Observations on SOFIA Observation Scheduling: Search and Inference with Discrete and Continuous Constraints

Jeremy Frank and **Michael A. K. Gross*** and **Elif Kürklü†**
NASA Ames Research Center
Mail Stop N269-3
Moffett Field CA 94035-1000
{frank,ekurklu}@email.arc.nasa.gov, mgross@mail.arc.nasa.gov

Abstract

In previous work we describe the problem of scheduling flights for SOFIA, an airborne observatory. Scheduling in this domain requires solving many Initial Value Problems (IVPs) and Boundary Value Problems (BVPs) to evaluate constraints on the observatory's ground track and telescope elevation limits. These are costly operations, and become more so when accounting for winds and fuel consumption. In this paper we show how to reduce the number of IVPs and BVPs needed to schedule SOFIA by restricting the set of flight plans we sample. Empirical studies show that the restriction costs us little in terms of the value of the flight plans we can build. The restriction allowed us to reformulate part of the search problem as a zero-finding problem. The result is a simplified planning model and significant savings in computation time.

Introduction

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is NASA's next generation airborne astronomical observatory. The facility consists of a 747-SP modified to accommodate a 2.5 meter telescope. SOFIA is expected to fly an average of 140 science flights per year over it's 20 year lifetime. The SOFIA telescope is mounted aft of the wings on the port side of the aircraft and is articulated through a range of 20 to 60 degrees of elevation. The telescope has minimal lateral flexibility; thus, the aircraft must turn constantly to maintain the telescope's focus on an object during observations. A significant problem in future SOFIA operations is that of scheduling Facility Instrument (FI) flights in support of the SOFIA General Investigator (GI) program. GIs are expected to propose small numbers of observations, and many observations must be grouped together to make up single flights. Approximately 70 GI flight per year are expected, with 5-15 observations per flight. The scope of the flight planning problem for supporting GI observations with the anticipated flight rate for SOFIA makes the manual approach for flight planning daunting.

Automated flight planning for SOFIA involves selecting observations to perform and scheduling these observations. Verifying that constraints on the observations are satisfied involves solving both Initial Value problems (IVPs) and Boundary Value Problems (BVPs) to find the aircraft's ground track, determine aircraft fuel consumption, and check that observations stay within proscribed elevation limits. In previous work (FK03) we describe ForwardPlanner, a progression style search algorithm that uses a combination of lookahead and heuristics to guide search. This algorithm is very costly to run, as it solves a large number of IVPs and BVPs merely to determine whether an unscheduled observation can be added to the flight plan, let alone decide whether it is a good idea to do so. A set of well-founded assumptions allowed us to eliminate a large number of calls to solve the IVPs and BVPs. While this reformulation actually eliminates feasible flight plans, empirical results show that the resulting algorithm produces high quality flight plans at a fraction of the computational effort.

The rest of the paper is organized as follows. We first describe the high fidelity SOFIA model. We then re-examine the ForwardPlanner algorithm and describe a principal source of the increased computational costs of flight planning. We then describe a way of migrating some of the search into the underlying constraint reasoning component by means of some well-founded assumptions. This allows us to eliminate a large number of expensive ground track construction steps without sacrificing the ability to construct good flight plans. We perform several experiments to validate the approach. Finally, we discuss the implications of our reformulation of the computational search on the planning model.

Improving Model Fidelity

The SFPP (Single Flight Planning Problem) consists of a number of observation requests, a flight day, and a takeoff and landing airport. The objective is to find a flight plan that maximizes the summed priority of the observations performed while obeying the constraints governing legal flights. The aircraft activities are *take-off, land, flight-leg* and *dead-leg*. Flight-legs require tracking an object and obeying visibility constraints, while dead-legs can be used to reposition the aircraft to enable flight-legs, and only consume time and fuel. A distinguished class of dead-legs are used to take off and return to the landing airport.

In previous work (FK03) we described the equations

*Universities Space Research Association
†QSS Group, Inc.

of motion of the aircraft using an Earth-centric coordinate system. These constraints are a simplification of the problem. In particular, the following factors were ignored:

- The impact of the true fuel consumption model of the aircraft on the flight time. Previously, we simply used a maximum flight duration as an analog of fuel consumption. The fuel consumption is actually a function of aircraft weight, Mach number, change in altitude, and outside temperature. Since flying reduces aircraft weight, the flight duration constraint in the old model is replaced by a differential equation that governs the fuel consumption.
- The impact of the Earth's shape on the ground track. The Earth is actually an oblate spheroid whose polar diameter and equatorial diameter are not the same. This has a reasonable impact on the actual ground track, and accounting for this invalidates the differential equations in the previous model that constrain the ground track.
- The impact of winds on the ground track. As the aircraft flies, the wind direction and velocity changes the ground speed. This invalidates the assumption in the old model that the ground speed is constant.

In this paper we use a more intuitive Cartesian coordinate system. The Earth is modeled as an oblate spheroid $\mathbf{E}$, whose surface is defined by the equation

$$\frac{x^2}{a^2} + \frac{y^2}{a^2} + \frac{z^2}{c^2} = 1 \tag{1}$$

where $c < a$. We define $\mathbf{p}$ as the aircraft's current position, and θ be the (Sidereal) time that the aircraft is at $\mathbf{p}$. Let $\vec{\mathbf{S}}$ be the vector from the center of $\mathbf{E}$ to $\mathbf{p}$. We then define $\vec{\mathbf{T}}$ as the vector defining the vector to an astronomical object o, and $\mathbf{P}$ as the plane tangent to E at $\mathbf{p}$. Define $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ as the unit vectors in the x, y, z directions respectively. Define $\vec{\mathbf{N}}$ as the vector normal to $\mathbf{P}$:

$$\vec{\mathbf{N}} = \frac{p_x}{a^2}\hat{\mathbf{i}} + \frac{p_y}{a^2}\hat{\mathbf{j}} + \frac{p_z}{c^2}\hat{\mathbf{k}} \tag{2}$$

(Note that $\vec{\mathbf{S}}$ and $\vec{\mathbf{N}}$ are generally not parallel since $\mathbf{E}$ is a spheroid.) Let $T_P = ||\vec{\mathbf{T_P}}||$, $N = ||\vec{\mathbf{N}}||$. Define $\vec{\mathbf{T_P}}$ as the projection of $\vec{\mathbf{T}}$ onto $\mathbf{P}$; this is the *object azimuth* at $\mathbf{p}$, and is given by

$$\vec{\mathbf{T_P}} = \vec{\mathbf{T}} - \frac{\vec{\mathbf{T}}\dot{\vec{\mathbf{N}}}}{N^2}\vec{\mathbf{N}} \tag{3}$$

Define $\vec{\mathbf{V}}$ as the desired heading of the aircraft. The observatory must track the object inducing $\vec{\mathbf{T}}$, subject to the constraint that the angle between $\vec{\mathbf{V}}$ and $\vec{\mathbf{T_P}}$ is 270°, because the telescope points out the left-hand side of the aircraft. Let $\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270}^\circ)$ be a rotation matrix that rotates a vector 270° around $\vec{\mathbf{N}}$, and v be the airspeed of the aircraft; then

$$\vec{\mathbf{V}} = v\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270}^\circ)\frac{\vec{\mathbf{T_P}}}{T_P} \tag{4}$$

Define $\vec{\mathbf{H}}$ as the elevation vector with respect to $\mathbf{P}$. We also require the angle h between $\vec{\mathbf{H}}$ and $\vec{\mathbf{T_P}}$ obey the constraint $20^\circ \leq h \leq 60^\circ$ throughout an observation. Most targets are infinitely far from Earth, so we assume $\dot{\vec{\mathbf{H}}}$ is given by:

$$\vec{\mathbf{H}} = \vec{\mathbf{T}} + \vec{\mathbf{S}} \tag{5}$$

From vector calculus we then get the equation for the elevation h:

$$h = \cos^{-1}\left(\frac{\vec{\mathbf{H}}\dot{\vec{\mathbf{T_P}}}}{||\vec{\mathbf{H}}||||\vec{\mathbf{T_P}}||}\right) \tag{6}$$

$\vec{\mathbf{T}}$ is a function of θ; this is because the Earth rotates on its axis. The vector $\vec{\mathbf{T}}$ traces a circle of radius $x^2+y^2 = \frac{c^2-d}{c^2}$, where $d = |\frac{\delta}{90^\circ}|$ in 24 hours (see (Mee91) for an explanation of this). The instantaneous change in $\mathbf{p}$ as the aircraft tracks o is $\frac{d\mathbf{p}}{d\theta} = \vec{\mathbf{V}}$. Since $\vec{\mathbf{V}}$ is a function of $\mathbf{T}$, it is a function of $o, \mathbf{p}$ and θ. Computing the ground track requires solving an Initial Value Problem (IVP). Solving for the ground track is necessary to compute h and check the elevation constraints.

It is worth noting that this formulation also makes it easy to add the effect of winds by adding the appropriate vectors to $\vec{\mathbf{V}}$, and also correct for aircraft pitch by rotating about $\vec{\mathbf{V}} \times \vec{\mathbf{N}}$. Due to the presence of winds, dead-legs in which the aircraft flies a heading for a fixed duration also require solving an IVP. Finally, flying to a particular location on the Earth requires solving a BVP. This problem arises when determining whether there is sufficient fuel to return to the landing airport from a particular location.

We previously used a simplified Euler's method with constant step size (Fer81) to verify the elevation limit constraint and compute the heading changes. While fast, it is prone to large ground-track errors and thus not suitable for SOFIA's requirements. The constraints are now solved using 5^{th}-order Runge-Kutta (Fer81) with error-adaptive step sizing. A gridded wind and temperature model is available to correct the ground track in the face of winds and provide temperature data for calculating fuel consumption. In addition, an aircraft performance model from Boeing is used to compute the exact fuel consumption for each of the 747-SP's engines, providing a much better estimate of flight time.

Unfortunately, the costs of solving these new constraints lead to a serious degradation in computational efficiency. Under the new model, the ForwardPlanner algorithm takes roughly 300 times as long to build a flight plan than it does using the simpler constraints and constraint reasoning system.

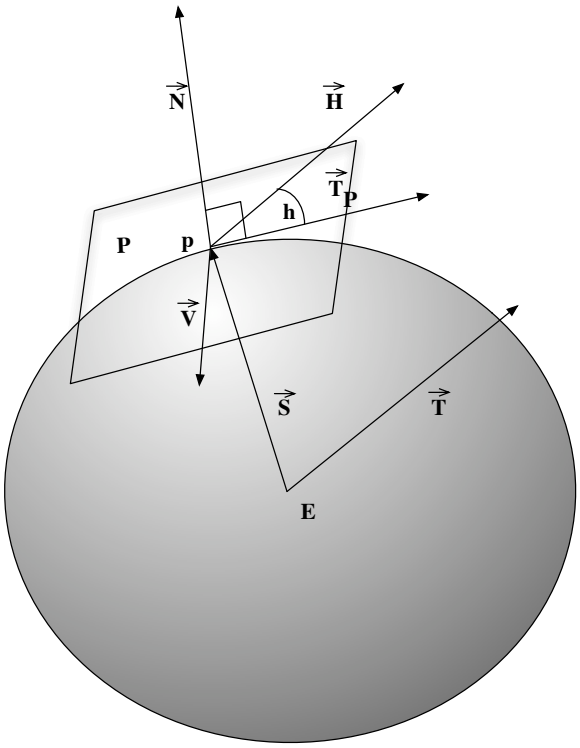

Figure 1: The Cartesian formulation of the instantaneous equations of motion of the aircraft and the elevation. We have exaggerated the spheroid E.

Explaining the Performance Hit

ForwardPlanner is a sampling-based approach that works as follows: unscheduled observations are checked for *feasibility*, then evaluated heuristically. An observation o is feasible at time θ and position $\mathbf{p}$ if there is a *dead-leg* of possibly zero duration that ensures that the observation is within the elevation limits after flying the dead-leg, the observation stays within the elevation limits for the required duration of the observation, and the aircraft can fly to the landing airport after the observation is finished. If the observation is not visible at position $\mathbf{p}$ at time θ, ForwardPlanner performs a search for the *shortest dead-leg* that satisfies these conditions. This is due to an efficiency requirement on flight plans; the cost of jet fuel is deemed one of the largest components of the ongoing operations cost of the observatory. We assume that the best global policy is to minimize dead-leg time required to set up an observation. The feasible observations are then evaluated by performing lookahead to construct a short flight plan, which also requires feasibility testing. These plans are used to rank each observation, and the ranks are used to bias a sampling approach to choose the next observation.

Our investigation into the ForwardPlanner algorithm revealed that we spend a considerable amount of time deciding which observations are feasible. This search is done by first changing the latitude of the aircraft to make the object visible, then performing a brute force search to reduce the dead-leg duration. If the resulting dead-leg exceeds a bound D the observation is considered infeasible. Each flight-leg and dead-leg construction step requires solving an IVP, while each check to ensure the aircraft can fly to the landing airport requires solving a BVP. In the worst case, this requires ForwardPlanner to solve a very large number of IVPs and BVPs. A typical number is $500,000$ IVPs, evenly split between flight-legs and dead-legs. This is true even though the dead-leg duration is limited, as are the heading choices for the enabling dead-leg. The computational expense of solving the BVPs and IVPs motivates us to search for ways to do less work to establish observation feasibility.

The shortest dead-leg making the object visible immediately after the dead-leg may not make the object visible for long enough. Suppose the aircraft is at high latitudes (above 60° absolute value). It is possible to fly West towards an object that is setting and make this object appear to rise. Observing the object will require flying perpendicular to the object, thus making it appear to set again. It is easy to construct a case where the aircraft may need to fly a longer dead-leg to enable an observation of the right duration. Such an object would have to be valuable to justify adding it to the flight plan; however, recent studies indicate that such Northerly flights are likely to be common, so this is a case worth bearing in mind. Similarly, the shortest dead-leg making the object visible for long enough may not enable the aircraft to fly home after the observation is completed. However, this only happens if the flight is almost finished. Thus, failing to establish this condition may lead to missing only one observation; the likelihood that this observation is critical to making the flight a good one is low, and is not as important a consideration as the previous issue.

Cheaper Feasibility Checks

In this section we describe how to change the solution methodology to reduce the cost of checking feasibility without sacrificing performance. First we describe a modification to the ForwardPlanner that *restricts* the set of plans that can be built, and explain why this leads to an increase in speed with a small impact on the value of the flight plans found. We then show how to leverage this change to get an even larger increase in speed, again with minimal performance impact.

Restricting the Set of Plans

The feasibility check may require a large number of expensive BVP checks to ensure that the aircraft can return to the landing airport. In some cases, a short dead-leg enabling an observation makes it impossible to return home, while a longer dead-leg both enables

the observation and allows the aircraft to return to the landing airport. This is not as counter intuitive as it seems, due to the complexities of the observation tracking constraints. However, it may be a waste of time to check this condition. SOFIA will normally take off and land again at the same airport, so the aircraft will trivially be in range of the landing airport for at least half the flight.

We can *restrict* the feasibility check in the following way: first, we find the shortest dead leg that enables the observation for the desired duration. If the aircraft can return to the landing airport after completing both this dead-leg and the observation, then the observation is feasible, otherwise it is not feasible. It might be possible to find a longer dead-leg that allows the aircraft to return to the landing airport, thus using this policy will exclude some flight plans. However, this change to the feasibility condition will generally affect observations in the latter half of the flight. We can then postpone the solution of the BVPs until *after* deciding to add an observation to the flight plan. If the aircraft can't return to the landing airport after performing the chosen observation and its shortest enabling dead-leg, then it is discarded and another observation is chosen to extend the flight. This will reduce the expected number of BVPs to solve significantly when most observations are feasible. We expect this modification to reduce the value of the flight plans found only when *high priority* observations are excluded late in the flight. In practice we find comparable flight plans after making this modification; in the interests of brevity, we do not report these results.

Changing the Division of Labor

Even after reducing the number of BVPs to solve, brute force search is still required to find the shortest dead-leg that enables the observation, and the performance gains achieved by eliminating BVPs are modest. However, we can take advantage of the new restricted feasibility condition by defining a function whose zeros define the properties of the shortest dead-leg enabling the observation. This defines a sub-problem that can be efficiently solved by using zero-finding algorithms such as Newton's Method. Because the resulting formulation allows us to search the full continuous space of dead-legs, we avoid discretizing the search space to enable brute-force search, and may also find *shorter* dead-legs.

Using the restricted conditions on object feasibility, the dead-leg construction phase of the feasibility check requires finding the heading and duration of the shortest dead-leg that enables the observation for a sufficient amount of time. A dead-leg may be necessary for one of two reasons. The first reason is that an observation is not visible at the current position and time. The second reason is that the observation is not visible for long enough We will treat these cases separately.

Let us consider the *feasible region* of an observation o. This region is the set of positions on the Earth from which the observation is visible, and is the annulus defined by two circles centered at the nadir position of o whose radii are the coelevation limits of the telescope (in SOFIA's case, the radii of these circles are 30° and 70°). Suppose the aircraft is outside the feasible region. We want the aircraft to be in the feasible region after completing the dead-leg. Now, the *shortest* leg would put the aircraft on the *boundary* of the feasible region, as opposed to anywhere strictly inside it. This means that the object elevation h will equal one of the two elevation limits after flying the dead-leg. If the aircraft begins inside the inner circle of the annulus, then we want the object to be precisely at the the upper telescope elevation limit of 60°, while if it is outside the outer circle, we want the object to be at the lower telescope elevation limit of 20°.

If the object was fixed relative to the ground, we could simply fly directly towards or away from the object, since this maximizes the rate of change of the object elevation. However, as we mentioned, the object appears to move across the Earth as the Earth rotates. We could fly a dead-leg that tracks the object as it moves, but that would not minimize the flight distance. We use the following intuition: we fly a dead-leg that *ends* with the aircraft flying either directly towards or directly away from the object to be observed. Intuitively, this is the correct policy when the object is nearly in view, or near the end of longer dead-legs. Observatory policy will normally prevent dead-legs longer than a few tens of minutes, so this intuition will likely produce very short, if not "locally optimal" dead-legs.

Suppose flying a dead-leg on heading b_i for duration d results in aircraft heading vector $\vec{\mathbf{V_d}}$ at the aircraft's new position. We can now compute the angle r between $\vec{\mathbf{V_d}}$ and the object azimuth at the new position $\vec{\mathbf{T_P}}$:

$$r = \cos^{-1}\left(\frac{\vec{\mathbf{V_d}}\dot{\vec{\mathbf{T_P}}}}{||\vec{\mathbf{V_d}}||||\vec{\mathbf{T_P}}||}\right) \tag{7}$$

Thus, we have the following problem: find b_i, d such that $F_1(b_i, d) =< f_1(b_i, d), f_2(b_i, d) >=< 0, 0 >$ where $f_1(b_i, d) = r$ i.e. the difference between the object azimuth and the final heading of the aircraft after flying the dead-leg defined by b_i, d, and $f_2(b_i, d) = e - h$ is the difference between the final object elevation and the telescope elevation limit e closest to the initial object elevation.

Now let us consider the case where the object violates the elevation limits at some point during the observation, regardless of whether or not it is initially visible. Using the geometric interpretation of the feasible region again, we see that the flight track exits the annulus (and possible re-enters it later on). In this case, we can set up a function very similar to that we used when the observation was initially outside the feasible region. We now want to find b_i, d such that $F_2(b_i, d) =< f_1(b_i, d), f_3(b_i, d) >=< 0, 0 >$, where $f_3(h, d)$ is the difference between the *extreme* object elevation achieved during the flight-leg and the telescope elevation limit violated during the observation. The intuition behind this is that the dead-leg we wish to fly should just barely nudge the observation inside the fea-

sible region. f_1 remains the same. Unlike the previous case, where we only needed to compute quantities like position and object elevation at fixed times, we now must find either the minimum or maximum of the elevation over the course of the flight-leg. We perform binary search over the ground track to find the extreme of the object elevation. Figure 2 shows a situation in which we would zero F_2 while searching for a dead-leg. Initially, the aircraft could not observe the object without the elevation exceeding the upper elevation limit, whose boundary is shown. However, it is possible to fly a 10 minute dead-leg to a new position, from which the maximum elevation achieved during the flight-leg does not exceed the elevation limits.

In both cases, we have now reduced the problem of finding the shortest dead-leg to the problem of finding a zero of a function, which can be solved efficiently using a variety of methods as long as F satisfies some simple conditions.

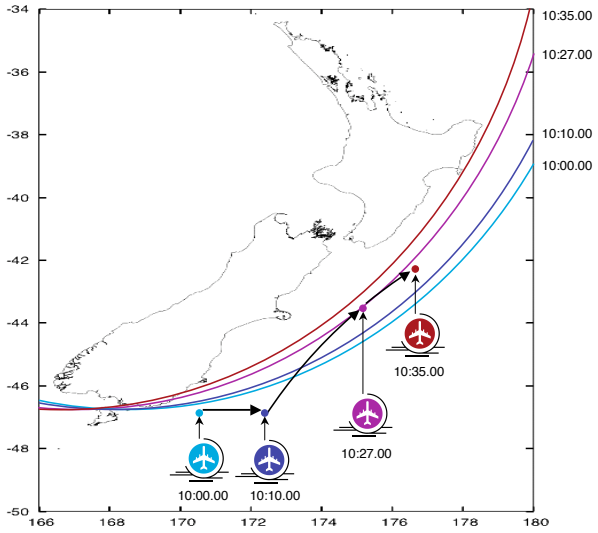

Figure 2: Flying a short dead-leg to enable an observation. The feasible region boundary shown is the upper coelevation limit. In this case we would zero F_2 to search for the dead-leg enabling this observation. The aircraft's initial location is shown at 10:00:00. The dead-leg lasts 10 minutes, after which the flight-leg begins. At 10:27:00 the object elevation achieves a maximum; the figure also shows the feasible region at 10:27:00, and shows that the elevation limits are not violated by the flight-leg. The flight leg ends at 10:35:00.

Properties of the dead-legs We now consider the zeros of the functions F_1 and F_2 and the dead-legs that are defined by them. The behavior of zero-finding algorithms depends on how many zeros there are and how they are distributed. Also, the resulting dead-legs may not be feasible given other constraints on how the aircraft flies that are not present in the definition of the zero-finding problems.

First of all, we observe that there are a countably infinite number of zeros of both F_1 and F_2. This is because we have imposed no restriction on b_i and d. This does not pose a serious problem; these zeros are widely separated, requiring that the aircraft fly all the way around the world multiple times. The dead-leg duration restriction imposed by the ForwardPlanner algorithm will eliminate long dead-legs. However, Newton's Method might not find the shortest dead leg, and either incorrectly conclude that some observation is not feasible or return a suboptimal dead-leg.

Also, not all zeros correspond to valid dead-legs. For example, a dead leg whose duration is negative is impossible for the aircraft to fly; similarly, a dead-leg whose duration exceeds the maximum allowed is forbidden. Also, short dead-legs may violate the minimum turn duration of the aircraft. A standard rate turn for a 747 is 180 degrees in 2 minutes. If the heading change and duration of the dead-leg violate this constraint, then the minimum dead-leg is impossible to achieve. Under these circumstances, Newton's Method would incorrectly report that an observation is infeasible.

Despite these potential drawbacks, we should point out that this method has two significant advantages over the brute force approach we used previously. First, we have imposed no limitations on the heading or durations of the dead-legs. Thus, we might find dead-legs we were unable to find before using this new method. Second, since zero-finding algorithms are usually quite fast, we hope that employing such a method will dramatically speed up the feasibility check, and therefore the flight planning algorithm overall.

Finding dead-legs By Zeroing

In this section we will describe how to find dead-legs by zeroing F_1 and F_2.

Newton's Method and Cramer's Rule

Newton's Method is our choice for finding the zeros of F_1 and F_2. It is simple to implement and very fast (GMW81). Newton's Method requires an initial guess for the zero; let this be denoted b_1, d_1 with future iterates denoted b_i, d_i . For functions F of 2 inputs and 2 outputs, the method proceeds as follows:

1. Compute $F(b_i, d_i) = \langle f_1(b_i, d_i), f_2(b_i, d_i)\rangle = \langle f_1, f_2\rangle$
2. Compute the Jacobian (matrix of partial derivatives):
$$J = \begin{pmatrix} \frac{\partial f_1}{\partial b}(b_i, d_i) & \frac{\partial f_1}{\partial d}(b_i, d_i) \\ \frac{\partial f_2}{\partial b}(b_i, d_i) & \frac{\partial f_2}{\partial d}(b_i, d_i) \end{pmatrix} \equiv \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$
3. Compute the determinant of J : $|J| = ps - qr$. If this is smaller than error tolerance t then set $|J| = t$ (preserving the sign).
4. Compute the Cramer's Rule update: $db = \frac{f_2 q - f_1 s}{|J|}$ and $dd = \frac{f_1 p - f_2 r}{|J|}$

5. Set $b_{i+1} = b_i + db$ and $d_{i+1} = d_i + dd$
6. If $\langle b_{i+1}, d_{i+1} \rangle \approx \langle 0, 0 \rangle$ or step limit reached, then halt, otherwise go to step 1.

Computing Derivatives Numerically

Directly calculating the derivatives of the functions F_1 and F_2 is difficult because of the gridded wind model that influences the ground track, which in turn influences the elevation (remember, the elevation is a function of time and position). Consequently, we use finite differencing to compute all of our derivatives numerically (GMW81). Of the available schemes, we chose forward differencing over centered differencing because of the smaller number of function evaluations required. We use two step size parameters s_1, s_2 in forward differencing. Suppose we are computing the derivatives at step i. Forward differencing for functions F of 2 inputs and 2 outputs proceeds as follows:

1. Compute $F(b_i, d_i) = \langle f_1(b_i, d_i), f_2(b_i, d_i) \rangle = \langle f_1, f_2 \rangle$
2. Compute $f_{1b} = f_1(b_i + s_1, d_i)$
3. Compute $f_{1d} = f_1(b_i, d_i + s_2)$
4. Compute $f_{2b} = f_2(b_i + s_1, d_i)$
5. Compute $f_{2d} = f_2(b_i, d_i + s_2)$
6. Compute $\frac{\partial f_1}{\partial b} \approx \frac{f_{1b} - f_1}{s_1}$
7. Compute $\frac{\partial f_1}{\partial d} \approx \frac{f_{1d} - f_1}{s_2}$
8. Compute $\frac{\partial f_2}{\partial b} \approx \frac{f_{2b} - f_2}{s_1}$
9. Compute $\frac{\partial f_2}{\partial d} \approx \frac{f_{2d} - f_2}{s_2}$

Note that more elaborate forms of numerical derivative computations are available. One reason for avoiding them is the number of calls to compute F_1 or F_2, which in this case requires constructing either flight legs, dead-legs or both. Since we want to minimize this cost, for the time being we stick with the simple forward differencing scheme.

Flying a dead-leg requires solving an IVP. Zeroing F_1 requires solving three IVPs per step of Newton's Method, since dead-legs must account for winds, and the finite differencing method requires evaluating the results of three different dead-legs. Zeroing F_2 requires solving six IVPs, three dead-legs and three flight-legs, and one function optimization step to find the elevation extremes per step of Newton's Method. The observation feasibility check requires solving one IVP to compute the ground track for the flight-leg when zeroing F_1, and one BVP to fly to the landing airport after zeroing either F_1 or F_2.

The Initial Guess

Algorithms like Newton's Method are highly sensitive to the closeness of the initial guess to the actual zero of the function. Newton's Method has *quadratic convergence* near a zero, which (roughly) means that the number of correct digits in the guesses doubles at each step. The brute-force dead-leg search performed previously does a blind search over possible headings and durations, so the number of correct digits in each guess improves by only a constant factor (at best) each step. Thus, using this methodology to find dead-legs should be an obvious performance win. However, we must make good initial guesses to benefit from rapid convergence.

Guessing the initial heading requires determining how an object's elevation is changing, and choosing the flight direction to make the elevation change correctly. Guessing the initial dead-leg duration requires estimating the difference in elevation that the dead-leg must achieve, and then estimating the rate of change of the elevation during the dead-leg. In both cases we construct a "test-leg", that is, we track the object for the desired duration whether it is with the elevation bounds or not. We use properties of this test-leg to decide both whether to zero F_1 or F_2, and make the initial guess.

Suppose the object always moving into the feasible region while it is outside the feasible region during the test-leg. For example, either the object is below the lower elevation limit and rising or above the elevation limits and setting. In this case we zero F_1. If the target is initially too high we want it to set faster. In this case we fly away from it, i.e. we guess $b_0 = 180° - A_f$. Similarly, if the object is initially too low, we want it to rise faster, so we fly towards it, i.e. we guess $b_0 = A_f$.

Now suppose the object is moving our of the feasible region at some point during the test-leg. We might also determine that this happens after successfully zeroing F_1. In these cases we zero F_2. If the object initially is rising, either it will rise continuously or eventually set. We want to make it rise slower initially, set faster later, or both. In any case, we want to fly away from the object, so we guess $b_0 = 180° - A_f$. If the object is initially setting, either it continually sets or sets then rises; we either want it to set slower, rise faster, or both. In any case, we want to fly towards the object, so we guess $b_0 = A_f$.

Only at high latitudes is it possible for an object to move into and then out of the feasible region; under these circumstances, we zero F_2.

Guessing the duration is somewhat more complex. Calculating the maximum required change in elevation Δh at the current position and time is simple once we have calculated the test-leg. However, we have to account for the rate of change of the elevation both as a function of time, and the change in position as the aircraft flies. Define r_e as the equatorial radius of the Earth, ϕ_p as the latitude component of the aircraft's location $\mathbf{p}$, and v as the aircraft's estimated groundspeed. We compute the instantaneous vectors of the aircraft's ground speed and Earth's rotation, then use the law of cosines to determine the aggregate effect on the object elevation, resulting in the following guess:

$$v_{rot} = \frac{2.0\pi r_e}{24.0\phi_p} \tag{8}$$

$$d_0 = \frac{\Delta h r_e}{\sqrt{v^2 + v_{rot}^2 + 2.0 v \sin(b_0)}} \tag{9}$$

We attempted to improve convergence when zeroing F_2 by first using Euler's Method with constant step size to construct the flight-legs. Once a zero of F_2 was found this way, we then used this as an initial guess and re-ran Newton's Method using Runge-Kutta to construct the flight legs. This did not improve convergence and was more expensive, and so we did not consider it further.

Matters of Convergence

Newton's Method depends on the function being zeroed to obey some properties to guarantee convergence. Our functions do not obey these properties all of the time, and so Newton's Method occasionally fails to converge.

Newton's Method is "non-local", in the sense that it can generate any point in $\Re^2$ during any step. Thus, if F is not defined on every element of $\Re^2$, Newton's Method may fail to converge to a solution even if one exists. F_1 and F_2 are not well defined for sufficiently short or long dead-leg durations. The problem with long durations is due to the built-in nature of the fuel model. Essentially, if a Newton step requires the aircraft to fly long enough that it would run out of fuel, we can't evaluate the ground track of the flight-leg. The problem with short durations has been explained above.

These factors mean that convergence of Newton's method may be interrupted if any intermediate step violates one of these conditions. This is a problem because it is conceivable that the zero found by Newton's method can correspond to a legitimate supporting dead-leg even if an iteration of Newton's method corresponds to a senseless dead-leg. If the function or the derivatives can't be evaluated during Newton's Method, our only option is to truncate the feasibility check and report that the observation is not feasible. Additionally, we could find Newton's Method failing to converge or converging after a large number of steps; we thus use a cutoff value to terminate search.

Empirical Results

In this section we describe experiments designed to test the value of using Newton's Method to speed up the feasibility check for ForwardPlanner.

Sample Problems

We used the Single Day problem instances from(FK03) to determine the utility of our new techniques. Figure 3 lists some salient characteristics of the Single Day Instances. We tabulate the number of observations, the archived flight duration, and the airport.

We first report on the change in the number of IVPs we must solve for problems $1, 2, 3, 5, 6, 43$ and 44. This covers flights from each of the airport sets. We generated 20 flight plans for each of these problems using ForwardPlanner and the restricted feasibility conditions described previously. We compared the brute-force dead-leg search approach to the use of Newton's Method for establishing feasibility. We fixed all other parameters of ForwardPlanner. We used a lookahead depth of 5, and each of the 4 heuristic features (priority, efficiency, distance to landing airport, and turning time) were weighted equally. ForwardPlanner sampled according to the heuristic 70% of the time, and randomly chose the next observation the rest of the time. The maximum dead-leg duration was set to 4 hours. For the brute-force search, we used a dead-leg duration increment of 1 minute and a heading increment of 7.5°. For Newton's Method we used a step cutoff of 150 and error tolerance $t = 10^{-6}$. The step parameters used in forward differencing were: $s_1 = 0.01°$ and $s_2 = 60$ seconds. Experiments were run on a Sun Workstation with dual 600 MHz CPUs and 2048 Mb memory. The aircraft takeoff weight was fixed at $210,000$ pounds of fuel for all flights. The same temperature and winds were used for all flights and the altitude was fixed at $35,000$ feet. These results are shown in Figure 4. Each leg-count cell contains the mean of the leg counts. The best plans found always contained all of the observations but the efficiency varied (time spent collecting data vs flight time) so we report that as well. Finally, we report the average CPU seconds to generate each plan.

Problem	Flight-legs	Dead-legs	Efficiency	CPU
1-B	366,416	365,108	0.788261	2,935.84
1-N	3,824.5	5,221.55	0.87923	60.3515
2-B	326,840	324,890	0.908678	2,585.04
2-N	5,854.4	5,577.75	0.862177	90.2115
3-B	326,396	323,615	0.930612	2,506.54
3-N	5,078.35	4,337.6	0.923638	69.3355
5-B	66,391.1	64,877.1	0.898895	590.346
5-N	2972.45	1737.6	0.8969.33	48.0125
6-B	222,304	223,116	0.863278	1781.72
6-N	4242.5	6158.9	0.858684	92.2705
43-B	226,353	223,023	0.910907	1787.43
43-N	6590.75	6563.6	0.812689	96.579
44-B	143462	141464	0.814273	1124.84
44-N	9316.85	9143.75	0.857633	159.142

Figure 4: Comparison of Newton's Method and Brute Force method of establishing observation feasibility on a small set of sample problems.

The results indicate Newton's Method dramatically reduces the number of IVPs to solve, and the computation time drops significantly along with it. The largest speedup is a factor of 48.9, while the smallest speedup is a factor of 7. While the resulting flight plans do not suffer in terms of the number of observations schedules, the story for the flight efficiency is different. Newton's Method found more efficient flight plans for problems 1,6 and 43, while the brute force method found more efficient flight plans for problems 2 and 44; for the others, the efficiency difference is negligible. Note that the performance gap can be narrowed significantly by increasing the duration increment used in the brute-force search, and by decreasing the maximum dead-leg duration, but this restricts the set of flight plans that can be generated.

We ran ForwardPlanner on all 47 problems in the test suite using Newton's Method. Again, we generated 20 flight plans for each problem. In only 6 cases were we unable to find a flight plan with all of the observations scheduled; in these 6 cases, only 1 observation was not scheduled. In no case did the generation of a single flight plan take more than 4 minutes, and frequently the time was 1-2 minutes.

We ran ForwardPlanner on selected problem instances in the test suite to analyze convergence of New-

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Airport	H	H	H	H	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	MH
# Obs	9	9	10	10	7	8	8	6	10	8	8	6	11	10	8	9	10	8	8	8	9	9	6	8

Index	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Airport	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	N	N	N	N	N
Obs	7	4	7	6	7	9	8	11	10	8	7	7	7	3	9	8	8	8	4	10	8	8	8	8

Figure 3: Characteristics of Single Day Instances.

ton's Method. This time we generated 5 flight plans for each problem. We used Newton's Method to try and establish feasibility of the observations. We record the outcomes: Newton's Method suceeded (O), failed to converge before the step limit was reached (C) or failed to zero F_1 and subsequently trying to zero F_2 (F), step failure because of a dead leg duration that was too short (including negative duration) (S) or too long (L). Finally, in the cases where Newton's Method failed, we used brute-force search to try and establish feasibility, and record the number of times when brute-force succeeded in enabling an observation when Newton's Method could not. We present this data in Figure 5.

We see all manner of failures of Newton's Method. The data indicate that dead-leg failure occurs roughly 10% of the time over these problems. The worst case in the samples we collected was for Problem 1, where roughly 12% of calls to Newton's Method to construct a dead-leg resulted in failure. Brute Force was sometimes able to construct a dead-leg when Newton's Method failed, with the worst case being arount 11% for Problems 5 and 6; however, this would increase the success percentage only a small amount.

Conclusions and Future Work

The work in this paper was motivated by an examination of the time spent by the ForwardPlanner algorithm in establishing observation feasibility. We have described a modification of the observation feasibility condition to reduce the number of IVPs and BVPs to solve. We show how this allows us to define functions whose zeros correspond to the properties of enabling dead-legs, and describe how these functions are zeroed. Empirical results indicate that ForwardPlanner is significantly faster on sample problems, and finds plans of comparable quality.

We can also *relax* the definition of feasibility and drop the check on the return to the landing airport altogether. The consequence of doing this is that a flight plan may violate the fuel constraint and either need to be repaired or rejected completely. This may be justified because throughout most of the planning process this condition is trivially satisfied. Relaxing the feasibility check by only ensuring that the observation is visible immediately after a dead-leg is a more dangerous proposition, because the frequency of high latitude observing almost ensures rejection sampling or repair will be needed.

Problem	O	F	L	E	S	Brute
1	21,784	0	1,527	6	1,686	32
2	30,487	48	1,853	22	738	53
3	20239	9	529	0	2,215	62
5	5,266	2	456	1	103	66
6	30,148	85	244	10	950	144

Figure 5: Analsysis of Newton's Method convergence for problems.

We can define an f_4 which accounts for the difference between the amount of fuel remaining to get the aircraft to the landing airport and the amount of fuel consumed by the leg home. Doing so would allow us to use the more restrictive definition of feasibility while paying only minimal overhead in the number of BVPs and IVPs that we must solve. However, no method of doing so we have yet discovered avoids the pitfalls described previously. Letting $f_4 = 0$ if the landing airport is reachable creates many situations where the first derivative of f_4 is zero, which is bad. Forcing $f_4 = 0$ only if exactly enough fuel is left to get to the landing airport is equally bad.

We intuitively define criteria to minimize the dead-leg duration (e.g. dead-leg ends with us flying towards the object). We have not proved this leads to the shortest dead-leg, even on spherical Earth with no winds. Since neither of these assumptions hold, other criteria might be better. Furthermore, the condition on finding the shortest dead-leg that makes an observation feasible is "locally optimal" in the sense that it is the best action to support one observation. Currently, we rely on repeated sampling to find good plans, but we know we only sample some of the possible plans, and may miss the best possible plan. We have considered post-processing the resulting flight plans, either using local search or by defining suitable functions to optimize, in an attempt to find better flight plans.

Acknowledgments

We would like to thank Karen Gundy-Burlet for her comments and insights. This work was funded by the SOFIA Projects Office and by the NASA Intelligent Systems Program.

References

J. Ferziger. *Numerical Methods for Engineering Applications.* John Wiley and Sons, 1981.

J. Frank and E. Kürklü. Sofia's choice: Scheduling observations for an airborne observatory. *Proceedings of the* 13^{th} *International Conference on Automated Planning and Scheduling*, 2003.

P. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, 1981.

J. Meeus. *Astronomical Algorithms.* Willmann-Bell, Inc., 1991.