Designing Noise-Minimal Rotorcraft Trajectories

Robert A. Morris
NASA Ames Research Center, CA (USA)
robert.a.morris@nasa.gov

K. Brent Venable
University of Padova, Italy
kvenable@math.unipd.it

James Lindsey
Monterey Technologies, CA)
NASA Ames Research Center(USA)
james.e.lindsey@nasa.gov

Abstract

The ability to predict rotorcraft ground noise is important in determining and assessing environmental noise impact. The noise generated by rotorcraft can limit their usage and restrict operations, particularly near cities and populated regions. The two primary approaches commonly used to reduce rotorcraft noise are to make vehicle design modifications and to make changes in operational flight procedures. The latter have the advantage that they can often be implemented to achieve significant noise reductions at a lower cost than new design efforts. Computer modeling capabilities for developing low noise procedures have received much attention over the last 15 years. These models, when paired with an automated optimization approach, can facilitate the design of new approach trajectories for improving the environmental impact. This paper describes recent work in applying a constraint-based optimization model and local search, paired with a robust noise simulator, to solve the noise minimal trajectory optimization problem.

Introduction

There is considerable interest by NASA and the commercial sector to develop a transportation infrastructure that is based on an increased use of rotorcraft, specifically helicopters and aircraft such as a 40-passenger civil tilt rotor. Rotorcraft have a number of advantages over fixed wing aircraft, primarily in not requiring direct access to the primary fixed wing runways. As such they can operate at an airport without directly interfering with major air carrier and commuter aircraft operations.

While it is possible to build civil aviation rotorcraft and tiltrotors of various sizes and capacities, the aviation industry and U.S. government officials are concerned with the impact of noise on the communities surrounding the transportation facilities. One way to address the rotorcraft noise problem is to design and test low-noise flight profiles which can be tested in simulation or through field tests.

The objective of this paper is to introduce a *Trajectory Noise Optimization Problem* (TNOP) for designing noise minimal rotorcraft approach trajectories. The model includes a graphical representation of the computational search space based on the state of the aircraft and the control decisions made by the pilot; a representation of constraints that identify trajectories that are 'flyable' based on pilot-elicited rules of comfort and safety of the aircraft; a noise simulator tool (Rotorcraft Noise Model, RNM) for calculating the effects of sound propagation over varying ground terrain, enabling the quantitative assessment of the overall ground noise produced by a given trajectory; two cost functions that aggregate and quantify the cumulative noise level to allow for trajectories to be compared and ordered based on the noise they produce; and an optimizing search approach using local search. The local search uses a neighborhood function based on a simple exchange of control decisions. The search is initialized using a seed solution manually crafted by a pilot based on standard approach procedures.

The remainder of this paper is organized as follows. A brief introduction to the quantification of rotorcraft noise is presented, followed by an introduction to the TNOP. We then describe the solving approach for finding noise minimal trajectories. Some preliminary results are presented and discussed.

Background

Introduction to Noise and how it is Measured

Noise is unwanted sound. Sound is variation in air pressure detectable by the human ear in the form of vibration of the ear drum. The decibel is a ratio that compares the sound pressure of the sound source of interest (e.g., the rotorcraft overflight) to a reference pressure (the quietest sound we can hear). Humans can detect sound pressure over a wide range, 10^{-9} to 10^{-3} pounds per square inch (psi). Because the range of sound pressures is very large, we use logarithms to simplify the expression to a smaller range, and express the resulting value in decibels (dB).

Sound can be broken down into frequencies (low, medium, high). The ear is more sensitive to mid- and high frequency sounds, so we find noise in these ranges more annoying. The so-called *A-weighting* approximates the sensitivity of the human ear and helps to assess the relative loudness of various sounds.

Sound levels vary with time, which is important if we are interested in the noise associated with a certain event of interest (e.g. an approaching rotorcraft). To take exposure du-

ration into account, the most common measure is the *Sound Exposure Level (SEL)*. *SEL* 'summarizes' the variable energy level of an event with arbitrary duration by mapping it to an event of one second duration with the same overall energy and a constant energy level. *SEL* provides a comprehensive way to describe noise events for use in modeling and comparing noise environments. Computer noise models base their computations on *SEL* values.

The US Federal Aviation Administration (FAA) considers a 1.5 dB the minimum significant change where cumulative exposure is above 65 DNL. Any abatement strategy that promises over 5 dB change in noise level is considered definitely beneficial. As we show later, we will use this value in assessing and comparing noise cost functions for trajectories.

Helicopter noise sources include the main rotor, the tail rotor, the engine(s), and the drive systems. The most noticeable acoustical property of helicopters is the modulation of sound by the relatively slow-turning main rotor. The resulting sound can become impulsive in character and is referred to as BVI (Blade Vortex Interaction Noise). Impulsive noise occurs during high-speed forward flight as a result of blade thickness and compressible flow on the advancing blade. This causes the blades airloads to fluctuate rapidly. These fluctuations result in impulsive noise with shock waves that can propagate forward. At lower airspeeds, and typically during a descent, rotor impulsive noise can occur when a blade intersects its own vortex system or that of another blade. This type of noise is BVI noise. When this happens, the blade experiences locally high velocities and rapid angle-of-attack changes. This tends to produce a sound that is loud and very annoying in character (Fly, 2009), (Greenwood and Schmitz, 2010).

Rotorcraft Noise Simulation

The Rotorcraft Noise Model (RNM)(Conner et al., 2006) is a simulation program that predicts how the sound of a rotorcraft will propagate through the atmosphere and accumulate at observer (receiver) locations. RNM is capable of calculating cumulative noise exposures such as A-weighted SEL. The input to RNM consists of

- a set of computational parameters, including identity of rotorcraft, and the dimensions and resolution of a grid that will display output noise (discussed further below);
- a specification of points of interest; and
- a specification of the flight trajectory, including position, velocity and orientation.

RNM contains a model of how sound propagates through the atmosphere. In general, the noise that propagates from a source to a receiver at a given distance from the source can be expressed as a sum of the following factors:

- the sound level at the source;
- the geometrical spherical spreading loss (since energy is a conserved quantity, the energy per unit area (intensity) of an expanding spherical pressure wave decreases as $1/r^2$. This is called spherical spreading loss, which obeys an inverse square law.)
- loss due to atmospheric absorption effects, based on theoretical predictions and experimental data;
- ground reflection and attenuation (including the effects of terrain); and
- effects due to wind.

RNM allows for there to be multiple sources of noise from the same rotorcraft.

Noise data either experimentally or analytically generated from models is stored in the form of a *sound sphere*. Points on the sphere are described in terms of a radius from the source and two spherical angles. A sphere is associated with one noise source and one flight condition (flight path angle, nacelle angle (for tilt-rotors) and airspeed). There may be more than one sphere for the same flight condition; for example, one sphere for different locations on the rotorcraft. Figure 1 shows an example sphere (actually, a hemisphere).

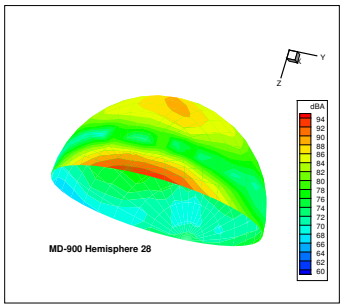

Figure 1: Example of sound hemisphere of an MD-900 helicopter.

There are three main computational components of the RNM simulation:

- *Input Module*: Linear interpolation over the input trajectory as a pre-processing step. Input data are interpolated (if required) to a default of 2 second spacing.
- *Source Database Lookup and Selection*: Selecting and interpolating over the sound spheres to determine the best representative of the noise generating for a given location and flight condition in the input trajectory; and
- *Source to receiver propagation*: Accumulating and storing the sound for a given receiver.

The second and third components in the list are repeated for each trajectory point, sound source, flight operation and receiver location.

RNM simulation produces predictive noise data in various formats. Of interest to our work, is the generation of *ground noise contour plots*, a set of values representing ground noise exposure using A-weighted SEL or other metric over a designated grid of x-y points around the evaluated trajectory. Figure 2 shows an example of such a plot, where each color corresponds to a dB level (redder and lighter colors noisier). These plots provide the data used to compute the aggregate cost functions used during local search, as discussed below.

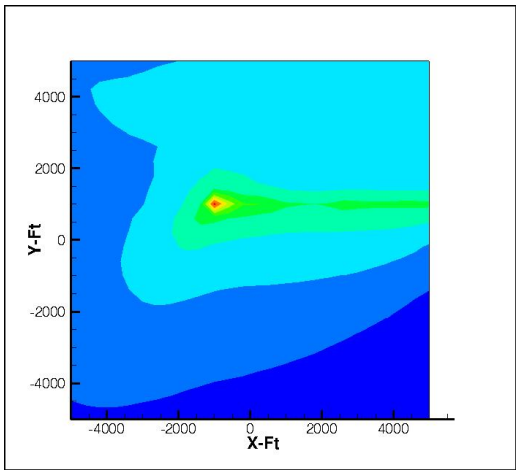

Figure 2: A Noise Contour Plot.

Trajectory Optimization

The field of trajectory optimization has a long history, with many applications in aerospace and robotics. The basic description of the problem, which we adapt here, is stated informally as follows: *given a set of states and control actions, find a sequence of actions (trajectory) that minimizes a cost function subject to a set of dynamic constraints, and constraints on start- or end-states.* In addition to noise, trajectories have been optimized with respect to time, fuel, path length and obstacle avoidance.

Methods of solving trajectory optimization problems range from numerical methods (Betts, 1998) to non-linear programming problems (Goplan et al., 2003) or dynamic programming (Hagelauer and Mora-Camino, 1998). In addition, path planning methods from robot motion planning has been used (P. Cheng and LaValle, 2001). Randomized optimization methods such as simulated annealing and genetic algorithms have also been applied (Xue and Atkins, 2006).

Local Search

Local search (Hoos and Stutzle, 2004; Aarts and Lenstra, 1997) is one of the fundamental paradigms for solving computationally hard combinatorial problems. Given a problem instance, the basic idea underlying local search is to start from an initial search position in the space of all possible assignments (typically a randomly or heuristically generated assignment, which may be infeasible, sub-optimal or incomplete), and to improve iteratively this assignment by means of minor modifications. At each *search step* we move to a new assignment selected from a *local neighborhood*, chosen via a heuristic evaluation function. The evaluation function typically maps the current candidate solution to a real number and it is such that its global minima correspond to solutions of the given problem instance. The algorithm moves to the neighbor with the smallest value of the evaluation function. This process is iterated until a *termination criterion* is satisfied. The termination criterion is usually the fact that a solution is found or that a predetermined number of steps is reached, although other variants may stop the search after a predefined amount of time. Different local search methods vary in the definition of the neighborhood and of the evaluation function, as well as in the way in which situations are handled when no improvement is possible. To ensure that the search process does not stagnate, most local search methods make use of random moves: at every step, with a certain probability a random move is performed rather than the usual move to the best neighbor.

In hill-climbing search (Selman and Gomes, 2006), we select any local change that improves the current value of the objective function. Greedy local search is a form of hill-climbing search where we select the local move that leads to the largest improvement of the objective function. Traditionally, one would terminate hill-climbing and greedy search methods when no local move could further improve the objective function. Upon termination, the search would have reached a local, but not necessarily global, optimum of the objective function. In recent years, it has been found, perhaps somewhat surprisingly, that simply allowing the local search to continue, by accepting 'sideway' or even 'downhill' moves, i.e. local moves to states with, respectively, the same or worse objective values, one can often eventually still reach a global optimum.

Trajectory Optimization Problem Formulation

The *trajectory noise optimization problem* (TNOP) is the problem of designing flight approach trajectories for rotorcraft that minimize the cumulative ground noise exposure from the vehicle. We focus on approach trajectories (and the nearly identical problem of take-off) because that is where all the community noise problems arise. We will focus on A-weighted SEL as our noise exposure metric. RNM simulation provides a black box scoring function for candidate trajectories. Specifically, RNM produces an output file that assigns predicted noise for a set of ground points arranged in a two-dimensional grid on the X-Y plane (Figure 3). The grid size is defined in terms of the values of the corner nodes and the distance between nodes.

Upon this grid our model superimposes an organization of nodes associated with the state of the aircraft and the control decisions being made by the pilot. More formally, each *node* n at position x_n, y_n is associated with two vector variables:

- the state vector: $s_n = \langle x_n, y_n, z_n, v_n \rangle$, where z_n is altitude and v_n is speed (velocity). Since x_n and y_n are fixed we will omit them and write $\langle v_n, z_n \rangle$. Intuitively, they represent the altitude and speed of the rotorcraft when flying over position (x_n, y_n);
- the control vector: $d_n = \langle \Delta v_n, \Delta z_n \rangle$, where Δv_n represents a decrease in speed, and Δz represents a decrease in altitude. Intuitively the control vector represents the changes in speed and altitude that will be applied to the rotorcraft starting from node n.

As shown in Figure 3, all pairs of horizontally, vertically or diagonally adjacent nodes are connected by an edge. We assume that each edge has a value representing a distance between the nodes and that the rotorcraft can only move along

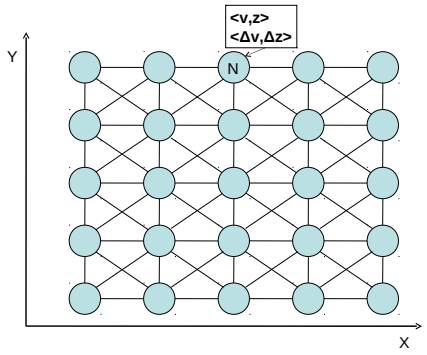

Figure 3: Two dimensional grid of the X-Y space.

edges. Moreover, we do not model turns directly and we assume that the rotorcraft can turn instantaneuosly at any node, involving no action.

In general, we can define a distance D between arbitrary pairs of nodes along a path, as the sum of the distances of the edges in the path. An edge implicitly represents two directed arcs (one for each direction).

We define a $trajectory$ as path on the grid plus the assignment to all the state vectors and control vectors of the nodes traversed by the path. Given a trajectory t through node n, with vectors s_n and d_n, we denote the assignment to the vectors of n in t as $val(t, s_n) = \langle z_{t,n}, v_{t,n} \rangle$ and $val(t, d_n) = \langle \Delta z_{t,n}, \Delta v_{t,n} \rangle$. A trajectory t is said to be *consistent* if, for every pair of consecutive nodes p and n, where p precedes n, we have that the value of $val(t, s_n)$ is the result of applying the control actions $val(t, d_p)$ in state $val(t, s_p)$. More precisely,

$$v_{t,n} = v_{t,p} + \Delta v_{t,p} \ , \ z_{t,n} = z_{t,p} + \Delta z_{t,p}.$$

We will be searching for flyable trajectories that minimize noise. Conditions that make a trajectory suitable to fly are, however, usually expressed in terms of constraints over the descent angle and deceleration. In particular, any part of a trajectory should be characterized by an angle of descent $\gamma \in [0^o, 10^o]$ and a deceleration $a \in [0g, 0.1g]$ (or $a \in [40ft/sec^2, 201ft/sec^2]$). Such restrictions induce constraints on the domain of Δv and (Δz as follows. Given a pair of nodes n_i, n_j and a path between them of distance D we have:

- the deceleration constraint: $Dom_{D,v_i}(\Delta v_i) = \{\delta_v \| \exists a \in [0, 0.1], \delta_v = \sqrt{v_i^2 + 2a \times D} - v_i\}$, where a is expressed in gs.
- the angle-of-decent constraint: $Dom(\Delta z_i) = \{\delta_z \| \exists \gamma \in [0^o, 10^o], tan(\gamma) = \frac{\delta_z}{D}\}$.

A trajectory is said to be *flyable* if it satisfies all the deceleration and angle-of-decent constraints along its path.

In our setting we are given two nodes designated as start and finish, with fixed state and control vectors, and a solution is any consistent flyable trajectory between them. To control the size of this space we initially start by limiting the paths to those that would be considered 'standard' by pilots. One example of a standard approach is a box pattern, as the one shown in Figure 4.

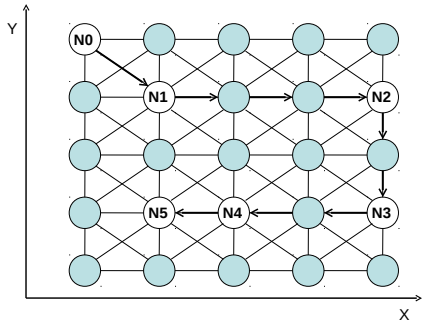

Figure 4: A "box"-like approach pattern.

A box pattern can be represented by a sequence of 6 nodes $N_0 \ldots N_5$ where turns take place at nodes N_1, N_2 and N_3, N_0 is the start of the approach and N_5 is the landing point. Given a box pattern, say $(N_0, \ldots, N_5)$, our goal is to find an assignment, say $(val(s_1), \ldots val(s_5), val(d_0), \ldots val(d_5))$, to the state and control vectors of the nodes not fixed by the initial and final conditions, such that the noise simulated by RNM on the corresponding trajectory minimal.

It is thus important to define a way to evaluate the overall noise of a trajectory. In order to do so, we propose the following two heuristic functions.

Binning Heuristic function Given in input a solution t, RNM computes the A-weighted SEL value for each of the grid points. Let us denote with $SEL(t, x, y)$ such a value for the grid point (x, y) given trajectory t. We define a sequence of decreasing ranges, $\langle r_1, r_2, \ldots, r_n \rangle$ partitioning the SEL values of the grid points. Given a trajectory t let us denote by $S_i(t) = \{(x, y) | SEL(t, x, y) \in r_i\}$. We define the following vector $b(t) = \langle b_1(t), b_2(t), \ldots, b_n(t) \rangle$ where $b_i(t) = |S_i(t)|$. The bin-score of solution t is

$$Bin(t) = \Sigma_{i=1\ldots n} w_i b_i(t)$$

where w_i is the weight associated to the i-th bin, $w_i > w_{i+1}$ and $\Sigma_{i=1,\ldots,n} w_i = 1$. The intuition behind this function is that of evaluating a solution by how it partitions the grid points into regions of different levels of noise. Thus a solution that assigns lower levels of noise to larger regions of the grid is to preferred. Weights are used to model this and to further penalize the presence of, even small, extremely noisy regions. Given this heuristic function the goal is to minimize its value.

Significant Improvement Heuristic function Let s denote a reference solution and t another solution. Then the significant improvement score of t w.r.t. s is

$$SI(s, t) = \\ |\{(x, y) \| SEL(s, x, y) - SEL(t, x, y) \geq 1.5dB\}| \\ - |\{(x, y) \| SEL(t, x, y) - SEL(s, x, y) \geq 1.5dB\}|.$$

In other words, this heuristic function considers a reference solution (that, in our case will be seed solution of the local search), and then scores all other solutions counting the number of grid points where they produce a noise that is at least 1.5dB lower than the one produced by s at the same point. As noted earlier, the 1.5dB threshold has been chosen since it is the smallest improvement that can be perceived by a human. The intuition behind this heuristic function is that of promoting solutions that improve significantly in the largest number of grid points. Given this heuristic function the goal is to maximize its value.

This paper does not present any results on the use of this heuristic, but future work will show the results of comparing it with the binning approach to evaluating noise.

Box-TNOP-HC(Trajectory σ_{seed}, function $score$, integer $threshold$)
$\sigma_{cur} = \sigma_{seed}$ // current trajectory
$\sigma_{best} = \sigma_{seed}$ // best incumbent trajectory
$step = 1$
do
  $\sigma_0 = neighbor(\sigma_{cur})$
  $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  while $neighborhood(\sigma_{cur}) \neq \emptyset$ and $score(\sigma_0) \leq score(\sigma_{cur})$
    $\sigma_0 = neighbor(\sigma_{cur})$
    $neighborhood(\sigma_{cur}) = neighborhood(\sigma_{cur}) \setminus \{\sigma_0\}$
  $\sigma_{cur} = \sigma_0$
  if $flyable(\sigma_{cur})$ and $score(\sigma_{cur}) > score(\sigma_{best})$
    $\sigma_{best} = \sigma_{cur}$
  $step++$
while $step \leq threshold$
return σ_{best}

Neighbor(Trajectory σ)
1 $n = random(\sigma)$ // randomly pick a node
2 $p = partner(n)$ // choose adjacent node, either forward or backward
3 *select* $c \in \{\Delta v, \Delta z\}$ // change rate of deceleration or descent
4 $v_c = val(c, p, n)$ // find a an allowable value to transfer
5 $\sigma_n = transfer(n, p, v_c, \sigma)$ // add the value to p and subtract from n
6 $(n, p, c) = used$ // mark triple as used
return σ_n // return the neighbor

Figure 5: Greedy Local Search Algorithm

Local Search for Box-TNOP

The technique we propose here to solve the optimization problem described in the previous section is a hill-climbing local search approach. The reasons for preferring local search include:

1. *Anytime performance:* On average, local search behaves well in practice, yielding low-order polynomial running times (Aarts and Lenstra, 1997). Since the trajectory space is large, it is difficult *a priori* to characterize globally preferred solutions. Consequently, we are interested in a system that can examine large parts of the search space quickly.
2. *Flexibility and ease of implementation:* deployment-related deadlines suggest the use of techniques which are easy to implement.
3. *Simulator Compatibility:* running RNM is heavy from a computational point of view. This means that the repetitive evaluation of partial trajectories, required by complete incremental solving paradigms (e.g. Branch and Bound), may be unacceptably time consuming. Local search, on the other hand, only requires the evaluation of complete solutions.

Figure 5 describes the pseudocode of our algorithm, which we call *Box-TNOP-HC*. The inputs to the algorithm are

- a seed solution σ_{seed};
- a scoring function $score$ that can be either Bin or SI;
- a positive integer $threshold$, representing the number of search steps after which the execution must terminate.

We note that the box trajectory is implicitly represented in σ_{seed}. Moreover, since in our case there is no way to test if an optimal solution as been found, the algorithm will always run for $threshold$ number of steps.

The output of *Box-TNOP-HC* is a solution denoted by σ_{best}. During the execution we keep track of the current solution, the neighborhood of which we are exploring, denoted by σ_{cur}, and the best flyable solution found so far, denoted with σ_{best}. Both such solutions are initially assigned the seed solution. Then, the algorithm starts exploring the neighborhood of σ_{cur}. As soon as it finds a solution that is better than the current one, it checks if it is flyable and if so it saves as the best incumbent. *Box-TNOP-HC* then updates σ_{cur} and starts scanning its neighborhood. Whenever no better solution is found, a random move in the neighborhood is taken.

Neighborhood Function

So far our procedure is a standard hill-climbing local search where stagnation in local optima is avoided by random moves in the neighborhood. The key aspect is, of course, on how the neighborhood is defined. The intuition guiding the design summarized here is that a neighbor of a trajectory σ is the result of applying operators that alter the order and magnitude of the decrease in speed or altitude at two adjacent nodes of σ.

More formally, from Figure 6, to generate a neighbor of the illustrated trajectory, a node $N_i, i = 0 \ldots 4$ is chosen at random (the final node N_5 being unalterable), and a partner (N_{i-1} or N_{i+1}) and a control variable, Δv_i or Δz_i, is chosen. Where, v_i is the current value of the chosen variable at N_i, a value $0 < v'_i \leq v_i$ is computed and *transferred* to its partner; that is, v'_i is added to the partner and subtracted from the value of the originating control variable associated with N_i.

Assuming a trajectory of length (number of nodes) L, the size of such a neighborhood can be determined as follows. First, any neighbor can be defined by the triple (n, p, c) used to generate it, where n and p are a node and its partner, respectively, and c is the control variable Δv or Δz. So the size of the neighborhood is the number of such triples. Assuming the final node cannot be altered, only $L - 1$ nodes can have any partners. Since there are at most 2 partners and 2 control actions, a given node can participate in at most 4 triples. But

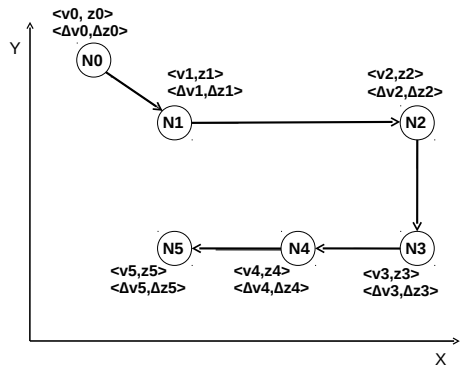

Figure 6: A "box"-like approach trajectory.

the 'fringe' nodes, the first, and the $L-1$st, only have one partner, so only participate in 2 triples. The remaining $L-3$ nodes participate in 4 triples; so the neighborhood size is $4(L-2)$. For example, the size of the neighborhood of any trajectory with the box structure of Figure 6 is $4(6-2) = 16$.

A triple (n, p, c) will be called 'used' if this combination was used during local search to generate a neighbor, and the neighborhood of a trajectory is empty if all the triples have been marked as used. In selecting the triple, only the node is generated randomly; the others are selected in an arbitrarily chosen order, forward partner before previous, Δv before Δz (lines 1-3 of **Neighbor**, Figure 5).

Once a triple is selected, a suitable value v_c to transfer between a node and its partner must be determined in order to preserve the feasibility of the new trajectory. We first experimented with a simple 'swap' routine, where control values were exchanged between nodes. The main problem with simple approaches like swapping is that the result of an arbitrary swap is usually an infeasible trajectory due to the violation of constraints related to rate of descent or deceleration, as a result of the varying distances between nodes. For example, a Δz that can be accomplished smoothly in 10000 ft might be too uncomfortable or even unsafe if attempted in 2000 ft (see Figure 6). This led to a more detailed consideration of the conditions that allow quantities to be transferred between nodes. The problem can be viewed as similar to planning with resources, with capacity and usage requirements.

Specifically, for each control variable, the amount of control that can be transferred between nodes is determined by two factors: how much additional rate (deceleration or descent) can be added to a node, and how much loss of velocity or altitude is allowed for the state associated with a node. Intuitively, for safety and/or comfort reasons, a pilot will not tolerate too much deceleration or descent over a given segment; nor will the pilot allow the craft to fly too slowly or too close to the ground at certain distances from the landing site.

Consequently, line 4 of **Neighbor** attempts to find the largest value v_c that can be transferred between a node and its partner that does not violate these safety and comfort constraints. First, each node is associated with a upper bound on the $\Delta v, \Delta z$ that it can support, related to the distance between it and its forward adjacent node. The value transferred to it by either its forward or backward partner cannot exceed this value. Secondly, each node is associated with a minimal velocity and altitude ($vmini, zmini$) it must support. No transference of $\Delta v, \Delta z$ can result in a state associated with this node in which v or z go below these values.

Experiments

In this section we present some preliminary experimental results on the performance of local search in solving the TNOP. We explain the rationale for the experimental design, the tests we have run to this point, and summarize and interpret the results. Some of the designs have provided inputs to field tests that will be conducted by NASA at Eglin Air Force Base in Florida in June, 2011.

Data Resolution

It is important to emphasize the distinction between the number of grid points in a TNOP instance and the number of nodes. A grid point represents a data point for our cost functions, a point on the ground at which a noise prediction is made by the simulator. A grid point may correspond to a node (if they share the same x-y values) but not necessarily (one may choose to define a node that is outside of the grid if he or she does not care to measure the noise at that point). The *data resolution* of the problem is the number of distinct data points used to evaluate trajectories, and is also something the designer can vary. Clearly, there is a computational cost incurred by a high resolution, both during the simulation, and in the post-processing within the cost functions. Also not surprisingly, this burden is felt more intensely in the simulation than in the post-processing, where more complex math is performed.

The data resolution is an input parameter to RNM: one specifies the x-y values for the bottom left and upper right corners of the grid, and specifies the distance in feet between any pair of nodes. For these tests, we've looked at resolution in the range between roughly 75 and 22000 data points. Table 3 shows a sample of resolutions, as well as the cost in time for using the resolution in evaluating trajectories. The columns of the table show distance between grid elements, the number of data points generated (for all the trajectories explored in this paper) and the time it takes to run RNM on a single trajectory (with 7 nodes) and generate a score (which involves constructing the bins and performing the weighted sum). The important thing to note about the table is that too high a resolution (100 or 50 grid distance) makes it prohibitively expensive to sample a large number of trajectories during local search.

Search Parameters

The critical parameters in any local search are the threshold size and the number of restarts. We are also interested in finding a set of weights and bin ranges to the binning cost function that makes sense and provides useful results. Although we can't claim to have achieved a unique perfect

Table 1: Data Resolutions

Grid Distance	# Points	Speed per Evaluation (sec)
50	22,761	122.3
100	5781	31.2
200	1497	8.2
400	396	2.3
1000	75	1.2

setting for these parameters (if indeed one exists) we here state some preliminary results on some of these meta-issues.

Bin ranges and Weights In specifying bin ranges, we tried to provide a characterization of 4 qualitative noise categories: very noisy, noisy, moderately noisy, and low noise. We chose SEL-A ranges $\langle[115, +\infty], [100, 114], [85, 99], [-\infty, 84]\rangle$. Weights were chosen as an indicator of how important it is to reduce noise in that bin range. Somewhat surprisingly, it was decided not to weigh the loudest range significantly higher than the others, but rather to prefer the more egalitarian weight distribution $\langle 0.4, 0.3, 0.2, 0.1\rangle$ for each of the bins, respectively. The reason for this assignment is that the loudest reflects a location close to, or on, the landing field, and it is not important (or indeed possible) to lower the noise over such non-residential areas.

Trajectory Size Intuitively, the number of nodes (trajectory size) corresponds to the number of distinct control decisions a pilot will be required to make to execute the trajectory during approach and landing. Pilots typically make 5 or 6 control actions during this flight segment, and for reasons of safety it is not desirable to add significantly to the pilot's control actions. Nonetheless, as on-board automated control technology improves, the constraint on the number of nodes based on pilot decisions will become less important, and so it is not in general an important factor in our TNOP design. Nonetheless, because we were interested in submitting trajectories to be validated on a field test with little automated control available, we have here focused on smaller trajectory sizes; specifically, for the experiments here, we are looking at trajectory size of 6.

Initial Seeds For these experiments we (one of us being a helicopter pilot) generated 'standard' approach trajectories to be used as seeds. Figure 7 describes graphically one of the seeds used in the experiments. The 'box' has three segments: the downwind segment (the upper horizontal line) of length 7500 feet, the base segment (the vertical line) and the final approach segment (the lower horizontal line with 4 nodes). The final approach segment is the most important to examine, because they usually produce the most noise. Typically they are executed at a constant rate of descent and deceleration, where rate of descent is measured in terms of *glide slope angle* (γ) ranging from $3 \leq \gamma \leq 9$ degrees, although for the experiments the designers wanted to consider γ up to 12 degrees. Figure 7 shows a seed with a 9 degree final approach. In our local search, we start with seeds that are constant in deceleration and rate of descent, but are allowed

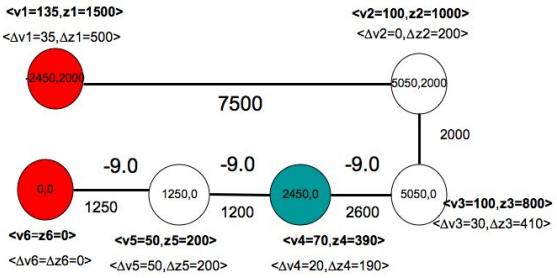

Figure 7: A Seed solution.

to produce solutions that vary in both during final approach. In general, since the goal of the research is to design new trajectories that are lower in noise, it is reasonable to seed our optimizer with standard trajectories.

Table 2: Comparison of Seeds

# Points	Angle	Seed	Best e
1497	6	309.9	305.2
	9	285.5	265.2
	12	263.1	256.1
396	6	81.9	60.6
	9	75.7	69.3
	12	69.5	67.3

We first wanted to compare the noise quality of the different seeds at different resolutions, and the ability of local search to seek alternative trajectories with lower noise. Table 2 shows some results for two different resolution settings (grid distances 200 and 400 feet). The trajectory with the 12 degree final approach scored better than the 9 degree or 6 degree final approaches, due no doubt to the less ground traversed by a steeper descent. The Best score was obtained by running local search with threshold of 50 and 5 starts, taking the best score among the five runs. With the higher resolution, the most improvement was found with the 9 degree approach, whereas in the lower resolution the 6 degree seed produced the most improvement, finding the best score among all the seeds.

It should be pointed out that each of the seeds has advantages other than noise to consider (the 12 degree approach is at the upper edge of comfort and safety limits), so the point is not to choose among them, but to compare them on one criterion, noise. It is also important to note the different run time behaviors of local search on the three seeds (the statements that follow are meant to be anectodal observations, to be verified by collecting more statistics). Running with 6 degree seed with the same local search settings (i.e. thresh-

old and restart number) is always slower than either 9 or 12 degree (with the same box structure). The reason seems almost certainly to do with the number of feasible neighbors. RNM is only run if the neighbor of a trajectory is feasible. 12 degree seeds don't have many feasible neighbors, because 12 degree is the upper bound of feasible γ and so applying the transference almost always produces infeasible neighbors. By contrast, 6 degree seeds tend to have many feasible neighbors, which means RNM is called much more often to evaluate candidates, thus making the local search procedure slower. Only a small percentage of the 6 degree candidates selected improve the noise cost, for reasons still under investigation. One possible explanation is that the 6 degree seed has constant deceleration and descent, and this might be in general the best (in terms of noise) final approach. The run time behavior of the 6 degree seed is therefore dominated by the need to run RNM for many or all of a neighborhood before the step size is incremented. A 9 degree seed tends to have somewhat fewer feasible neighbors, but more of these are quieter approaches; thus the 9 degree seed is "in the middle" between 6 and 12 with respect to run time behavior.

Anytime Behavior An optimal threshold setting is partly determined by how quickly, on average, the algorithm converges to an optimal solution. Figure 8 shows the result of running local search on the three initial seeds (6 9 and 12 degree glide slope) with a threshold of 200, for a data resolution of 396 data points, and recording the score of the best solution found in increments of 25. The value plotted represents the average over 10 runs.

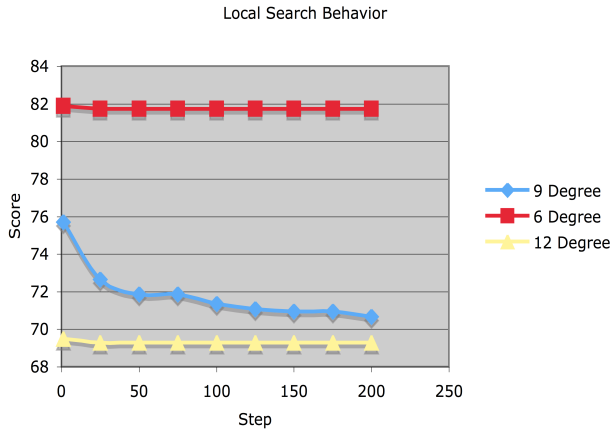

Figure 8: Runtime behavior of *Box-TNOP-HC* starting with 3 "standard" seeds.

For 6 and 12 degree seeds, convergence to the local optimal occurs by step 25. For the 9 degree seed, convergence is not complete at step 25, but rather there is a slow improvement throughout the runs. Nonetheless, local search on TNOP tends to find most improvement early in the search, which suggests a strategy with more restarts, higher resolution, but a smaller step size. We plan other tests to confirm that the same behavior occurs at all resolutions.

Variable Resolution Search We explored the possibility that local search with lower and higher resolutions could be effectively combined. Starting with the initial standard seed, local search with low resolution (in this instance, 75 data points) could explore larger areas of the solution space. The best solution found in the low resolution search could then be used as a seed for higher resolution search (here, 5781 data points), which allows for greater discrimination of solutions, but is more focused.

Table 3 compares constant and variable search. With constant Resolution, local search is run with high resolution only, with threshold of 100 and 5 starts, with the 9 degree seed trajectory. With variable resolution, the same seed was run first with low resolution, for threshold of 100 and 50 starts. The best trajectory found becomes the seed for a more limited higher resolution search, with a threshold of 50 and 5 starts. The result, for this instance, is the same solution at a lower time. It should be strongly stressed that this simple example is only suggestive of the value of the variable resolution approach, and more experiments are needed to determine whether, and the extent to which, variable resolution produces better results, and allows us to use a limited amount of higher resolution local search.

Table 3: Local Search With Constant and Variable Resolution

	Best Score	Time (sec)
Constant Resolution	1006	2323
Variable Resolution	1006	1059

Conclusions and Future Work

This paper describes a constraint-based optimization model for the problem of minimizing the noise of approach trajectories for rotorcrafts. We show how such a model is a suitable representation when a local search based approach is used to find better solutions. The presented framework is appealing since it allows to incorporate constraints representing the knowledge provided by experts (such as pilots) and provides a structured and straightforward way to embed simulation results within search. Experimental results, while preliminary, look promising and suggest this line of research as having the potential for providing significant improvements concerning rotorcraft noise minimization.

An extensive experimental scenario is the first item on our agenda. We also plan to design new heuristic functions to be used within the hill-climbing schema as well as to investigate the possibility of systematic search. In the near future we plan to incorporate turns, either directly into search or as modifications of the solutions found by our system. Other, less imminent, issues regard incorporating information on different sensitivity levels for different areas around heliports and the embedding of our system with on-board guidance tools.

Acknowledgements

We would like to thank Eric Greenwood (NASA Langley Research Center) for the extremely helpful insight he has provided on matters regarding acoustics theory. Thanks to Sharon Padula (NASA Langley Research Center) for helping us to understand and use RNM, and for imparting knowledge gained from her previous work on trajectory optimization. The first author would like to thank David Smith for helpful feedback and suggestions for designing the neighborhood function.

References

(2009). Fly neighborly guide. Technical report, Helicopter Association International.

Aarts, E. and Lenstra, J. K. (1997). *Local Search in Combinatorial Optimization*. Princeton University Press.

Betts, J. T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control and Dynamics*, 21(2):193–207.

Conner, D. A., Burley, C. L., and Smith, C. D. (2006). Flight acoustic testing and data acquisition for the rotor noise model (rnm). In *Proceedings of the 62nd Annual Forum of the American Helicopter Society*, pages 1–17.

Goplan, G., Xue, M., Atkins, E., and Schmitz, F. H. (2003). Longitudinal-plane simultaneous non-interfering approach trajectory design for noise minimization. In *Proceedings of the 59th AHS International Forum and Technology Display*, pages 1–18.

Greenwood, E. and Schmitz, F. (May 11-13, 2010). A parameter identification method for helicopter noise source identification and physics-based semi-empirical modeling. In *American Helicopter Society 66th Annual Forum*, Phoenix, AZ.

Hagelauer, P. and Mora-Camino, F. (1998). A soft dynamic programming approach for on-line aircraft 4d-trajectory optimization. *European Journal of Operational Research*, 107:87–95.

Hoos, H. H. and Stutzle, T. (2004). *Stochastic Local Search: Foundations and Applications*. Elsevier - Morgan Kaufmann.

P. Cheng, S. Z. and LaValle, S. M. (2001). rrt-based trajectory design for autonomous automobiles and spacecraft. *Archives of Control Sciences*, 11(3-4):167–194.

Selman, B. and Gomes, C. (2006). Hill-climbing search. In *Encyclopedia of Cognitive Science*. John Wiley & Sons.

Xue, M. and Atkins, E. M. (2006). Terminal area trajectory optimization using simulated annealing. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada. AIAA.