated to reflect these new constraints.

3.3 Overview of the Generation Algorithm

In this section we present an overview of the `generateTCPs` function that translates a linear plan into a kinematic plan. A detailed description is provided in section 3.6. The generation algorithm uses multiple passes in order

to decompose the problem in a manner suitable for a future formal analysis. It first processes turns, then ground speed changes, and then vertical speed changes. The instantaneous transitions of the linear plan are converted into constant acceleration segments that are defined by TCPs. In other words, we want to transform a plan that does not have a continuous velocity into one that does. But as mentioned above, it is not clear exactly what a linear plan means—it is incredibly ambiguous. We are attempting to develop a reasonable interpretation that is both simple and useful.

In the first pass of the translation, instantaneous turns are replaced by turn TCPs which introduce a circular arc path. This circular arc path is shorter than the path in the linear plan because it turns before the vertex point. This causes the first issue. Consider the linear plan in Figure 2. The linear plan

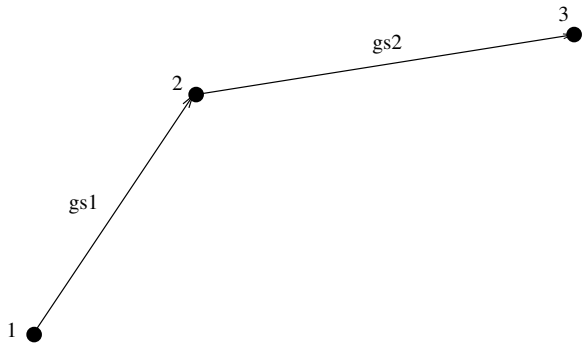

Figure 2: Simple Linear Plan

consists of 3 waypoints that contain a 3D position and a time; however, the figure only shows the horizontal view. The times at the waypoints implicitly define the ground speed on each of these legs that are easily calculated because in a linear plan there are no accelerations. We have labeled the edges with $gs1$ and $gs2$ that represent these ground speeds.

An aircraft cannot make an instantaneous turn at point 2. Instead, a turn must be made somewhere around point 2 and end up on the second leg. It is possible to turn before waypoint 2 or try and go through waypoint 2. In the current algorithm, we are seeking solutions where the aircraft turns before waypoint 2. In other words, we want to find the points where an inscribed circle is tangent to both the legs 1-2 and 2-3. In Figure 3, these points are colored green and labeled BOT for beginning of turn and EOT for end of turn. (In Figures 3 and 4 we have added an additional middle of turn (`MOT`) point to more easily differentiate the linear and kinematic paths in the discussion.)

Note that the circular arc path `BOT-MOT-EOT` is shorter than the linear path `BOT-2-EOT`. If ground speed is kept constant through the turn and subsequent points are unchanged, this implicitly redefines the ground speed following the

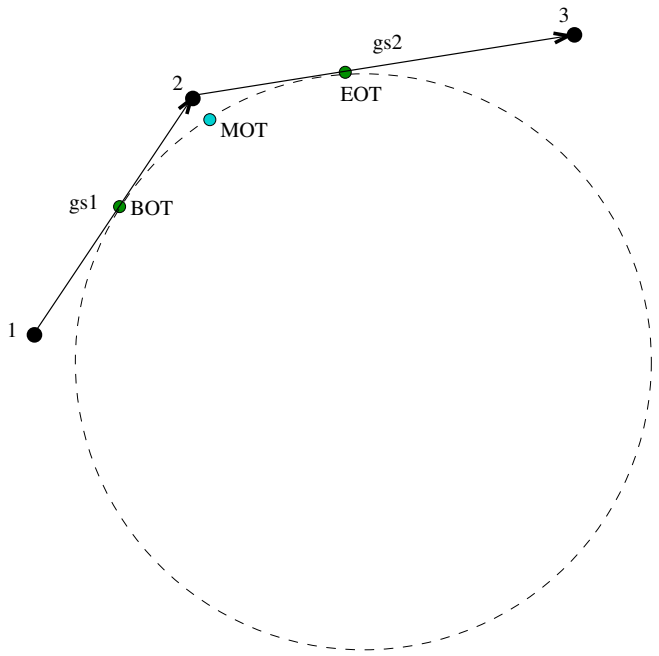

Figure 3: Kinematic Plan Development from Linear Plan (Turn Generation)

turn. Therefore it is impossible to preserve both the original ground speeds and the original point times when converting from a linear plan to a kinematic plan. We have decided to prioritize ground speeds in our conversions. This means that the times of the waypoints will change.

We can repair the ground speed into point 3 by changing the time, but what do we do if the ground speed on the second leg was different from the first leg in the linear plan? We can address this scenario by performing a ground speed acceleration or deceleration after the turn is complete. This is illustrated in Figure 4, The pink circles are two waypoints that indicate where the ground speed acceleration begins and ends. The turn generation is made under the assumption that the ground speed is constant throughout the turn and so we must be careful to delay the BGS waypoint until after the turn is complete. In Figure 4, the ground speed is $gs1$ from point 1 to BGS and the ground speed is $gs2$ from EGS to point 3.

Naturally, ground speed accelerations may also be necessary on collinear segments, such as the one illustrated in Figure 5.

The linear plan also specifies a vertical profile that describes a desired set of altitudes (Figure 6). Each of the dots in the figure represents a 4-D waypoint that contains the horizontal position, altitude, and time. There are two different ways of interpreting this vertical profile:

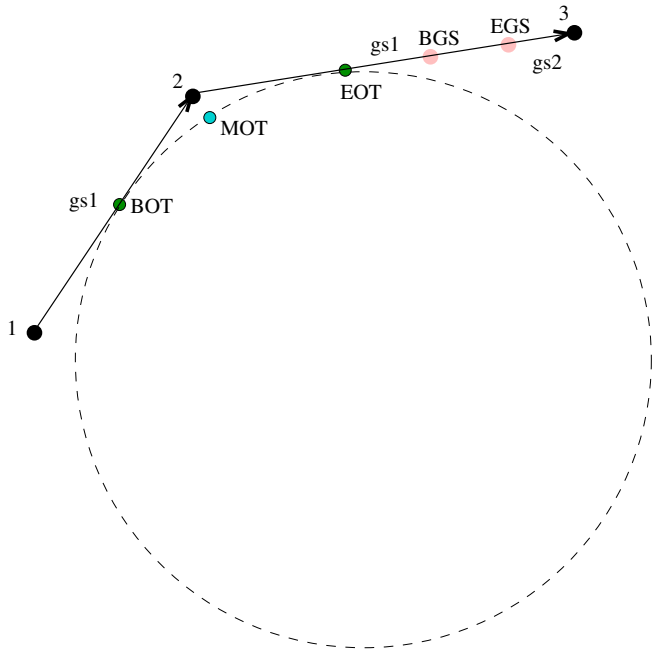

Figure 4: Kinematic Plan Development from Linear Plan (Turn Generation with Ground Speed Adjustment)

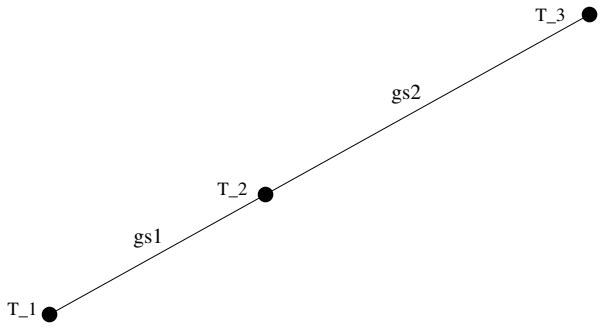

Figure 5: Simple Linear Plan

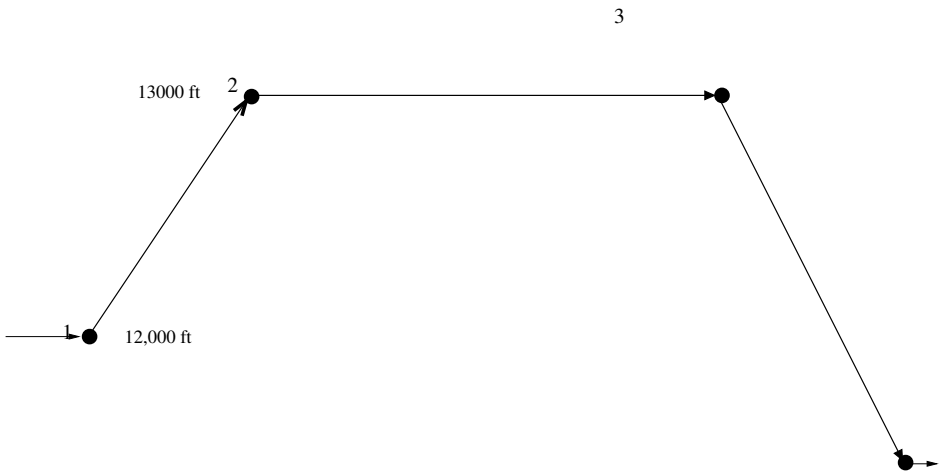

Figure 6: Simple Linear Plan

- The altitudes are a function of time.
- The altitudes are a function of the horizontal position, that is, the altitude changes at specified locations.

This distinction is more important than one might expect. We have already shown that the times of the waypoints change as ground speed changes. If the ground speed is varied do we want the location of the altitude changes to move? If not then one must think of the altitudes as a function of position and not time.

3.4 Infeasible 4D Plans

There are many cases where it is not possible to generate a kinematic plan from the linear plan given limits on the rate of acceleration. We assume that there are three parameters that specify the maximum accelerations:

`maxBankAngle`	maximum bank angle used for the turn
`maxGsAccel`	maximum ground speed acceleration
`maxVsAccel`	maximum vertical speed acceleration

We note that the maximum bank angle (ϕ) is related to the angular velocity (ω) of the turn once the aircraft speed is given, via the following calculation:

$$\omega = \frac{g \tan \phi}{v}$$

where v is the ground speed of the aircraft throughout the turn. Note that this also determines the radius R of the turn:

$$R = \frac{v}{\omega} = \frac{v^2}{g \tan \phi}$$

As we choose to store ω instead of ϕ in the TCP information, direction information is encoded in the sign: a positive value indicates a right or clockwise turn, while a negative value indicates a left or counterclockwise turn.

3.5 Implementation Limitations

Our software implementation of this algorithm has some inherent limitations on points that can be included in plans, the most significant being that points cannot overlap. This is primarily intended to filter out numerical instabilities and spurious data that could result in anomalies such as plan segments with zero velocity or brief velocity spikes. Two points overlap if they are very close both in space and time. "Very close" is determined by a set of minimum distances and time units that are allowed. The default values are tied to GPS specification limitations, though they can be modified by the user.

In order to keep the mathematics of the algorithm simple, we do not allow plans to including overlapping horizontal acceleration zones or overlapping vertical acceleration zones. Turns and ground speed changes may not overlap and vertical speed changes may not overlap. However it is possible to have vertical acceleration zones to overlap with horizontal acceleration zones.

Additionally, in order to simplify the metadata stored in TCP points and reduce the amount of data needed to be transmitted, a TCP point may only indicate the start or end of a single acceleration type. This means there will be a slight delay between any two maneuvers—the aircraft may start a climb and then start a turn, for example, as opposed to doing both simultaneously.

As a side note, these limitations mean that in the implementation some very brief acceleration zones will not be included in the final plan—these may be indicated with special marker points that indicate that a small velocity discontinuity is expected and a minor course correction will be needed.

For simplicity's sake, these limitations are not directly addressed in the following discussion.

3.6 The Trajectory Generation Algorithm

The current algorithm to generate a kinematic plan from a linear plan is multi-pass. There are six passes:

1. Mark vertical speed change points
2. Turn generation pass
3. Ground speed correction pass
4. Speed acceleration generation pass
5. Constant vertical speed pass
6. Vertical acceleration generation pass

Accelerations are assumed to be predefined for plan generation, and should represent a set of nominal maximal values. Actual accelerations used in the final kinematic plan will be noted as TCP metadata and may be of a smaller magnitude than the specified ones. Any repairs to the plan (see Appendix B) are performed before the generation algorithm is initiated.

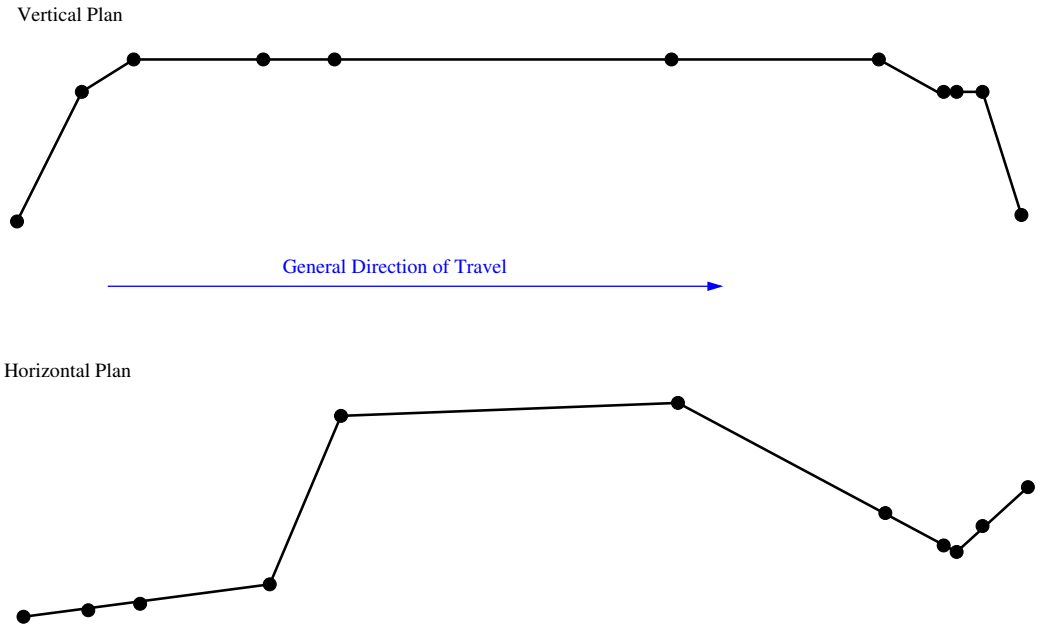

Figure 7: Example Linear Plan, Vertical and Horizontal Components

In the discussions that follow, please refer to the linear plan shown in Figure 7. Assume that this plan has times indicating a constant ground speed throughout.

Note this is a linear 4-D plan, with points indicating positions (landmarks, beacons, or other coordinates) to be flown over and the desired altitudes at those points. The time of the first point should reflect the start time for the trajectory, but otherwise timing is assumed to imply the desired ground speed for each segment, possibly adjusted for winds to reflect the desired airspeed, if this information is available. Similarly, the altitude information implies the approximate desired beginning and end of climbs and descents.

3.6.1 Mark Vertical Speed Change Points

The first pass of the algorithm begins by copying the original linear plan into the working kinematic plan. In this working plan, each point where there is a significant vertical speed change[4] is marked with a flag that designates the altitude of this point as *altPreserved*. In one of the last generation passes, the vertical speed profile of the evolving kinematic plan is smoothed. This is necessary because previous ground speed changes often result in a vertical profile with many undesirable vertical speed changes. This later pass of the generation algorithm will alter the altitude of many of the points to create a smoother vertical profile, but it will not change the altitude of any point that has been marked *altPreserved*. This is illustrated in Figure 8.

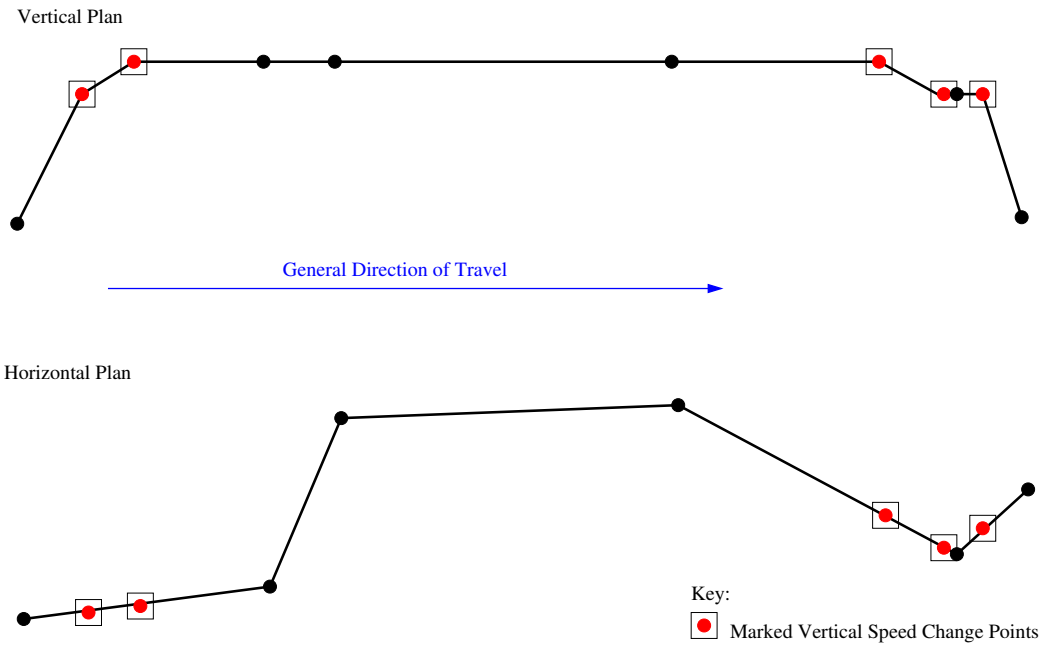

Figure 8: Example Linear Plan with Marked Vertical Speed Change Points

[4]For our implementation, this defaults to at least one second of vertical acceleration.

The next three passes will only alter the horizontal positions of points.

3.6.2 Turn Generation Pass

The second pass of the generation algorithm constructs kinematic turns. There are two significantly different ways to interpret a vertex (i.e. a turn point) in the horizontal perspective of a flight plan:

- The aircraft should begin the turn before the vertex point at precisely the right time so that when the turn is complete, the aircraft is pointed directly towards the next waypoint along the original track ("fly-by, track-to-fix").
- The aircraft should fly over the vertex waypoint, then begin the turn, heading directly to the next point ("fly-over, direct to fix").

A third option, "fly-over, track-to-fix" combines the notions above, but results in larger corrections and longer actual flight paths, and is not discussed in this paper.

Fly-by, Track-to-Fix Turns: Using the first approach, an aircraft will enter the turn on the track leading to the turn point (i.e. the vertex point) and leave the turn with the track leading away from the turn point. The aircraft will not actually pass over the turn point. This results in a shorter overall flight path.

The beginning of turn (`BOT`) and end of turn (`EOT`) points are calculated using Euclidean geometry. Given 3 `NavPoints` $np1$, $np2$, and $np3$ and a turn radius R (or equivalently a speed gs and a rate of turn ω). The center of the turn (`center`), beginning of turn (`BOT`), and end of turn (`EOT`) points can be calculated using vector calculations based on the positions p_1, p_2, p_3 corresponding to the (2 dimensional) positions of $np1, np2, np3$:

$$\begin{aligned}
\mathbf{a} &= \mathbf{p_3} - \mathbf{p_2} \\
\mathbf{b} &= \mathbf{p_1} - \mathbf{p_2} \\
\mathbf{u} &= \hat{\mathbf{a}} + \hat{\mathbf{b}} \\
k &= R/\sqrt{(\mathbf{u}\cdot\mathbf{u}) - (\mathbf{u}\cdot\hat{\mathbf{a}})^2} \\
\mathbf{w} &= k\mathbf{u} \\
\mathbf{w_a} &= (\mathbf{w}\cdot\hat{\mathbf{a}})\,\hat{\mathbf{a}} \\
\mathbf{w_b} &= (\mathbf{w}\cdot\hat{\mathbf{b}})\,\hat{\mathbf{b}} \\
\texttt{center} &= \mathbf{p_2} + \mathbf{w} \\
\texttt{BOT} &= \mathbf{p_2} + \mathbf{w_b} \\
\texttt{EOT} &= \mathbf{p_2} + \mathbf{w_a}
\end{aligned}$$

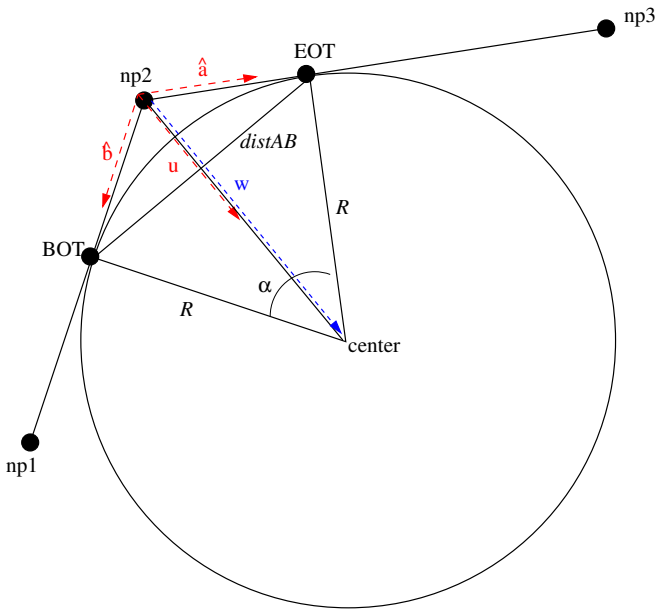

Figure 9: Calculating BOT and EOT For Fly-by

The geometry associated with these calculation is shown in Figure 9. These formulas are derived in Appendix A. Using these calculated positions and the speed into the turn (gs), it is possible to calculate the arc length of the turn, d, and the time spent in the turn, `turnTime`, as follows:

$$\begin{aligned}
&\texttt{distAB} = \|\texttt{BOT} - \texttt{EOT}\| \\
&\alpha = 2\sin^{-1}(\texttt{distAB}/(2R)) \\
&d = \alpha R \\
&\texttt{turnTime} = d/gs
\end{aligned}$$

These calculated times are used to construct the time component of the `EOT` point. The altitude is computed using the vertical speed profile in the linear plan. However, these points are usually not marked as *altPreserved* so they may be altered in a later pass. We note that the ground speed used in the turn is the same as the ground speed of the original leg of the 4D-plan into the turn vertex. We assume that the turn speed is constant until the turn is complete. If the speed of the second leg ($np2$ to $np3$) of the turn is different from the first leg ($np1$ to $np2$) in the original 4D-plan, we defer the ground speed acceleration until after the turn is complete. Because the total distance from $np1$ to $np3$ will have changed (i.e. the turn path is shorter), the ground speed of the second leg will have changed from its original value. We are once again faced with a semantics question. It is clear that once a turn is introduced, one cannot preserve both the original ground speed and the arrival time at the

next waypoint ($np3$). So what is the intended meaning of the original 4D-plan? Should times in the original plan have priority over ground speed or should it be the other way around? We have chosen to preserve ground speeds at the expense of time. This means that the times of all of the waypoints from $np3$ onward must be *time-shifted*. This process is described in the next section, Section 3.6.3.

There are several anomalous cases that arise where the leg distances are inadequate for proper turn generation:

- The first leg is so short that the `BOT` occurs before the first point.
- The last leg is so short that the `EOT` occurs after the last point.
- Two consecutive turns occur where the first turn has insufficient time to complete before the second turn needs to begin.

If these are encountered during generation, the process fails and an error report is created. Sometimes 4D-plans can be repaired before the generation process to prevent generation failure. See Appendix B for more details.

Figure 10 illustrates the successful generation of multiple turns in a 4D-plan. Note the altitudes for non-marked points are not necessarily identical

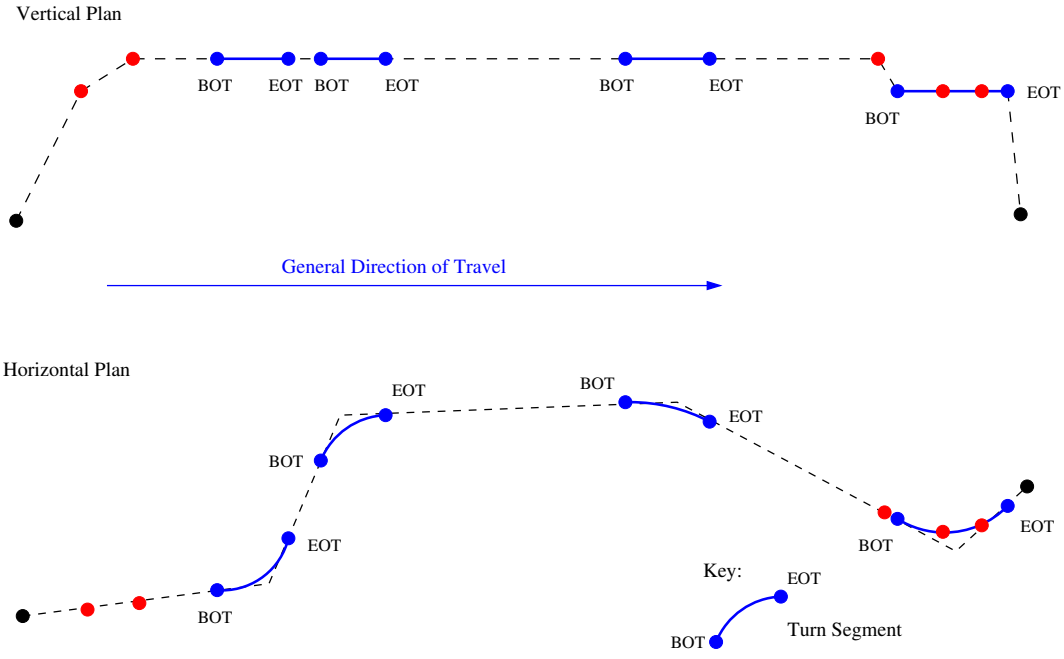

Figure 10: Example Kinematic Plan with (Fly-by) Turns

to the altitudes in the original linear plan. This can be seen in the `BOT` and

EOT of the final turn segment. Here they are level (as no special altitude was assigned to these new points), but the corresponding sections of the original linear plan both indicated descents—the only level segment in that part of the linear plan was between the two marked vertical speed change points (red in the diagram).

Fly-over, Direct-to-Fix Turns: This type of turn is actually simpler to compute, though it results in a trajectory with different tracks than the original plan. To generate a fly-by turn, the waypoint in question becomes the beginning of turn point and the aircraft simply turns in the appropriate direction until it is pointed at the next point. This tends to result in a longer overall flight path. The turn radius R is calculated as above. Also as above, the BOT

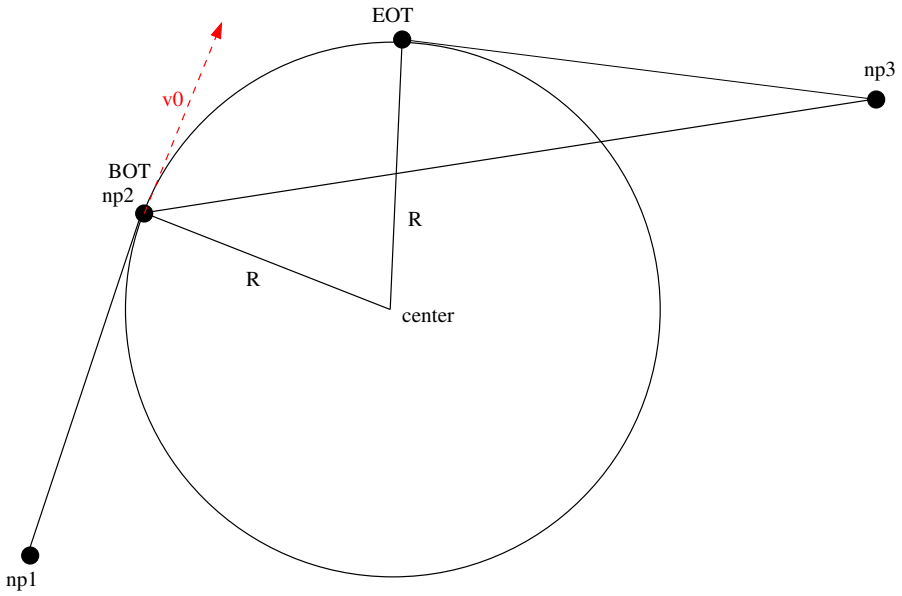

Figure 11: Calculating BOT and EOT for Fly-Over

and EOT points where the segments are tangent to the turn circle, with the center of the turn being on a segment perpendicular to the initial velocity v_0. Left or right ($\epsilon = -1$ or $\epsilon = +1$, respectively) is determined by the difference between the tracks of v_0 and (EOT − BOT) (which happens to be the same as the track of the segment $np2$ to $np3$). EOT is calculated by translating into a coordinate system with the center at $(0, 0)$ and calculating the appropriate point of tangency Q of the vector passing through $np3$ [6].

$$Q(s, R, \epsilon) = (\alpha s_x + \epsilon\beta s_y,\ \alpha s_y - \epsilon\beta s_x)$$

where $\delta = s \cdot s - R^2$ and $\alpha = R^2/(s \cdot s)$, $\beta = R\sqrt{\delta}/(s \cdot s)$ This calculation is only valid when $\delta >= 0$. After computing Q:

$$\texttt{center} = \texttt{BOT} + \hat{v_0}^{\perp} R$$
$$\texttt{EOT} = Q(s, R, \epsilon) + \texttt{center}$$

3.6.3 Ground Speed Correction Pass

Once turns have been generated, points must be time shifted in the kinematic plan in order to restore the ground speeds from the original linear plan. We note again that it is impossible to preserve both the ground speeds of the original 4D-plan and the waypoint times, because the turns alter the path distance. A semantic choice has to be made and we have decided to preserve ground speeds rather than times. Figure 12 illustrates this time-shifting process for the running example. In order to preserve the original ground speed,

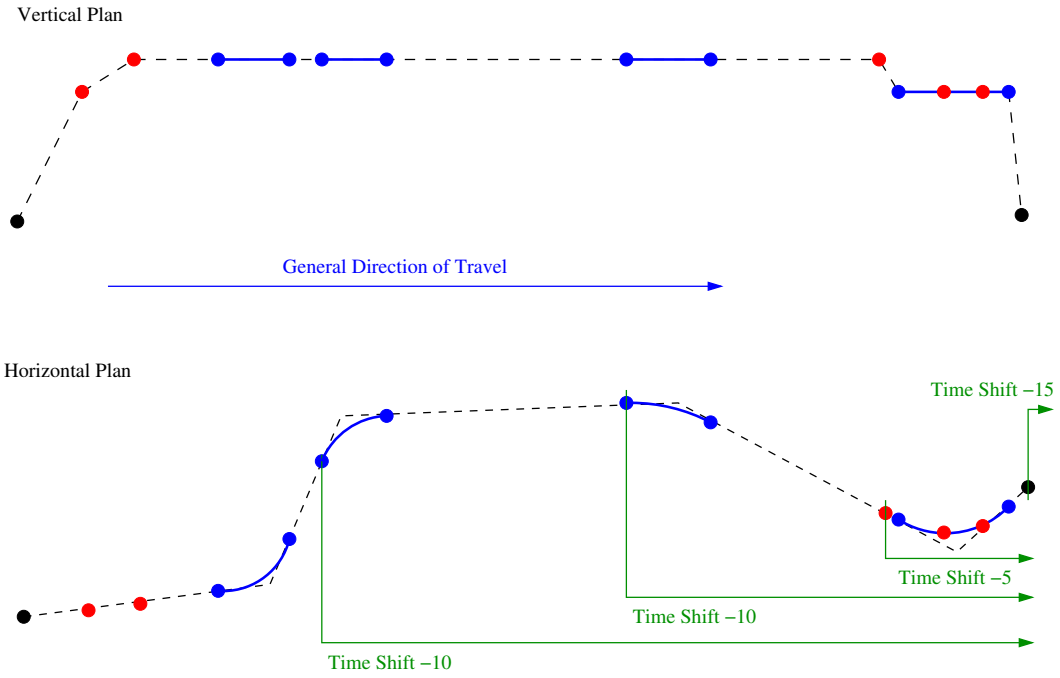

Figure 12: Example Kinematic Plan with Ground Speed Corrections

all points after the turn need to be time shifted back by exactly the same amount to account for the shorter distance traveled. Note that this ensures that all ground speeds and vertical speeds remain the same after the shift point. However, the absolute times may have changed significantly.

3.6.4 Speed Acceleration Generation Pass

Two consecutive collinear legs implicitly specify a ground speed acceleration when the ground speeds of the two legs are different. This is illustrated in Figure 5. The ground speeds $gs1$ and $gs2$ are not explicitly specified in the 4D-plan. However, the specified times, T_1, T_2, and T_3, implicitly prescribe specific ground speeds. The total time required to achieve the ground speed change, is given by the following equation:

$$t_{end} = (gs_2 - gs_1)/a_{gs}$$

with ground speed acceleration a_{gs}. A semantic decision has to be made concerning the beginning time of the ground speed acceleration. There are two basic choices:

- The acceleration begins at the point in the 4D-plan where the ground speeds change.
- The acceleration begins prior to the ground speed change point. A reasonable choice would be to start so that the original ground speed change point in the 4D-plan is located half way through the acceleration zone.

We have used the first option in our generation algorithm. This choice was largely made because of our restriction of not allowing overlapping horizontal acceleration zones—it is necessary to delay ground speed accelerations until after turns have completed.

The beginning of ground speed acceleration (`BGS`) is the point in the plan where there is specified speed change, while the end point (`EGS`) is determined by the calculated acceleration time t_{end} and a simple quadratic formula:

$$s(t_{end}) = s(0) + v(0)\, t_{end} + \frac{1}{2}\, a_{gs}\, t_{end}^2$$

where $s(0)$ is the location of the `BGS` point and $v(0)$ is the velocity into the (`BGS`) point. After the `BGS-EGS` acceleration zone, all of the points after the `EGS` have to be time-shifted to preserve the original ground speeds in the 4D-plan.

The primary complication that occurs in `BGS-EGS` generation is preventing overlapping with other acceleration zones. We solve some of the overlap problems by delaying the beginning of ground speed acceleration until the other horizontal acceleration has been completed. Another problem occurs when there is insufficient distance (before the next point) to complete the ground speed acceleration. The latter case results in a generation failure.

The running example assumes a constant ground speed, so there is no change to the figure for this step.

3.6.5 Constant Vertical Speed Pass

The altitudes and times at the first, last, and all marked points (i.e. *altPreserved*) in the plan are used to determine average vertical speeds. The altitudes for all points between marked ones are adjusted so that the vertical speeds into and out of those points match the appropriate average.

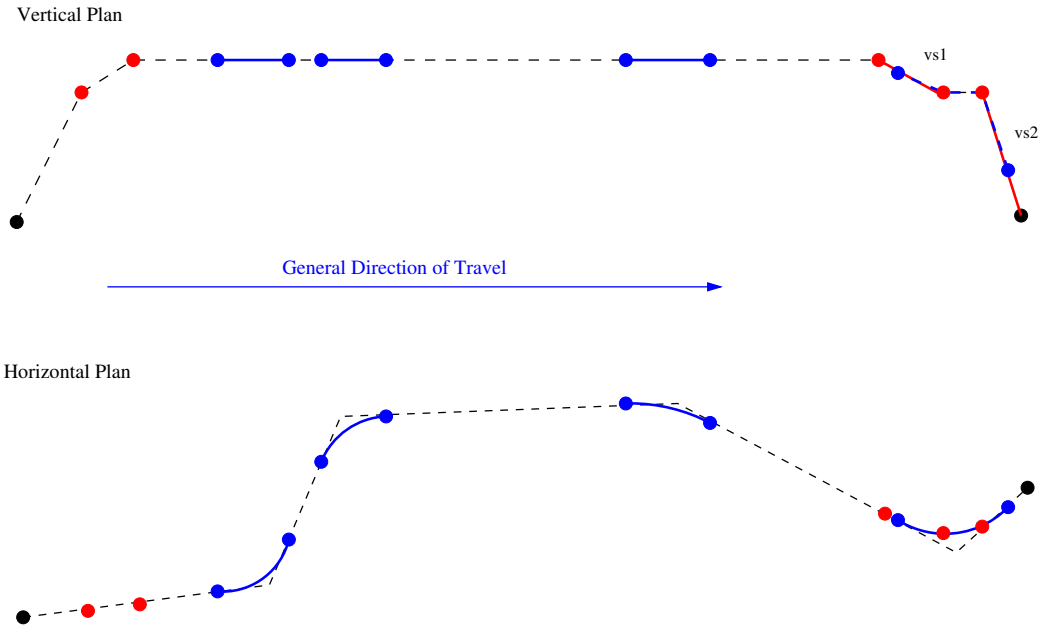

Figure 13: Example Kinematic Plan with Constant Vertical Speed Corrections

This is illustrated in Figure 13. We note that the altitudes of the *altPreserved* points will match the original 4D-plan altitudes because none of the previous passes of the generation algorithm alter the altitudes of these points. This will preserve, for example, areas of level flight at the appropriate altitudes. However, the times of these points are often changed.

3.6.6 Vertical Acceleration Generation Pass

Vertical acceleration points are only concerned with the vertical frame of reference. If we have two consecutive points with altitudes z_1 and z_2 and times t_1, t_2, a vertical speed, v_2 is implicitly defined between these points. The vertical speed into the first point v_1 is also needed to generate the vertical TCPs. The total time dt to achieve the vertical acceleration is given by the following equation

$$dt = \frac{\Delta v}{a_{vs}} = \frac{v_2 - v_1}{a_{vs}}$$

where a_{vs} is the vertical acceleration. The beginning and end times of the acceleration zone can then be easily computed as follows:

$$T_{begin} = t_2 - \frac{dt}{2}$$
$$T_{end} = t_2 + \frac{dt}{2}$$

An example of these values can be seen in Figure 14. Given the begin time

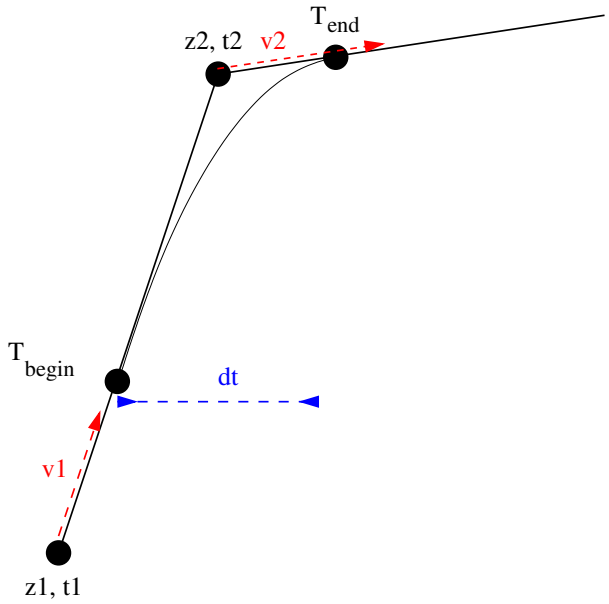

Figure 14: Calculating T_{begin} and T_{end}

and end time, the horizontal positions can be calculated from the working kinematic plan at those times, because the horizontal generation passes have all been completed. See Figure 15. Because the vertical acceleration zones are comparatively small, they are shown in the insets. Note that the last two (new) vertical speed change points have their horizontal positions determined by the turn.

The final plan is shown in Figure 16. Note that the vertical acceleration zones are shown by a single red dot in the figure, though they are actually each denoted by a pair of closely spaced TCPs (as shown in the figure inserts).

3.7 RTA Points

This trajectory generation algorithm preserves the ground speeds of the original plan, and assumes that times are estimated time of arrival (ETA), soft

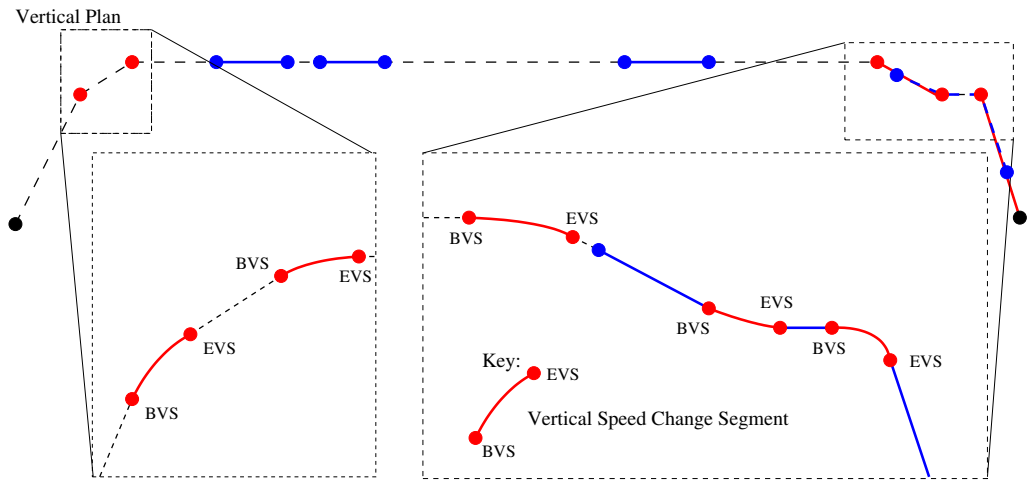

Figure 15: Example Kinematic Plan with Vertical Accelerations

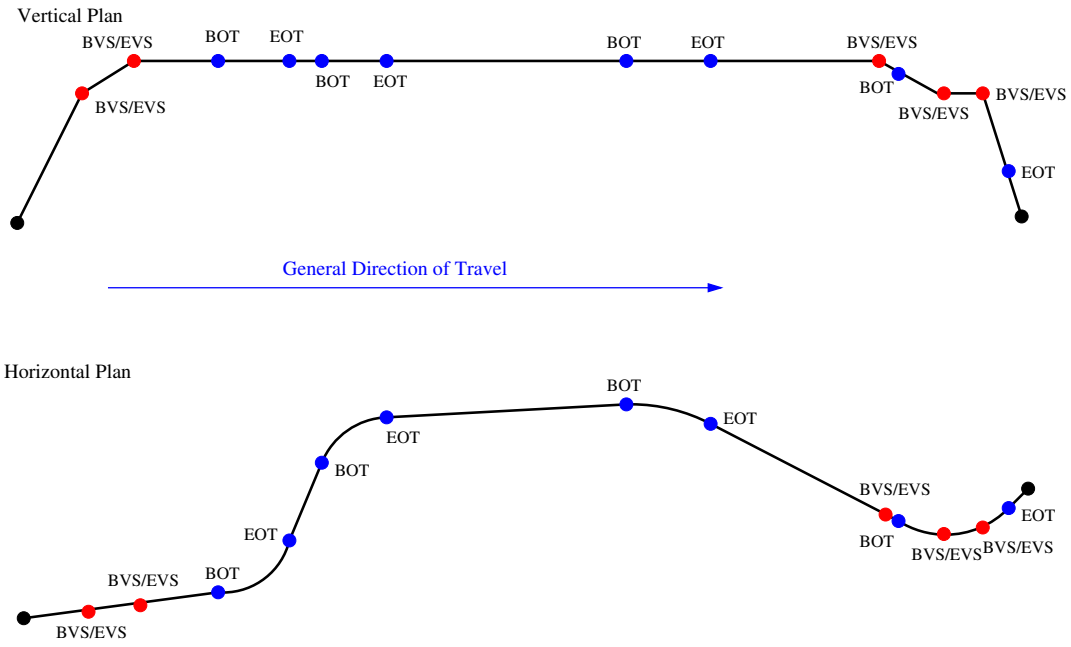

Figure 16: Finished Example: Kinematic Plan

constraints. What if instead of specified ground speeds, there are one or more points with required time of arrival (RTA) hard constraints?

In this case, there are two main options: using acceleration zones tailored to achieve the desired RTA (possibly introducing large speed changes), or smoothing out the overall ground speed on all or part of the plan. The first option can be accomplished as part of the ground speed acceleration pass (section 3.6.4), but it may introduce unreasonable speeds (low or high) or accelerations (generally due to distance limitations). The second option would primarily involve modifying the ground speed correction pass (section 3.6.3), but this would require the additional correction of the turn metadata. Turn points would not necessarily require any recalculation of position, but their time and rate of turn values may need to be adjusted. In general, this is safe if the original RTA is later than the generated kinematic point time, because taking a gentler turn should never be a problem, but if the RTA is *earlier* than the calculated kinematic time, then this could result in a turn that is too severe for normal operation. With fly-by turns, this latter case can only possibly arise if the original plan was not feasible, because the kinematic plan should never have an overall surface distance that is greater than the linear plan's. The same is not necessarily true of trajectories with fly-over points, as distances (and so times) may be greater than in the linear plan.

4 Communication of Trajectories

Trajectories generated as a sequence of 4D points separated by a short time interval have the advantage of being able to model the actual trajectory that will be flown with very high precision. Trajectories of this form can be created by a trajectory generator that is based on aircraft dynamics. Unfortunately, such trajectories require a large amount of data to transmit them from one location to another. To transmit these trajectories in an efficient way would require some kind of compression algorithm. One approach is to represent trajectories as mathematical functions such as those advocated by Delahaye [2]. This paper advocates a less drastic approach to compression by adding more semantics to some of the points of the trajectory. The trajectories defined in this paper can be transmitted with only minor variations to proposed ADS-B standards. The current ADS-B signal includes state information in the form of a time stamp, position, and velocity (along with some additional data). Proposed extensions have included the addition of one or more intent points, containing similar data. Transmitting TCPs would only need the additional inclusion of an enumerated type field and sequence number (possibly a single byte of data combined) and an acceleration value (typically 2 or 4 bytes, depending on accuracy). Even if each message only contains data for one TCP, a complete trajectory could be transmitted incrementally over several

transmissions. Given this information, any receiving party can recreate the same trajectory from the TCPs using simple kinematic calculations.

5 Concluding Remarks

In this paper, the idea of a formal semantics for 4D flight plans is proposed. Traditionally the meaning of a flight plan has been a rough prescription of the path actually flown by a pilot trying to follow it. The pilot (or flight management system) can thus be viewed as providing an operational semantics for a flight plan. But, since different pilots (or flight management systems) may follow a flight plan in different ways, a flight plan is ambiguous. Prior to a flight, a flight plan can be seen as a set of constraints, but there is not an a-priori rigorous definition of its meaning. Thus, a flight plan is inherently too imprecise to form the basis of an advanced concept for the National Airspace System. We have shown that even a linear 4D flight plan is too ambiguous to serve as a standard.

In this paper, we present the idea of a denotational semantics for 4D flight plans, i.e., a mathematically precise meaning of a 4D linear flight plan. We present an algorithm that translates a 4D linear plan into a trajectory that has a continuous velocity vector. This trajectory provides a rigorous meaning for the 4D-plan. However, in the process of developing this translation, we have discovered many 4D-plans that are inherently ill-formed. We also discovered that many aspects of a 4D-plan cannot be preserved at the same time. A translation algorithm must make choices about what properties are preserved. Our trajectory generation algorithm makes many prioritizations and tradeoffs. The rationales for these choices are presented. The trajectory generation algorithm has been implemented in Java and C++ and is available via an open-source release license. We do not present all of the mathematical details of the translation. Instead, the algorithm is presented via diagrams and equations. In future work we hope to develop a formal mathematical model of this algorithm and thus provide a rigorous formal semantics for 4D flight plans.

References

1. INTENT Consortium. Intent WP2: Results of real-time and fast time simulations, d2-3, part 2, appendices. www.intentproject.org.

2. Daniel Delahaye, Stéphane Puechmorel, P. Tsiotras, and E. Feron. Mathematical models for aircraft trajectory design: A survey. In *Air Traffic Management and Systems*, pages 205–247. Springer, 2014.

3. Yancy Diaz-Mercado, Sung G. Lee, Magnus Egerstedt, and Shih Yih Young. Optimal trajectory generation for next generation flight management systems. In *Proceedings of the 32 Digital Avionics Systems Conference*, Oct 2013.

4. Federal Aviation Administration (FAA). Trajectory Based Operations. www.faa.gov/nextgen/portfolio/sol_sets/tbo/.

5. Stéphane Mondoloni and Daniel Kirk. Proposed trajectory prediction and exchange information items for flight information exchange model (fixm). Technical report, MITRE, 2012.

6. Anthony Narkawicz, César Muñoz, and Gilles Dowek. Provably correct conflict prevention bands algorithms. *Science of Computer Programming*, 77(1–2):1039–1057, September 2012.

7. RTCA SC-186. Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-B), 2002.

8. R.C.J. Ruigrok and M.S.V. Valenti Clari. The impact of aircraft intent information and traffic separation assurance responsibility on en-route airspace capacity. In *5th FAA/EUROCONTROL ATM R&D Seminar*, Jun 2003.

9. Sip Swierstra and Steven Green. Common trajectory prediction capability for decision support tools. In *5th USA/Eurocontrol ATM R&D Seminar, Budapest, Hungary*, 2003.

10. Robert A Vivona, Karen T Cate, and Steven M Green. Abstraction techniques for capturing and comparing trajectory predictor capabilities and requirements. In *AIAA Guidance, Navigation and Control Conference, Honolulu, HI*, 2008.

Appendix A

Derivation of Turn TCPs Equation

The derivation of the formula 3.6.2 in section 3.6.2 is based on Figure 9. This has been simplified in Figure A1. Geometrically, the problem is simply to

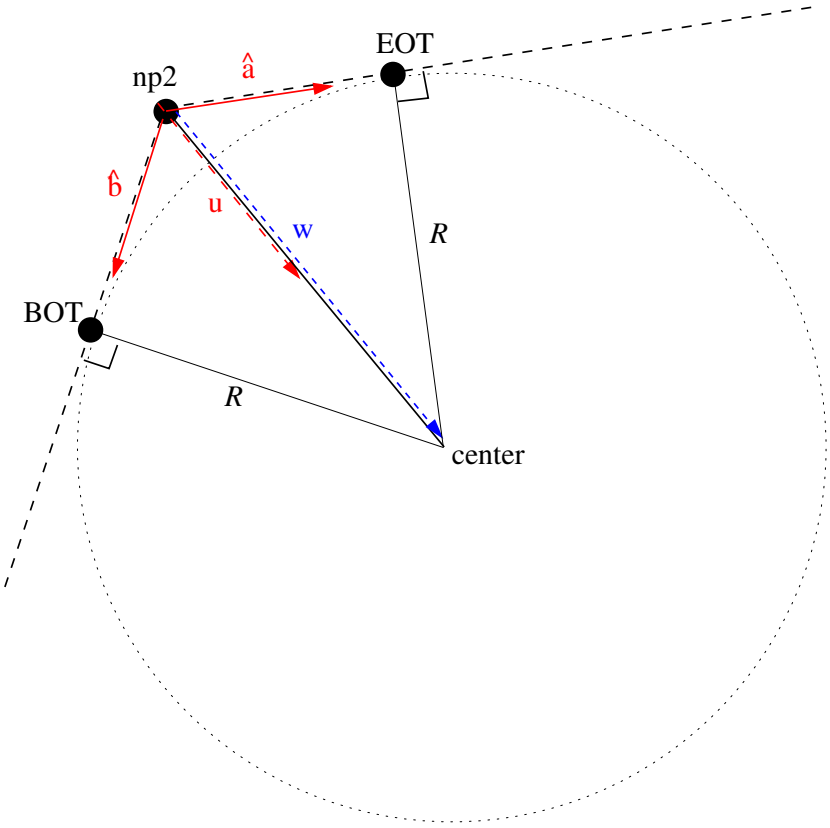

Figure A1: Calculating BOT and EOT For Fly-by

inscribe a circle of radius R into the angle at $np2$ and find the tangent points. We begin with the property that the (unit) bisector vector $\mathbf{u}$ can be computed as follows

$$\mathbf{u} = \hat{\mathbf{a}} + \hat{\mathbf{b}}$$

The key to finding the TCPs is to first find the center of the turn. We let

$$\mathbf{w} = k\mathbf{u}$$

represent the vector that begins at point $np2$ and ends at the center. We solve for k using the Pythagorean theorem. We note that the lengths of the legs of

the triangle are R, $|\mathbf{w}|$, and $|\hat{\mathbf{a}} \cdot \mathbf{w}|$. Thus we have

$$|\mathbf{w}|^2 = R^2 + (\hat{\mathbf{a}} \cdot \mathbf{w})^2$$

Solving for k we obtain

$$k = R/\sqrt{(\mathbf{u} \cdot \mathbf{u}) - (\mathbf{u} \cdot \hat{\mathbf{a}})^2}$$

Using this value for k, we can calculate the needed TCPs:

$$\texttt{center} = \mathbf{p_2} + \mathbf{w}$$
$$\texttt{BOT} = \mathbf{p_2} + (\mathbf{w} \cdot \hat{\mathbf{b}})\ \hat{\mathbf{b}}$$
$$\texttt{EOT} = \mathbf{p_2} + (\mathbf{w} \cdot \hat{\mathbf{a}})\ \hat{\mathbf{a}}$$

Appendix B

Repair of 4D-Plans

We have discussed many inherent ambiguities that arise from a linear 4D plan. Many of these ambiguities can be resolved by selecting precisely what features of the plan will remain invariant and which ones will be allowed to be modified. However, there are plans that are unachievable given the specific values of the acceleration parameters. Ideally, such plans would need to be modified so that the translation will produce the intended result. In some circumstances, however, it is not possible to divine the original maker's actual intent, necessitating an interpretation of what might otherwise be considered ill-formed plans. Several of these cases are explored in the next subsections, along with potential pre-processing actions that can help to allow the translation to complete into a well-formed, consistent trajectory.

B.1 Short First Leg

Suppose that the 4D-plan has an initial leg that is too short. This is illustrated in Figure B1. We assume that the aircraft is currently located at the first

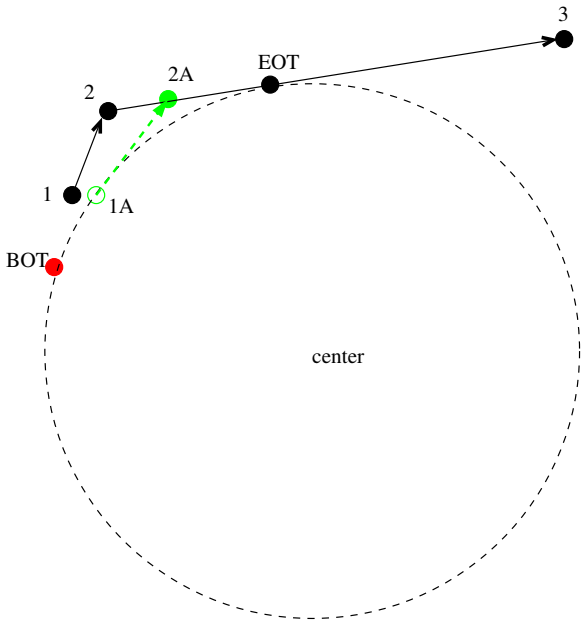

Figure B1: Short First Leg

black dot, point 1. The radius of the circle is determined by the speed of

the aircraft and the specified turn bank angle (which determines the turn rate). Unfortunately, given this initial leg, the radius needs to be smaller. To achieve the turn in a manner that completes the turn on the second leg with the proper track, the current location of the aircraft would have to be where the green circle (1A) is located, with a velocity direction tangent to the turn circle (leading to point 2A). Note that the actual current location of the aircraft is indeed located on a tangent line to the circle but given that it intersects with point 2, it occurs *after* the `BOT` point, and thus the turn is unachievable.

What should we do with a 4D-plan that has a short leg like this? Assuming point 1 represents the aircraft's actual positions, it is impossible to just move the aircraft instantly to point 1A, and you cannot merely delete the vertex point 2 because this would create an instantaneous velocity change for the aircraft.

One approach is to move the second vertex rather than delete as shown in Figure B2. The current location of the aircraft (i.e. point 1) becomes the

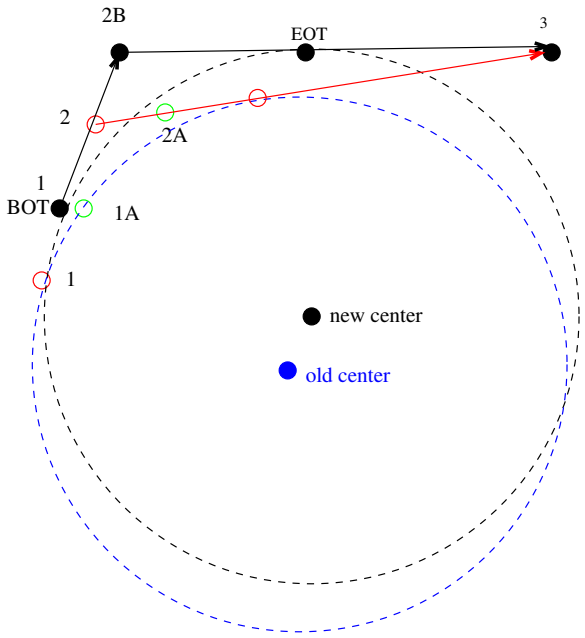

Figure B2: Repair Approach For Short First Leg

revised `BOT`. The current leg is extended over the previous vertex (point 2) to the next waypoint, the new point 2B. Note that the track of the second leg will be different from the original 4D-plan, but it is not possible to retain both the turn and the end vector.

A similar problem arises when the final leg is too short.

B.2 Second Leg Short

If the second leg is too short there are several different reasonable repair approaches. This situation is illustrated in Figure B3. In this case, a second turn

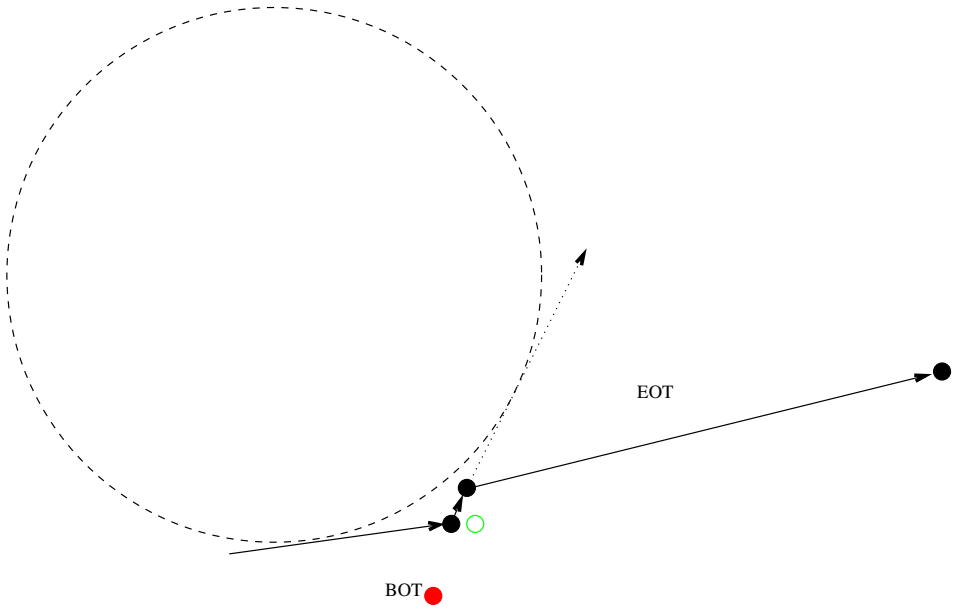

Figure B3: Second Leg Short

is initiated before the first one can be completed. One approach would be to try to calculate exactly where the second turn should start and perform a kind of s-maneuver. Another approach is to just delete the second and third points and go straight to the fourth. Still another approach would be to merge points two and three into one point half way between them—we are currently implementing the last option. But what if the second turn is in the same direction as show in Figure B4? There is no obvious repair for this situation. It may only be possible to delete one or more of the offending points and attempt to recover what is left of the plan.

B.3 Ground Speed Cases

There are short leg cases where there is insufficient room to achieve the speed before the next waypoint. This is illustrated in Figure B5. The plan specifies that the aircraft travels at speeds 300, then 500, then 450. But if the distance between point 2 and 3 is too short, then at the specified acceleration rate, 500 knots by point 3 cannot be achieved. How can this be repaired? One approach is to eliminate either point 2 or 3 or both. This will effectively average out the speeds. We have chosen to just delete point 3.

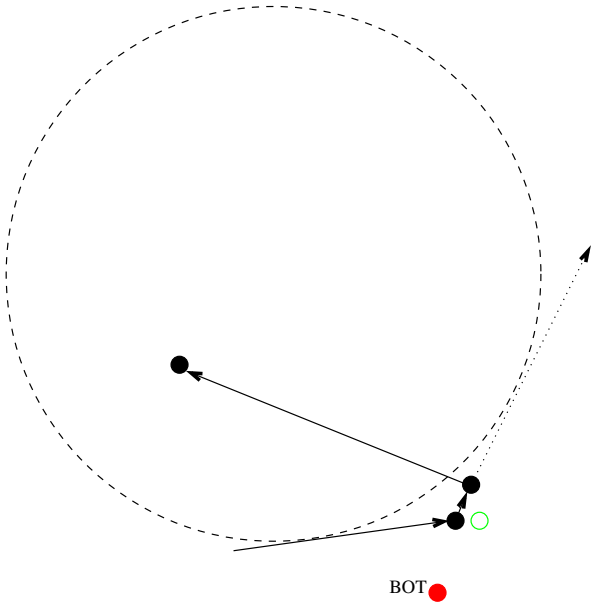

Figure B4: Second Short Leg With Turns In Same Direction

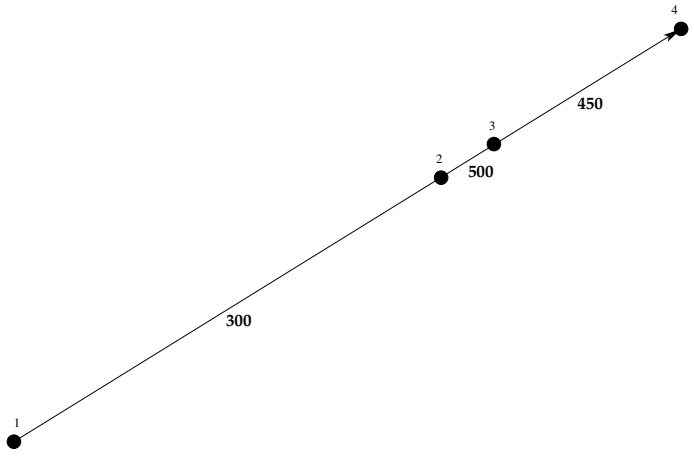

Figure B5: Second Short Leg: Insufficient Room For Speed Change

B.4 Vertical Speed Cases

There are vertical profiles where there are segments that have insufficient distance to meet the 4D-plan specification. This is illustrated in Figure B6. In

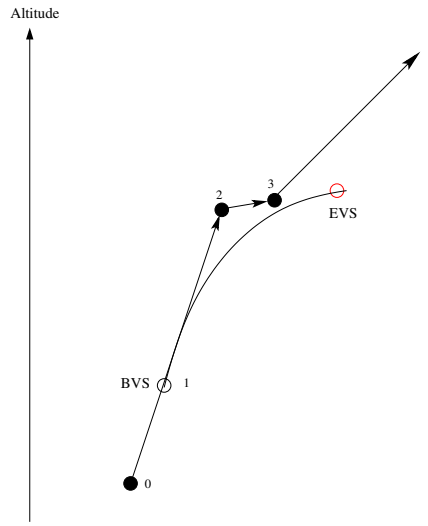

Figure B6: Vertical Climb With Short Segment

this figure, a vertical deceleration is specified from waypoint 1 to waypoint 2. The calculated beginning of the deceleration is at the point labeled `BVS`. However, the computed end point of the deceleration, `EVS`, is beyond point 2 where a vertical acceleration is specified. This flight plan can be repaired by removing points 2 and 3 or possible just one of these points. Our algorithm removes point 3.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 12 - 2014	Technical Memorandum	

4. TITLE AND SUBTITLE

Towards a Formal Semantics of Flight Plans and Trajectories

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hagen, George E.; Butler, Ricky W.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

411931.02.02.07.13.01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-20476

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA-TM-2014-218662

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 63
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In the current air traffic management system, flight plans are often used only as a planning tool. These plans are implicitly translated into trajectories by the pilot or by the flight management system, and subsequently flown by the aircraft. This translation process inevitably introduces differences between the plan and the trajectory. However, given the current intended usage, exact correspondence between the plan and the trajectory is not needed. To achieve greater capacity and efficiency, future air traffic management concepts are being designed around the use of trajectories where predictability is extremely important. In this paper, a mathematical relationship between flight plans and trajectories is explored with the goal of making feasible, highly accurate predictions of future positions and velocities of aircraft. One way to improve predicability int the airspace is to develop more sophisticated models of air- craft dynamics with more precise sensing of critical parameters. As valuable as these models may be, this paper takes a different approach to achieve improved predictability. The goal here is to describe, in mathematically precise detail, a formal language of trajectories, whereby the receiver of the trajectory information can know precisely what the sender intended. Although even a four dimensional flight plan is simple in structure, this paper will show that it is inherently ambiguous and will explore these issues in detail. In effect, we propose that a rigorous semantics for flight plans can be developed and this will serve as an important stepping stone towards trajectory-based operations in the National Airspace System.

15. SUBJECT TERMS

Air traffic management; Aircraft trajectory; Algorithms; Flight plan

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	43	(757) 864-9658