

2007

## Near optimal solutions for the path planning problem of a small UAV in wind

Alan Lance Jennings  
*University of Dayton*

Follow this and additional works at: [https://ecommons.udayton.edu/graduate\\_theses](https://ecommons.udayton.edu/graduate_theses)

---

### Recommended Citation

Jennings, Alan Lance, "Near optimal solutions for the path planning problem of a small UAV in wind" (2007). *Graduate Theses and Dissertations*. 3535.  
[https://ecommons.udayton.edu/graduate\\_theses/3535](https://ecommons.udayton.edu/graduate_theses/3535)

This Thesis is brought to you for free and open access by the Theses and Dissertations at eCommons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of eCommons. For more information, please contact [mschlangen1@udayton.edu](mailto:mschlangen1@udayton.edu), [ecommons@udayton.edu](mailto:ecommons@udayton.edu).

NEAR OPTIMAL SOLUTIONS FOR THE PATH PLANNING  
PROBLEM OF A SMALL UAV IN WIND

A Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree

Master of Science in Electrical and Computer Engineering

by

Alan Lance Jennings

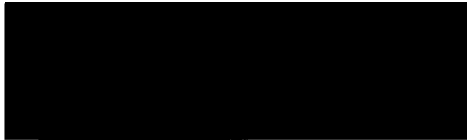
UNIVERSITY OF DAYTON

Dayton, Ohio

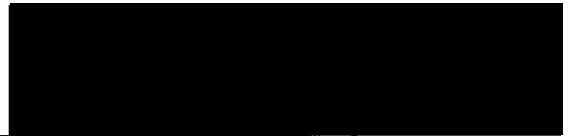
December 2007

NEAR OPTIMAL SOLUTIONS FOR THE PATH PLANNING PROBLEM OF A  
SMALL UAV IN WIND

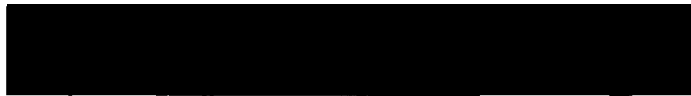
APPROVED BY:



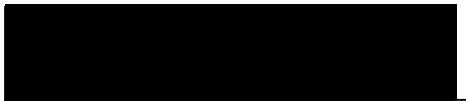
Raúl Ordóñez, Ph.D.  
Advisor Committee Chairman  
Associate Professor, Electrical and  
Computer Engineering



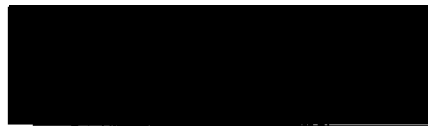
Russell Hardie, Ph.D.  
Committee Member  
Professor, Electrical and Computer  
Engineering



Corey Schumacher, Ph.D.  
Committee Member  
Air Vehicles Directorate, Air Force  
Research Laboratory



Malcolm W. Daniels, Ph.D.  
Associate Dean  
School of Engineering



Joseph E. Saliba, Ph.D., P.E.  
Dean  
School of Engineering

## ABSTRACT

### NEAR OPTIMAL SOLUTIONS FOR THE PATH PLANNING PROBLEM OF A SMALL UAV IN WIND

Name: Jennings, Alan Lance  
University of Dayton

Advisor: Dr. Raúl Ordóñez

This thesis presents a waypoint path optimization in travel time for an asymmetric environment, specifically a miniature air vehicle (MAV) in strong wind. Due to the wind, turning settling lengths depend on initial and final orientation and speed changes with heading. Discontinuities in path time for segment headings make optimization difficult. Four methods are presented for finding near optimal paths quickly. The methods used include discretized global search, radial basis network, stochastic optimization and dynamic programming. The global search provides good results but is computationally intensive. Cost discontinuities impair the radial basis approximation and little computation simplification was gained. Stochastic optimization approached the global search results and simplified complexity. Dynamic programming made a database of the quickest turns between discrete orientations. Paths from the database were perturbed to desired orientations. This method gave good results very quickly. When dynamic programming results were poor, they exceeded a threshold suggesting other methods would do better.

To my parents, who raised me to think.

## **ACKNOWLEDGMENTS**

My sincere appreciation goes to DAGSI for their generous support and recruiting me to the Dayton Area.

I am very grateful for Raúl Ordóñez and Nicola Ceccarelli for introduction to this project and providing guidance. I also acknowledge the support of my peers in the Controls lab and in my studies, most notably Kayode Ajayi-Majebi, Sudarshan Srinivasan and Emmanuel Otoo.

## TABLE OF CONTENTS

	Page
Abstract . . . . .	iii
Dedication . . . . .	iv
Acknowledgments . . . . .	v
List of Figures . . . . .	viii
 CHAPTERS:	
I. INTRODUCTION . . . . .	1
II. PROBLEM STATEMENT . . . . .	5
2.1 Analysis . . . . .	5
2.2 Solution Range . . . . .	10
2.3 Cost Relation . . . . .	12
2.4 Settling Lengths . . . . .	13
III. METHODS . . . . .	19
3.1 Discretized Global Search . . . . .	22
3.2 Radial Basis Network . . . . .	32
3.3 Stochastic Optimization . . . . .	40
3.4 Dynamic Programming . . . . .	45
IV. CONCLUSION . . . . .	52
 Appendices:	
A. Proof of Feasible Solution for Long Turns . . . . .	59

B. Settling Behavior . . . . .	63
Bibliography . . . . .	70



## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1 Headings have a nearly sinusoidal relationship on cost while segment lengths scale the cost linearly. . . . .	13
2 Path followed by MAV for paths with various turns. The settling length is the distance from the turning point when the path error stays below a threshold. Wind is 4 knots towards North. . . . .	15
3 Example of settling length constraint on cost. Left plot shows all costs while the right plot only shows costs of valid paths. . . . .	17
4 Example of cost relating to two headings. . . . .	18
5 The first grid (a) is coarse and the best point is selected to center grid refinement. The second, finer grid (b) has a smaller domain. If the finer grid extended across the entire domain, the number of points to test would grow to be intractable. Staging the refinements offers a compromise of accuracy and thoroughness. . . . .	23
6 Plots comparing outputs and cost from fine and wide meshes for 23 trials. Ideally, all points would lie near the line. Points deviating from the line show the solution diverging due to the mesh used. . . . .	24
7 Paths with more segments or a finer grid took longer to process. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Computation times were consistent for a method. A mean of longer than 10 min to compute a path is impractical. . . . .	25

8	Paths with more segments or a finer grid produced lower costs. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Note that having three segments paths has high costs and an infinite mean since some paths had no solution. The best is 4W when disregarding 4F and 5W for excessive computation time. . . . .	26
9	A compromise between having low cost and low computation time is achieved by using the sum of computation time and path cost. Computation time is weighted, so a weight of 0.01 means computation time is less important. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Five segment paths with a wide grid gives overall good behavior. . . . .	28
10	Sequential targets to simulate MAV flight over DGS paths. The numbers show the target order while the vectors show the desired heading. . . . .	29
11	Simulated MAV flight over DGS paths using simulated settling lengths with different wind gust magnitudes. The straight lines represent the desired path while the curves represent different trials. . . . .	30
12	Simulated MAV flight over DGS paths using simulated settling lengths plus the minimum segment length with different wind gust magnitudes. The straight lines represent the desired path while the curves represent different trials. . . .	31
13	The top plot shows how well the function is approximated with a radial basis network while the bottom plot shows the individual neuron contributions. . . .	36
14	Neural network structure is a set of radial basis functions spaced across the domain of $x$ . Training data is used to choose the neuron weights $a$ . . . . .	37
15	Plots show the output ( $\theta_1$ , $\theta_2$ and $\theta_3$ ) vs the parameters of the test points. The network seeks to emulate the relation of these test points. Note that though some relation exists, it is not a function in any individual parameter. . . . .	38
16	Results of comparison set with radial basis function compared to DGS (cost change mean=infinity, median=235 sec). Some test points had no solution. The following methods have better results. . . . .	39

17	Example of generating probability distribution function. Test points in top left corner represent an unknown functional relationship. For a specific input value, an individual probability distribution function is built (PDF). Three different examples are shown on the right. A surface of distribution functions across all inputs showing PDF response to input changes is in the bottom left corner. . . .	42
18	Results of comparison set with stochastic optimization compared to DGS (cost change mean=87 seconds, median=67 seconds). Cost is higher than DGS for the stochastic method, but stays close for most cases. Computation time is a fourth to a fifth that of DGS. . . . .	44
19	Paths considered for optimal turn progression of 50° to the left from flying with the wind. . . . .	46
20	Viable paths are possible in the range reached by extending the segments. Direct turns (top) have a limited range while long turns (> 180°, bottom) are viable for the entire space. . . . .	48
21	Results of comparison set with dynamic programming compared to DGS (cost change mean=105 sec median=47 sec), area of interest enlarged in (b). Results are normally very good, but a few cases had very large costs. Dynamic programming allowed more turns. Using more turns allowed lower costs than DGS, especially when DGS had high costs (> 175 sec). Computation was very fast (about 0.15 sec). . . . .	51
22	This plot is a comparison of the results of the different methods sorted by ascending cost change from the dynamic programming method. Though the cost ascends quite high, the dynamic programming method often has the best cost. For the cases when the cost is over 125 seconds, the probabilistic method often has better results than dynamic programming. . . . .	55
23	Here the results are sorted by ascending cost change from the probabilistic method. No order is seen among the other methods. . . . .	56

24	Results of comparison set with hybrid method compared to DGS (threshold=175 sec, cost change mean=105 sec and median=47 sec). Results are consistently very good, and the few cases of very large costs with dynamic programming are now normal. Computation remained much faster the DGS (about 15 sec total computation time). . . . .	57
25	Results of comparison set with hybrid method compared to DGS (threshold=175 sec, cost change mean=105 sec and median=47 sec). Results are consistently very good, and the few cases of very large costs with dynamic programming are now normal. Computation remained much faster the DGS (about 15 sec total computation time). . . . .	58
26	This shows the region for $\theta_i$ that guarantees a valid solution. . . . .	61
27	Settling behavior for wind speed of 4 knots. . . . .	64
28	Settling behavior for wind speed of 12 knots. . . . .	65
29	Settling behavior for wind speed of 16 knots. . . . .	66
30	Settling length function for wind speed of 4 knots. . . . .	67
31	Settling length function for wind speed of 12 knots. . . . .	68
32	Settling length function for wind speed of 16 knots. . . . .	69

## **CHAPTER I**

### **INTRODUCTION**

This thesis describes the process of finding near optimal solutions for waypoint paths for miniature uninhabited air vehicles (MAVs). The primary goal is to provide a steady path through a point using an existing waypoint following controller. Unmanned air vehicles (UAVs) have been selected as the tool of choice for surveillance over dangerous terrain or extensive assignments. Objectives may be either military, entrepreneurial or academic and may include target surveillance, search and rescue, weather monitoring or news reporting. MAVs are generally regarded as UAVs under a meter in wingspan. They offer several advantages and challenges over UAV or other traditional methods. The limited size offers potential for reduced power consumption, cost and transportation burden. Research has been directed to achieve these potentials and overcome the challenges of sensitivity to wind, reduced payload capacity and power storage. More massive planes and UAVs possess inertia to resist wind disturbances and fly at faster speeds attenuating the effect of the wind. A pilot would apply experience in compensating for the wind, however the goal of MAV flight is for autonomous navigation.

A primary goal of MAV surveillance is to take an image of a target. Sometimes a particular view of the target is desired. Consider the case when the view is obstructed from a direction, then the camera must view the target from a limited range of angles. In order to capture an

image of the target, the plane must pass through a point so that the sensor footprint falls on the target. Due to weight issues, many MAVs use a camera fixed to the body of the plane. When the camera is gimbaled and free to be rotated in one or more angles, this constraint may not apply. This thesis deals with the fixed camera case, or for cases that require a specific view of a target.

Many papers have addressed the need for autonomous control. Two primary objectives are considered, though others exist. The first objective is to increase data gathering. These are output driven methods seeking to resolve uncertainty in searches by covering the largest area, eg. [1], or improve placement of the sensor footprint, eg. [2]. Methods of reducing controller error would also fall under this category with the added benefit of assisting collision avoidance by reducing path uncertainty. The other primary objective is minimizing the time of the path. This has been done most notably with Dubin's car paths for constrained turning rates [3]. Dijkstra's Algorithm, Prim's Algorithm and A\* offer path optimizations for discrete event, discrete valued systems ([4][5][6]). Another optimization objective is decreasing the strain on the MAV by path smoothing. In [7], other curves (clothoid and Pythagorean Hodograph) are examined for fitting to Dubin's paths while meeting turning and lateral acceleration constraints. In [8], the proximity to the waypoint for switching the controller to next waypoint segment is optimized to minimize acceleration. For discrete systems, paths are limited from having switchbacks which are effectively U-turns [9].

Several methods have been shown for UAV path following. With no wind disturbance, a linear quadratic regulator (LQR) is shown to be sufficient to produce asymptotically stable (AS) results in following a waypoint line [8]. In [10], a sliding surface controller shows AS

results in wind gusts following a polynomial-smoothed Dubin's path. An added condition is the Dubin's turn rate must be set smaller than the maximum turn rate to provide control allowance. Another approach is to overlay a vector field of desired headings and control the MAV to follow the vector field. Sets of vector fields are provided in [11] for following straight lines and circular arcs in an unknown, unsteady wind. These results show AS both theoretically and experimentally. To eliminate waypoint overshoot in uncertain wind, in [12] the distance to transition to the next waypoint segment is set via a look-up table from the results of a wind observer.

Once the low-level, path following control law has been set, a high-level, path generating controller can be used based on the behavior of the path following controller. Often this is done by discretizing a continuous path into a set of waypoints ([1][2][9][10]). The obvious reason is this tremendously simplifies the path planning and the waypoint path following controllers offer great performance. Continuous time paths are much easier to optimize since they allow for optimal control techniques. The effect of wind is addressed in [13]. It shows that the optimal path to a fixed point in wind can be transformed to the optimal path to a moving point in no wind. Traditional Dubin's paths may be nonoptimal and two non-minimum length paths may provide minimum time. This method is adapted to waypoint paths in [2] and [10].

The other approach would be to view the path as a continuous valued, discrete event system. Since the path can take continuous values, it does not lend a characterization for traditional traveling salesman problems. For continuous values, a optimization is shown for sailboat paths as a set of discrete course changes [14]. However, this method does not incorporate a direction varying minimum length constraint. Not only does the wind affect the speed which a MAV

flies, it also affects the response from the control law. In order for the MAV to stay near the path, the segment length must be sufficiently long for the transient to settle. A direction dependent segment length requirement is the novel constraint imposed for this thesis. By accounting for the wind effect on behavior, this research can plan paths that take advantage of controller-wind synergy.

It is true that the Dubin's paths can be seen as discrete event systems. Some path following controllers can follow waypoint paths with curves rather than lines ([1][11]). For these controllers, the continuous Dubin's path results should still provide good results. Still, it is possible that these results can be improved when accounting for settling length constraints.

The motivation for this research is driven by tour planning algorithms. Many algorithms plan tours as tree searches. Branches can be pruned from the search if they can be shown to be suboptimal. Because tour planning problems are often exponential in complexity, subroutines in the tour planning algorithm should be very quick to allow for thorough searches.



## CHAPTER II

### PROBLEM STATEMENT

#### 2.1 Analysis

The objective of this research is to determine the minimum time waypoint path for a MAV to fly in wind. This problem can be generalized to situations where dimensions are weighted unequally and constrained in a possible nonlinear relation. The objective is to describe a path from  $[X_0 \ Y_0 \ \theta_0]^T$  to  $[X_{n+1} \ Y_{n+1} \ \theta_{n+1}]^T$  in  $\mathcal{R}^2 \times \mathcal{C}$  space while meeting a set of constraints with  $\mathcal{R}^2$  representing the cartesian location and  $\mathcal{C}$  representing the orientation (eg., heading).

Constraints in this thesis originate from requirements of MAV's waypoint flight. Waypoints mark the end of a segment and the beginning of a new segment. Changes in commanded waypoint occur at a fixed point in space due to waypoint navigation. This can be understood as a heading change. Movement (in  $\mathcal{R}^2$ ) is assumed to follow the assigned heading ( $\mathcal{C}$ ). After a turn this condition occurs once the transient has settled. To justify this assumption, segments must be greater than a certain length determined by the original and final heading. For all points to be reachable, the air speed of the MAV must be greater than the wind speed.

To solve this optimization problem, variables of heading and segment length are chosen with constants for wind, air speed and number of segments. For fully constrained problems

(e.g., only one path is possible), the path is checked for satisfying the constraints and no optimization is performed. For problems with multiple paths between points, a particular solution is generated and the null space is used to vary the solution to the optimal while satisfying constraints.

The path is broken into a number of segments satisfying

$$\begin{aligned} \begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} &= \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \sum_{i=0}^n \begin{bmatrix} \cos(\theta_i) d_i \\ \sin(\theta_i) d_i \end{bmatrix}, \\ \begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} &= \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \begin{bmatrix} \cos(\theta_0) & \dots & \cos(\theta_n) \\ \sin(\theta_0) & \dots & \sin(\theta_n) \end{bmatrix} \begin{bmatrix} d_0 \\ \vdots \\ d_n \end{bmatrix}, \end{aligned} \quad (1)$$

where  $n + 1$  is the number of segments,  $\theta_i$  and  $d_i$  are the heading and length of the  $(i + 1)^{\text{th}}$  segment. The path is considered invariant in  $R^2$  and can be reduced to a displacement ( $D$ ) in a direction ( $\theta_d$ ),

$$\begin{bmatrix} \cos(\theta_d) \\ \sin(\theta_d) \end{bmatrix} D = \begin{bmatrix} \cos(\theta_0) & \dots & \cos(\theta_n) \\ \sin(\theta_0) & \dots & \sin(\theta_n) \end{bmatrix} \begin{bmatrix} d_0 \\ \vdots \\ d_n \end{bmatrix}. \quad (2)$$

The constraint of segment length is given of the form

$$d_i \geq g(\theta_i, \theta_{i-1}), \quad \forall i \in \{1, \dots, n\}, \quad (3)$$

where  $g(\cdot, \cdot)$  is a function that maps  $\mathcal{C}^2 \rightarrow \mathcal{R}^+$  determining the minimum segment length. Though other parameters may affect the minimum length (e.g., wind speed and direction, payload, temperature, etc.), they are assumed to remain constant over the flight and optimization. Travel time is minimized through a scaled cost function. The travel speed is the magnitude of the vector addition of wind velocity and relative velocity (air velocity). Assuming

$V_a > V_w \sin(\theta_w)$ , the travel time for one segment is

$$\begin{aligned} t_i &= \frac{d_i}{V_w \cos(\theta_w - \theta_i) + \sqrt{V_a^2 - (V_w \sin(\theta_w - \theta_i))^2}} \\ &= \frac{d_i}{V_w \left( \cos(\theta_w - \theta_i) + \sqrt{V_r^2 - \sin^2(\theta_w - \theta_i)} \right)} \end{aligned} \quad (4)$$

where  $V_a$  is the MAV's airspeed,  $V_w$  and  $\theta_w$  are the speed and direction of the wind and  $V_r = V_a/V_w$ . Cost can be scaled by the wind speed and from now on only  $V_r$  will be used. The segment speed is bounded to the neighborhood of  $V_w(V_r \pm 1)$  and so  $V_r$  should be chosen greater than one to ensure finite, positive segment times. The total scaled cost is the sum of the individual costs,

$$\begin{aligned} C &= V_w \sum_{i=0}^n t_i \\ C &= \begin{bmatrix} \frac{1}{\cos(\theta_w - \theta_0) + \sqrt{V_r^2 - \sin^2(\theta_w - \theta_0)}} & \cdots & \frac{1}{\cos(\theta_w - \theta_n) + \sqrt{V_r^2 - \sin^2(\theta_w - \theta_n)}} \end{bmatrix} \begin{bmatrix} d_0 \\ \vdots \\ d_n \end{bmatrix}. \end{aligned} \quad (5)$$

The primary constraint is given by (2) and requires that the path actually reach the destination. For  $n = 1$ , (2) can be solved for segment lengths using matrix inversion, assuming unique angles. For the case  $\theta_0 = \theta_1$ , the matrix is degenerate and no solution will exist unless  $\theta_0 = \theta_d$ . For  $n \geq 2$  with unique angles, the problem is guaranteed to have a null space since the dimension of variables exceeds the number of constraints. This null space allows the possibility of multiple solutions and optimization. First the particular solution is created by setting  $d_i = 0, \forall i \in \{1, \dots, n-1\}$ . This reduces the number of variables and can easily be solved

for the case when  $\theta_0 \neq \theta_1$ , giving

$$\begin{aligned} \begin{bmatrix} \cos(\theta_d) \\ \sin(\theta_d) \end{bmatrix} D &= \begin{bmatrix} \cos(\theta_0) & \dots & \cos(\theta_n) \\ \sin(\theta_0) & \dots & \sin(\theta_n) \end{bmatrix} \begin{bmatrix} d_0 \\ 0 \\ \vdots \\ 0 \\ d_n \end{bmatrix} \\ \begin{bmatrix} \cos(\theta_d) \\ \sin(\theta_d) \end{bmatrix} D &= \begin{bmatrix} \cos(\theta_0) & \cos(\theta_n) \\ \sin(\theta_0) & \sin(\theta_n) \end{bmatrix} \begin{bmatrix} d_0 \\ d_n \end{bmatrix} \\ \begin{bmatrix} d_0 \\ d_n \end{bmatrix} &= \frac{D}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_n - \theta_d) \\ \sin(\theta_d - \theta_0) \end{bmatrix}. \end{aligned} \quad (6)$$

This choice is convenient since all the variables are known. The null space can be generated by setting  $d_i = 0$  and  $d_m = 1$ ,  $\forall i \in \{1, \dots, m-1, m+1, \dots, n-1\}$ ,  $m \in \{1, \dots, n-1\}$  where  $m$  is the index of the null space. This choice will guarantee that all null spaces are orthogonal to each other for unique  $\theta_m$  and contain one unknown variable,  $\theta_m$ . The solution is similar to (6),

$$\begin{bmatrix} d_0 \\ d_n \end{bmatrix} = \frac{-1}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_n - \theta_m) \\ \sin(\theta_m - \theta_0) \end{bmatrix}. \quad (7)$$

The set of solutions is the particular solution with linear combinations of the null space. Note that the null space magnitudes are equal to the intermediate segment lengths ( $\beta_i = d_i$ ,  $\forall i \in \{1, \dots, n-1\}$ ). The use of a different symbol is to show that these lengths are free to be varied

in the optimization, whereas  $d_0$  and  $d_n$  are functions of  $\beta_i$ . The solution of segment lengths is,

$$\begin{aligned}
 \begin{bmatrix} d_0 \\ \vdots \\ d_n \end{bmatrix} &= \begin{bmatrix} \left( D \sin(\theta_n - \theta_d) - \sum_{i=1}^{n-1} (\beta_i \sin(\theta_n - \theta_i)) \right) / \sin(\theta_n - \theta_0) \\ \beta_1 \\ \vdots \\ \beta_{n-1} \\ \left( D \sin(\theta_d - \theta_0) - \sum_{i=1}^{n-1} (\beta_i \sin(\theta_i - \theta_0)) \right) / \sin(\theta_n - \theta_0) \end{bmatrix} \\
 &= \frac{1}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_n - \theta_d) & -\sin(\theta_n - \theta_1) & \dots & -\sin(\theta_n - \theta_{n-1}) \\ 0 & \sin(\theta_n - \theta_0) & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sin(\theta_n - \theta_0) \\ \sin(\theta_d - \theta_0) & -\sin(\theta_1 - \theta_0) & \dots & -\sin(\theta_{n-1} - \theta_0) \end{bmatrix} \begin{bmatrix} D \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix} \quad (8)
 \end{aligned}$$

Now,  $d_i$  can be replaced in the cost function (5) to obtain

$$\begin{aligned}
 C &= \csc(\theta_n - \theta_0) \left[ \frac{1}{\cos(\theta_0) + \sqrt{V_r^2 - \sin^2(\theta_0)}} \quad \dots \quad \frac{1}{\cos(\theta_n) + \sqrt{V_r^2 - \sin^2(\theta_n)}} \right] \\
 &\quad \begin{bmatrix} \sin(\theta_n - \theta_d) & -\sin(\theta_n - \theta_1) & \dots & -\sin(\theta_n - \theta_{n-1}) \\ 0 & \sin(\theta_n - \theta_0) & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sin(\theta_n - \theta_0) \\ \sin(\theta_d - \theta_0) & -\sin(\theta_1 - \theta_0) & \dots & -\sin(\theta_{n-1} - \theta_0) \end{bmatrix} \begin{bmatrix} D \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix} \quad (9)
 \end{aligned}$$

Some optimizations methods can use the gradient to reduce computation time and improve results. The gradient of  $C$  can be taken with respect to the remaining variables,  $\Theta = [\theta_1 \dots \theta_{n-1}]^\top$  and  $B = [\beta_1 \dots \beta_{n-1}]^\top$

$$\begin{aligned}
 \nabla_B C &= \csc(\theta_n - \theta_0) \left[ \frac{1}{\cos(\theta_0) + \sqrt{V_r^2 - \sin^2(\theta_0)}} \quad \dots \quad \frac{1}{\cos(\theta_n) + \sqrt{V_r^2 - \sin^2(\theta_n)}} \right] \\
 &\quad \begin{bmatrix} -\sin(\theta_n - \theta_1) & \dots & -\sin(\theta_n - \theta_{n-1}) \\ \sin(\theta_n - \theta_0) & 0 & \dots & 0 \\ 0 & \sin(\theta_n - \theta_0) & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sin(\theta_n - \theta_0) \\ -\sin(\theta_1 - \theta_0) & \dots & -\sin(\theta_{n-1} - \theta_0) \end{bmatrix}, \quad (10)
 \end{aligned}$$

$$\nabla_{\Theta} C = \csc(\theta_n - \theta_0) \left[ \begin{array}{c} \frac{\beta_1 \cos(\theta_n - \theta_1)}{\cos(\theta_0) + \sqrt{V_r^2 - \sin^2(\theta_0)}} + \frac{-\beta_1 \cos(\theta_1 - \theta_0)}{\cos(\theta_n) + \sqrt{V_r^2 - \sin^2(\theta_n)}} + \\ \frac{\beta_1 \sin(\theta_1) \sin(\theta_n - \theta_0)}{\sqrt{V_r^2 - \sin^2(\theta_1)} (\cos(\theta_1) + \sqrt{V_r^2 - \sin^2(\theta_1)})} \\ \vdots \\ \frac{\beta_{n-1} \cos(\theta_n - \theta_{n-1})}{\cos(\theta_0) + \sqrt{V_r^2 - \sin^2(\theta_0)}} + \frac{-\beta_{n-1} \cos(\theta_{n-1} - \theta_0)}{\cos(\theta_n) + \sqrt{V_r^2 - \sin^2(\theta_n)}} + \\ \frac{\beta_{n-1} \sin(\theta_{n-1}) \sin(\theta_n - \theta_0)}{\sqrt{V_r^2 - \sin^2(\theta_{n-1})} (\cos(\theta_{n-1}) + \sqrt{V_r^2 - \sin^2(\theta_{n-1})})} \end{array} \right]^T \quad (11)$$

These gradients can be used for standard optimization techniques.

However, there is the additional constraint on segment lengths to ensure settling to the path, given by (3). The nature of settling depends on many factors in the MAV, controller and wind. Due to system complexity, simulations of the MAV with the waypoint following control law under various conditions were used to determine reliable settling lengths. This data can either be used as a look up chart using interpolation between points or can have a surrogate function that approximates the data. Uncertainty in wind can be accounted for using the maximum settling lengths over the variation. The disadvantage is that the larger the uncertainty, the further the path will be from the actual optimal path.

## 2.2 Solution Range

To obtain a comprehension of the nature of the problem the range space should be determined. Solutions may be visualized in three dimensions (two dimension space with a third dimension being the heading). On each level plane, the path may only travel in its MAV heading (a straight, directed line). The path travels vertically through headings at turns. The new path would twist as the heading changes with vertical travel similar to how a wrench changes its direction as it travels vertically with a bolt.

The simplest solution would be a turn exactly at the initial point ( $n = 1, d_0 = 0$ ). In order for this path to be a valid solution, the goal heading must be the same as the displacement heading and the displacement distance must be greater than the associated settling length. Visually, the range space of the simplest solution would then be a helical spiral with the center cut out in accord with the settling length function. Any point that lies on the range space has a feasible path.

The next order solution would be  $n = 1$ , or no intermediate section though the initial and final segments may have any length. Here the spiral moves forward (in the direction of the initial heading) through space. So rather than having a solution surface, a solution solid is formed. Anything that lies behind the spiral (opposite the initial heading) contains no valid solution. Valid paths have final headings facing away from the initial heading, have an intersection on the positive side of the initial heading and are far enough from the initial heading path to settle. Any other target would require a negative length path or not have space to settle.

Having one intermediate section,  $n = 2$ , the range space is the composite of two of the helical solids where a new helix starts at every point of the  $n = 1$  helix. Between the two helices, the world is covered. Previously, solutions were limited because only one point would be on both the initial and final path. Now with an independent intermediate segment, the intersection of the intermediate segment with the final segment may be set arbitrarily far from the target to ensure settling. It follows, allowing  $n \geq 2$  guarantees a solution. This is shown along with sufficient requirements in Appendix A.

This range space holds as long as all variables are allowed to change and  $g(\cdot, \cdot)$  is finite. When some of the variables are fixed, a solution is not guaranteed. One obvious example is fixing the intermediate heading to the initial ( $\theta_1 = \theta_0$ ) and the  $n = 2$  problem degenerates into a  $n = 1$  problem that may not have a solution. If a heading change has an infinite settling length, then the segment or segments that enable a valid solution may never settle. Requiring settling lengths to be finite should not diminish the practicality of this research. If settling lengths are kept finite but arbitrarily large, the associated turn would only be used if no other solution with a reasonable turn was possible. Instead of no possible path being found, a high-cost, valid path is found and can at least be used as an optimization starting point if unsatisfactory.

### 2.3 Cost Relation

The next step was to examine the cost behavior of solutions. From looking at (9), the relationship of  $\theta_i$  and  $\beta_i$  can be seen. Heading angles are found in two places: the speed weighting factor and how a segment lengthens or shortens the starting and finishing segments, as seen in the three terms in the gradient (11). The speed weighting factor (the last term) plays a dominant role for high wind speeds ( $V_r \rightarrow 1^+$ ) but a much smaller role for more moderate wind speeds. Normally the largest factor is how a specific heading affects the geometry of the path (the first and second terms). Since the equation is constrained to finish at the correct point, the intermediate angle acts similar to a crank shaft making the path longer or shorter. The segment length has a linear relationship to cost. The lengthening of an intermediate segment effectively scales the total path. Figure 1 shows a general cost vs a heading and a segment length. Only positive intermediate segment lengths are shown since settling lengths are required to be positive. For



higher order systems, this behavior is repeated for each additional intermediate segment. Because this relationship is easily distinguishable, the optimization can be broken into linear and nonlinear optimizations.

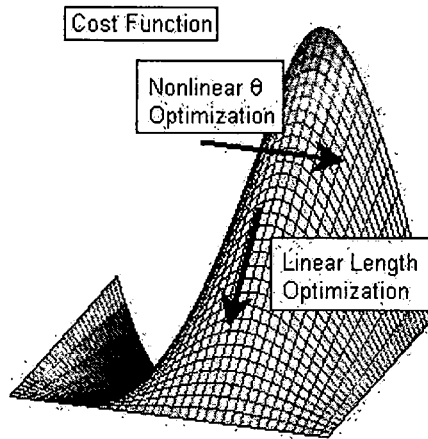


Figure 1: Headings have a nearly sinusoidal relationship on cost while segment lengths scale the cost linearly.

## 2.4 Settling Lengths

At this point more must be known of the settling length function,  $g(\cdot, \cdot)$ , from (3). The challenge is to describe the solutions with valid settling lengths. Many MAVs and other vehicles have a waypoint following guidance system. Each of these guidance systems are tuned to the vehicle for desired performance. Once this system is set, there is no direct control over aerial control. The purpose of this thesis is to optimize the path creation and not the waypoint follower. The behavior of the controller is considered deterministic for wind and heading changes, therefore simulations should give a practical length.

The path of a MAV was simulated going through turns at various wind speeds. An example is shown in Figure 2 with more examples in Appendix B Figures 27 to 29. The MAV would often overshoot the turn and often oscillate about the path segment. The settling length was the length from the turn point when the oscillations stayed below a threshold. The simulations were done with wind speeds from 0 to 18 knots in 2 knot increments. The MAV's airspeed was 24 knots and at wind speeds above 18 knots too many settling lengths were excessively long. Initial and final headings were tested in  $10^\circ$  increments. Symmetry with respect to the wind was exploited, so only  $180^\circ$  of initial headings needed to be simulated. Turns were done from  $120^\circ$  counterclockwise to  $120^\circ$  clockwise. Not all turns settled in the length simulated ( $\approx 4,000$  ft). Sometimes the settling length could be reasonably estimated, otherwise the settling length was set to a large value. For turns beyond the range tested, settling lengths were set to large values that linearly increased with turn magnitude. The simulation results were linearly interpolated in heading and wind speed to form a continuous settling length function. Examples of the settling length functions for a few wind speeds are also shown in Appendix B in Figures 30 to 32. This function satisfies the requirements of mapping  $\mathcal{C}^2 \rightarrow \mathcal{R}^+$  and is finite justifying the guarantee of a valid path. This function is not monotonic increasing and has long settling lengths for long turns or turning into the wind. The settling lengths abruptly change causing a great deal of difficulty in the optimization. Using these the simulated settling lengths for repeated DGS paths gave borderline results (see Figure 11). To improve the quality of the path, a minimum length (500 ft) was added to each settling length. Examples of repeated paths are shown in Figure 12.

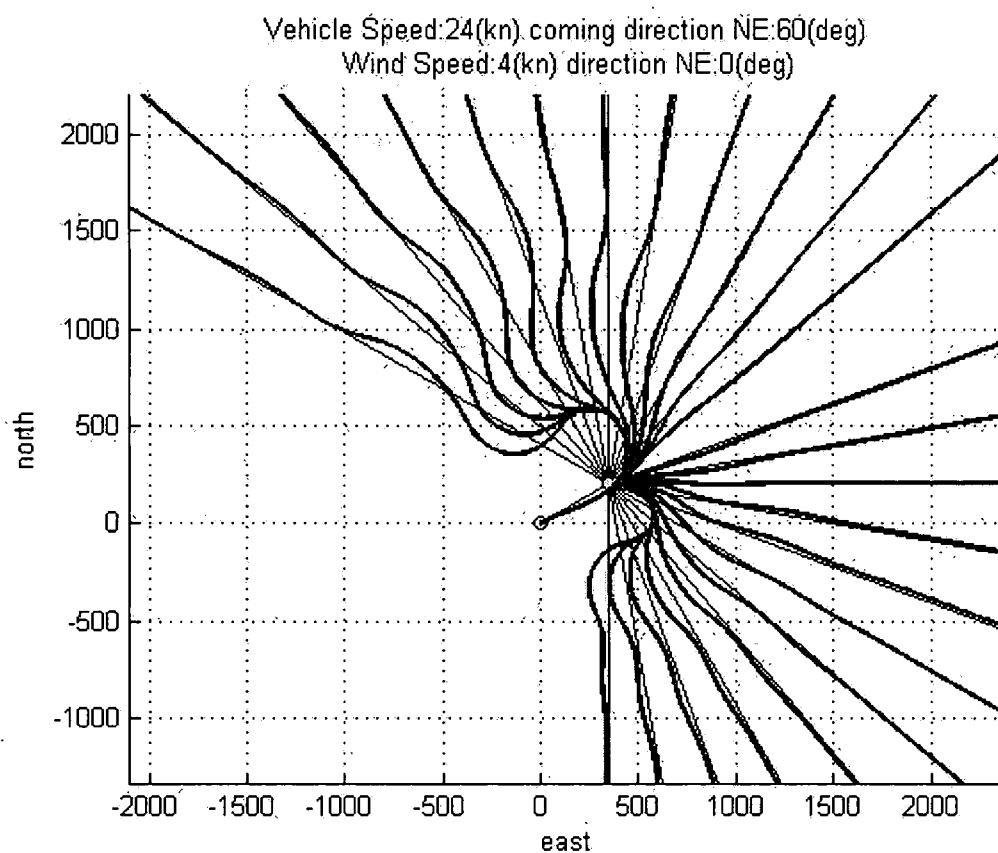


Figure 2: Path followed by MAV for paths with various turns. The settling length is the distance from the turning point when the path error stays below a threshold. Wind is 4 knots towards North.

The settling length function plays a significant role in complicating the optimization. Consider the case of  $n = 2$  with two free variables ( $\theta_1$  and  $\beta_1$ ). Figure 3 shows one representative example of the cost relation. The left plot shows all costs and could be easily optimized. The plot on the right excludes invalid paths and creates a jagged edge that generates most of the issues in optimization. The required settling lengths are determined by  $\theta_1$ . Once  $\theta_1$  is fixed, the settling lengths are fixed imposing a lower and/or an upper limit on  $\beta_1$ . A change in  $\theta$  across a linear boundary in  $g(\cdot, \cdot)$  may change the settling lengths significantly or in another direction. This effect can be seen from  $-180^\circ \leq \theta \leq 20^\circ$  where the feasible edge advances and retreats with an abrupt change at  $\theta = -60^\circ$ .

Note that one edge appears across the lower limit and another across the upper limit of  $\beta_1$ . The domain of  $\theta_1$  with no upper limit on  $\beta_1$  will always have a solution regardless of displacement distance. This conforms to the conclusions of the range space (Section 2.2); paths with one intermediate segment may reach any point and no upper bound on cost exists.

Looking at slightly more complex paths ( $n = 3$ ), four variables are involved:  $\theta_1, \theta_2, \beta_1$  and  $\beta_2$ . Since  $\beta_1$  and  $\beta_2$  both have an affine relation to cost with fixed domain for fixed  $\theta_1$  and  $\theta_2$ , they can easily be optimized using linear programming for a given choice of  $\theta_1$  and  $\theta_2$ . So the cost optimized in  $\beta_1$  and  $\beta_2$  can be shown across  $\theta_1$  and  $\theta_2$ . Figure 4 shows an example. However, no description can easily be made of the shape of the cost function. When there is a region with no linear programming solution, the cost in the area near it peaks sharply as if in a vertical asymptotic fashion. In other areas the cost has rolling regions with abrupt cliffs rather than a composite sinusoidal behavior. Rather than the one or two troughs, there are many local minima due to the behavior of  $g(\cdot, \cdot)$ . The cost does have a near symmetry for interchanged

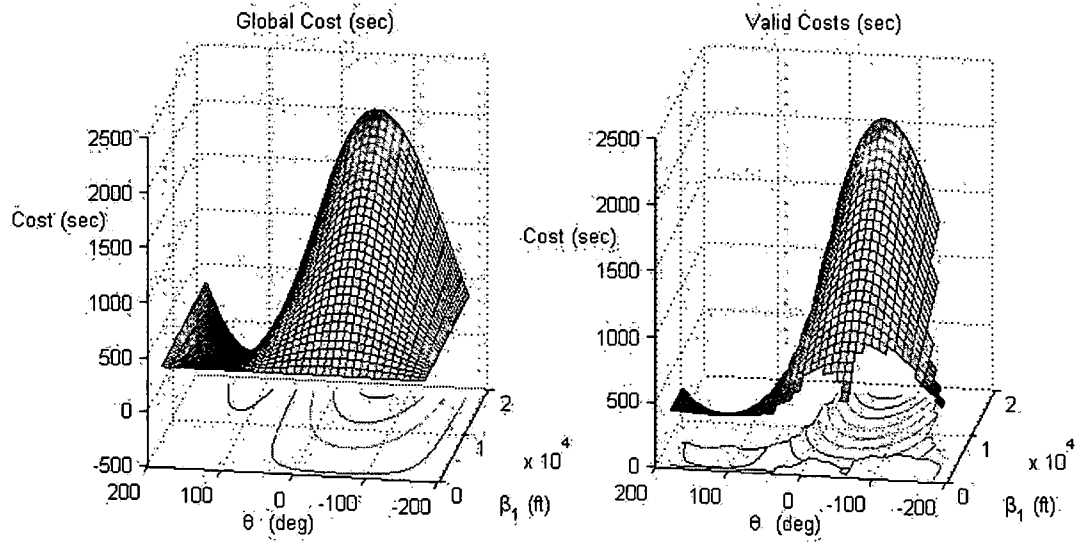


Figure 3: Example of settling length constraint on cost. Left plot shows all costs while the right plot only shows costs of valid paths.

$\theta_1$  and  $\theta_2$  because the paths are geometrically equivalent, but not exactly due to settling length variations. Numerical errors generated when optimizing the intermediate lengths affect the accuracy of the cost only slightly.

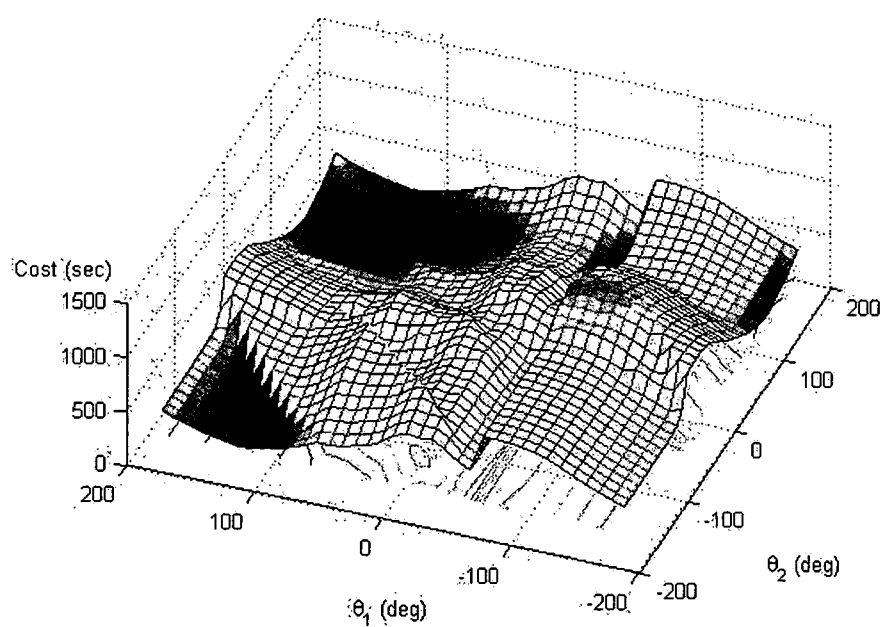


Figure 4: Example of cost relating to two headings.

## CHAPTER III

### METHODS

Four methods for finding low cost paths have been created and are compared here. The difficulty lies in finding the intermediate headings since segment lengths are easily found via linear programming. Each method outputs a set of intermediate headings  $(\theta_1, \dots, \theta_{n-1})$  that are then used to find the segment lengths  $(d_1, \dots, d_{n-1})$  and cost ( $C$ ). The methods act on test points which comprise an unique request (starting, finishing and displacement heading, displacement distance and wind speed). All headings are given with respect to the wind. Each method is briefly introduced here and then examined in a following section.

Since computation time is a concern, a general guideline of computation time is given. This is not to be interpreted as a best time but rather a means to compare the computational burden across methods. For implementation, some elements of the code could be simplified or a faster processor could be used improving the computation time. The computation times varied significantly enough for reasonable comparisons. Though the global search method, radial basis network and stochastic optimization all lend themselves to some parallel processing, sequential processing was done for all methods. The simulations were run in MATLAB<sup>®</sup> 2007 on a 2.20 GHz Intel<sup>®</sup> Pentium<sup>®</sup> 4 processor.

The standard of comparison is the discretized global search method (DGS). In this method, the range of intermediate angles is sectioned by a discrete grid with every point being tested. The point with the lowest cost being selected as the best path. This method is iteratively applied, normally three times, to reduce discretization error by narrowing the range and refining the grid. DGS is chosen as the standard since it reliably provides the best results. Training points for other methods are created using DGS. The disadvantage is the complexity grows by the power of the output dimension and searches may take a long time (about four minutes for a five segment path).

To estimate the best path faster, function estimation methods were used. The set of optimal intermediate angles can be seen as a function of the test point. A set of training data of test points with their optimal angles was used as instantiation of that function. Using the training data, the underlying function was estimated to find the set of optimal angles for a new test point. The first was done through a neural network approach using a weighted sum of radial basis functions centered across a grid of the domain. The closer a test point is to the center of the basis function, the higher the contribution of that weight. The weights are determined by fitting the sum to the training data. Given a test point, the function will give an estimation of the optimal location. Since this is only an estimation, that exact location may not give good results and should be coupled with another technique such as a local search. Another disadvantage is its complexity scales by the power of the input dimension. This method alone takes less than a second; when coupled to a nonlinear optimization, it takes one additional minute.

The other function estimation method follows a similar concept but creates a stochastic optimization. This randomness allows for multiple paths to be tested while retaining freedom for



coupling with another method. The two main advantages in this method are linear complexity and robustness by having multiple outcomes. Disadvantages of this method is that results are random and any individual results may be bad though most paths may be good. Parameters of the method may be varied so bad results can be made unlikely. It takes about 45 seconds for good results.

The final method described involves using dynamic programming. A database is compiled of turn progressions resulting in the minimum flight time for heading changes. The turn progressions are done for all points on the grid defining the behavior of  $g(\cdot, \cdot)$ . Dynamic programming is used to reduce the number of turn progressions evaluated.

This method exploits the nature of how lengths are constrained. Since the settling length constraints are linearly interpolated from a discrete grid, the settling lengths are piecewise linear. By not having the displacement constraint, (2), optimal paths will occur at linear boundaries and therefore must occur on the discrete grid. The search can be further trimmed by only considering combinations that involve optimal solutions. For a test point, the turn progressions for the neighboring initial and final heading and wind speeds will be considered with initial and final headings adjusted to the desired orientation. The great advantage of this method is that the number of segments does not need to be known a priori, as opposed to the other methods. Dynamic programming often found good results very quickly (less than a second). When dynamic programming results were poor, they exceeded a threshold suggesting other methods would do better.

Each method after the global search is compared to an identical set of 400 randomly generated test points. For the methods that involve training, they are trained using an identical set of

1,100 randomly generated training points with associated intermediate headings given by DGS. Initial, final and displacement headings span the domain with uniform randomness. Likewise the wind speed uniformly spans its domain of 2-18 knots. Displacement distances have an augmented normal distribution skewed to lower values. The mean and standard deviation of distances are roughly 600 and 325 feet respectively. This distance corresponds to close points with respect to the settling length function. For a specific application, the randomness of the training set and validation set should match the randomness of test points for the application. Since wind speeds are varied, the cost referred to is the flight time and not the scaled cost. Costs of the comparison set are shown in each section compared to the global search method (see Figures 16, 18 and 21). The change in cost is the most important comparison, so tests are ordered in ascending cost changes. For perspective on the magnitude of the flight time, the absolute cost of the global search method and the examined method is shown above.

### **3.1 Discretized Global Search**

The discretized global search (DGS) consists of testing all points on a discrete grid across all angles emulating a search across the entire domain. As the grid is refined, this discrete search approaches a true global search. However, as the grid is refined the number of points to test rises to the power of the search dimension. In order to compromise between good results and short computation, a coarse grid is used first and then a subsection of the domain is refined as shown in Figure 5. This refinement can be done any number of times; for this project two refinements were done. The desired effect is a cursory global search to identify a good region and then a local search to refine the first approximation. Identical test points were processed with a comparatively coarse or fine grid. The desired result would be the coarse mesh

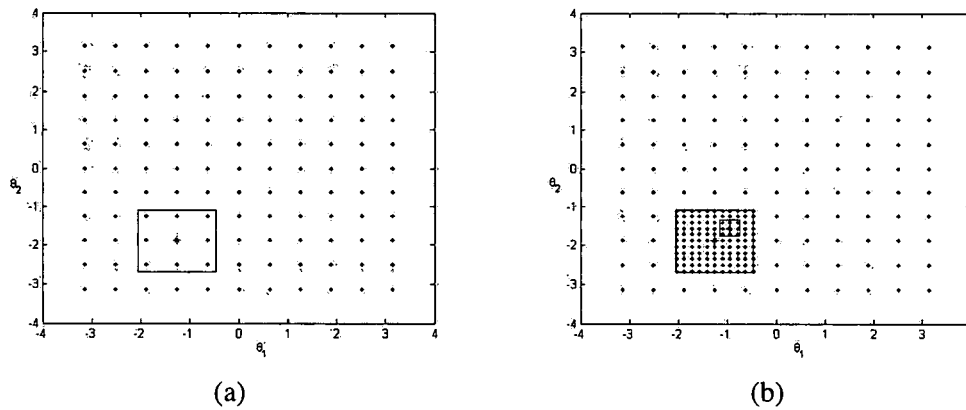


Figure 5: The first grid (a) is coarse and the best point is selected to center grid refinement. The second, finer grid (b) has a smaller domain. If the finer grid extended across the entire domain, the number of points to test would grow to be intractable. Staging the refinements offers a compromise of accuracy and thoroughness.

having the solution of a previous iteration of the fine mesh. Normally it did but sometimes the two methods would produce two substantially different paths. The fine mesh had consistently slightly better results (see Figure 6).

As expected the larger the search, the longer the search took and the results were generally better. Different parameters (number of segments and grid spacing) were used for random test points to ascertain what size of search was feasible and gave good results. Searches were done with a fine grid (about thirty points across each dimension) for three, four and five segment paths (2F, 3F and 4F respectively) and a wide grid (about 13 points across each dimension) for four, five, and six segment paths (3W, 4W, and 5W respectively). The number of random test points processed depended on the parameters and is represented by P. Computation time is shown in Figure 7. Computation time was very consistent for a parameter set so the median

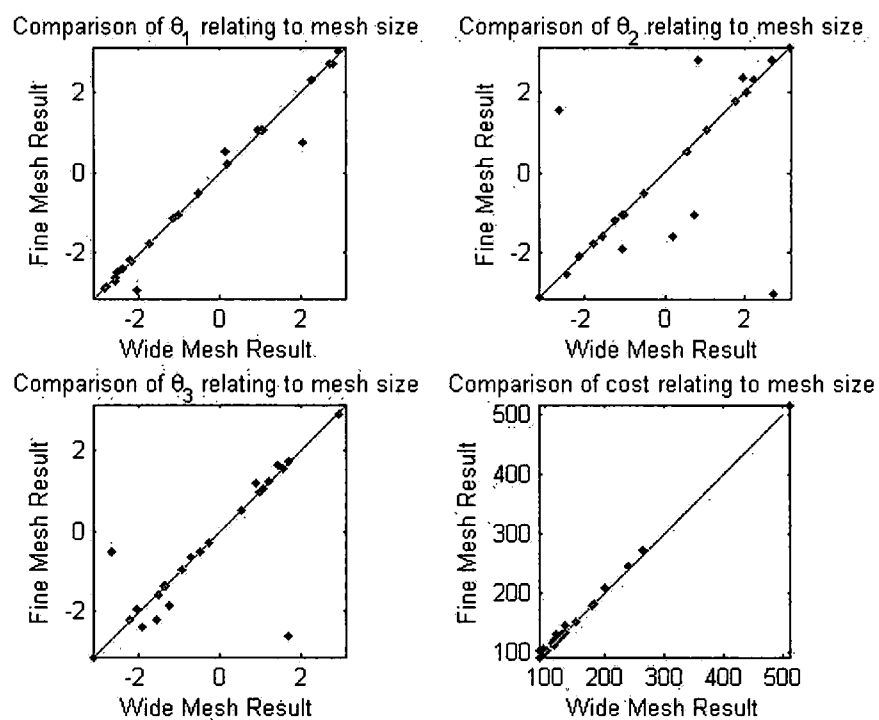


Figure 6: Plots comparing outputs and cost from fine and wide meshes for 23 trials. Ideally, all points would lie near the line. Points deviating from the line show the solution diverging due to the mesh used.

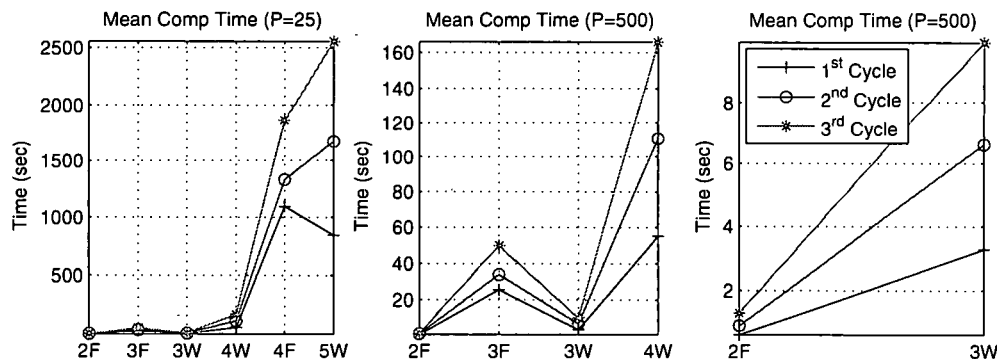


Figure 7: Paths with more segments or a finer grid took longer to process. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Computation times were consistent for a method. A mean of longer than 10 min to compute a path is impractical.

is not shown since it is now skewed from the mean. Taking longer than 10 minutes for a test point was decided to be too long.

Average path costs are shown in Figure 8. The path cost had a long left tail and the effect can be seen as the mean is skewed higher than the median. The extreme occurs with 2F where no valid path is found resulting in a infinite cost and infinite mean (upper left corner). This can occur when the grid points tested do not lie in the region guaranteeing a valid solution. Even among the paths that are created, 2F has high cost shown by the comparison of median costs (upper right corner). The cost is very close among 4F, 4W and 5W differing only a matter of seconds (upper middle). However, 4F has inordinate computation time and need not be considered. The comparison of the remaining methods in mean and median is shown across the bottom. Note that while 3F approaches 4W in the median, it does noticeably worse in the

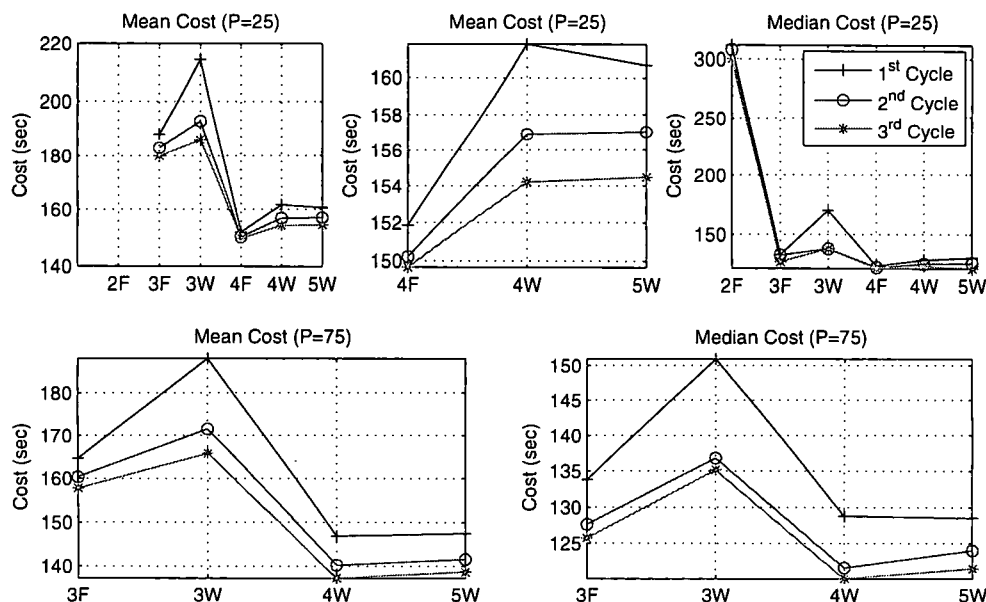


Figure 8: Paths with more segments or a finer grid produced lower costs. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Note that having three segments paths has high costs and an infinite mean since some paths had no solution. The best is 4W when disregarding 4F and 5W for excessive computation time.

mean evidencing a long tail. Though 5W also has inordinate computation time, it is included to show the diminishing gain of increasing the number of segments.

In order to balance the computational burden and path cost, the sum of the average path cost and a weighted average computation time is minimized. By using a weighted sum, the balance between path cost and computation time can be arbitrarily set. When the path cost is the primary desire, the weight can be set low. Results of a weight of 0.01 are shown in Figure 9 across the top. This weight implies that 100 seconds searching is worth a 1 second decrease in path cost. This weight would be appropriate for deciding on a method to generate training

data. If computation time is a driving constraint, a large weight would be used. Results of a weight of 10 are shown in Figure 9 across the top. This weight mean that a 10 second decrease in path cost is only worth 1 seconds searching. A high weight like this is appropriate for an real-time path generater. For training data and comparisons, 4W will be used because it gives good results in either case. For a actual implementation (with a weight of 10), 3W with one cycle would give the best results.

Paths created seem reasonable. A tour of random targets was created (see Figure 10). The numbers show the order of the targets and a vector points the desired heading. The results of the DGS were simulated for the actual MAV path. Since the wind speed remaining constant is unlikely, the path performance should be robust to wind gusts. Three simulations with wind gusts of varying magnitudes show the disturbed the flight performance. Originally, the exact simulated settling lengths were used, Figure 11. These results are of low quality. There may be a historical effect for repeated turns not observable by error from path. Since only single turns were simulated to determine the settling length, those results did not account for this effect. To improve the quality of the path, the minimum segment length was added to all settling lengths as described in 2.4. Using the updated settling length function, simulated path quality improves. Results are shown in Figure 12.

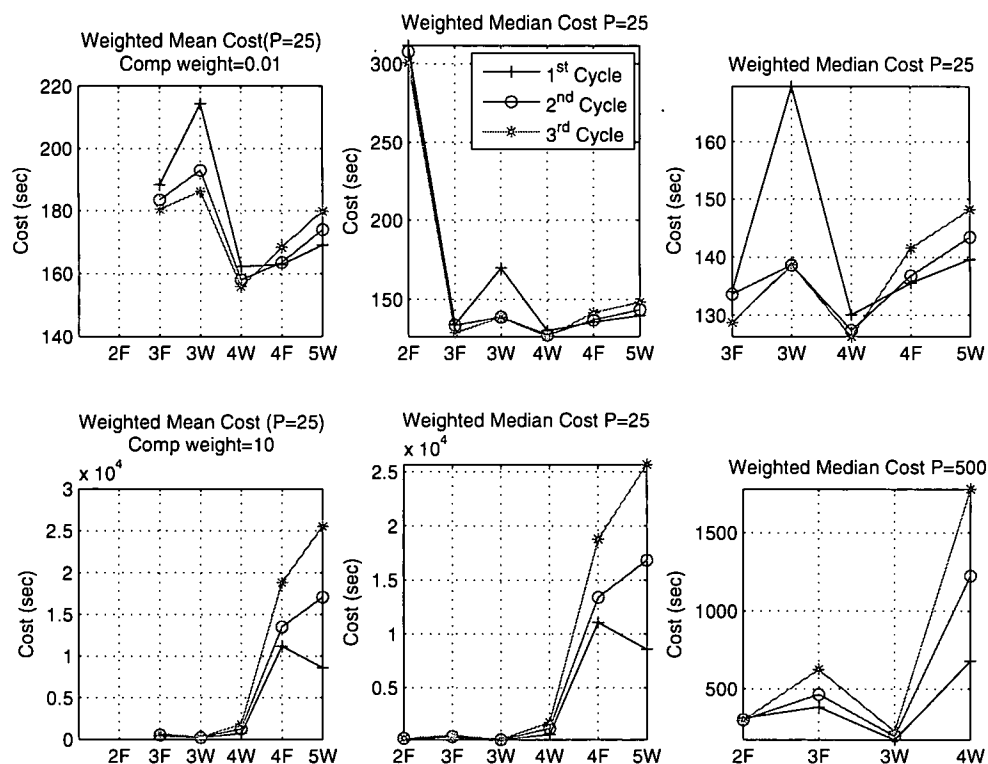


Figure 9: A compromise between having low cost and low computation time is achieved by using the sum of computation time and path cost. Computation time is weighted, so a weight of 0.01 means computation time is less important. The index, XY, represents searching for a path with X+1 segments by a fine (Y=F) or wide (Y=W) mesh. Five segment paths with a wide grid gives overall good behavior.



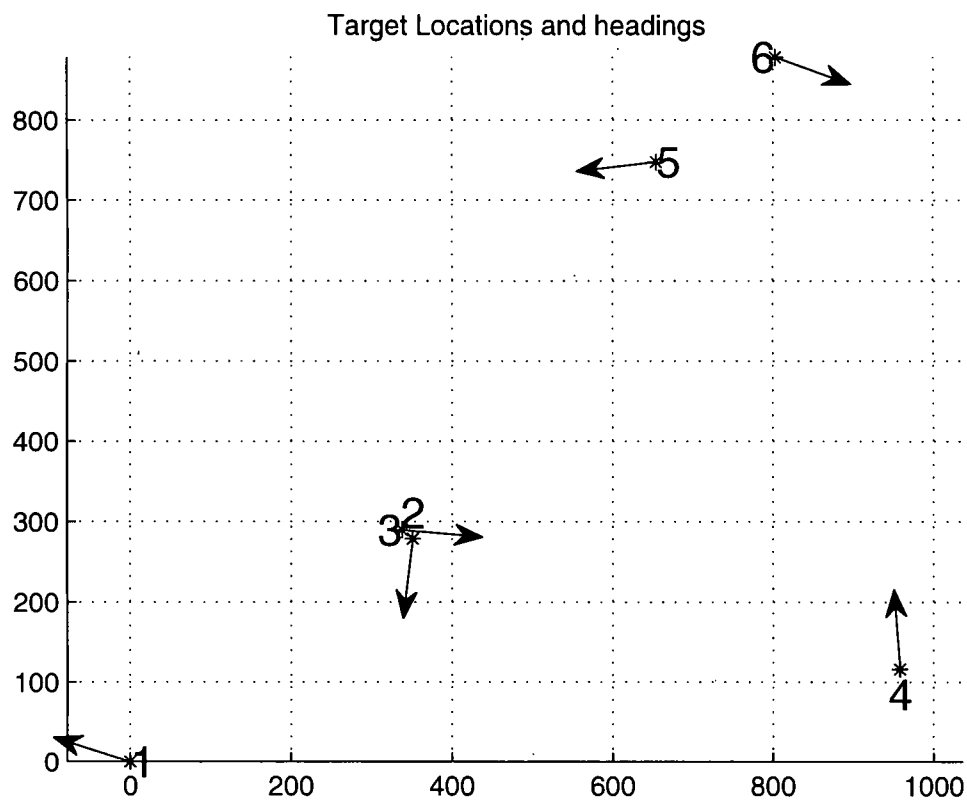


Figure 10: Sequential targets to simulate MAV flight over DGS paths. The numbers show the target order while the vectors show the desired heading.

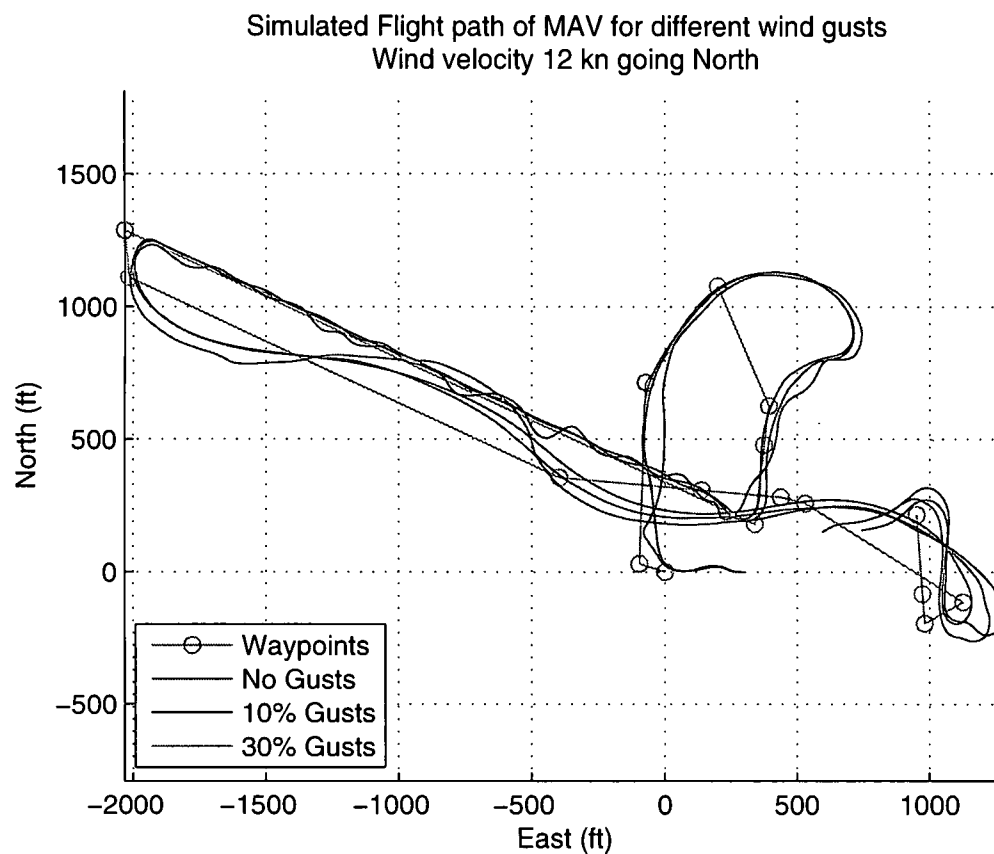


Figure 11: Simulated MAV flight over DGS paths using simulated settling lengths with different wind gust magnitudes. The straight lines represent the desired path while the curves represent different trials.

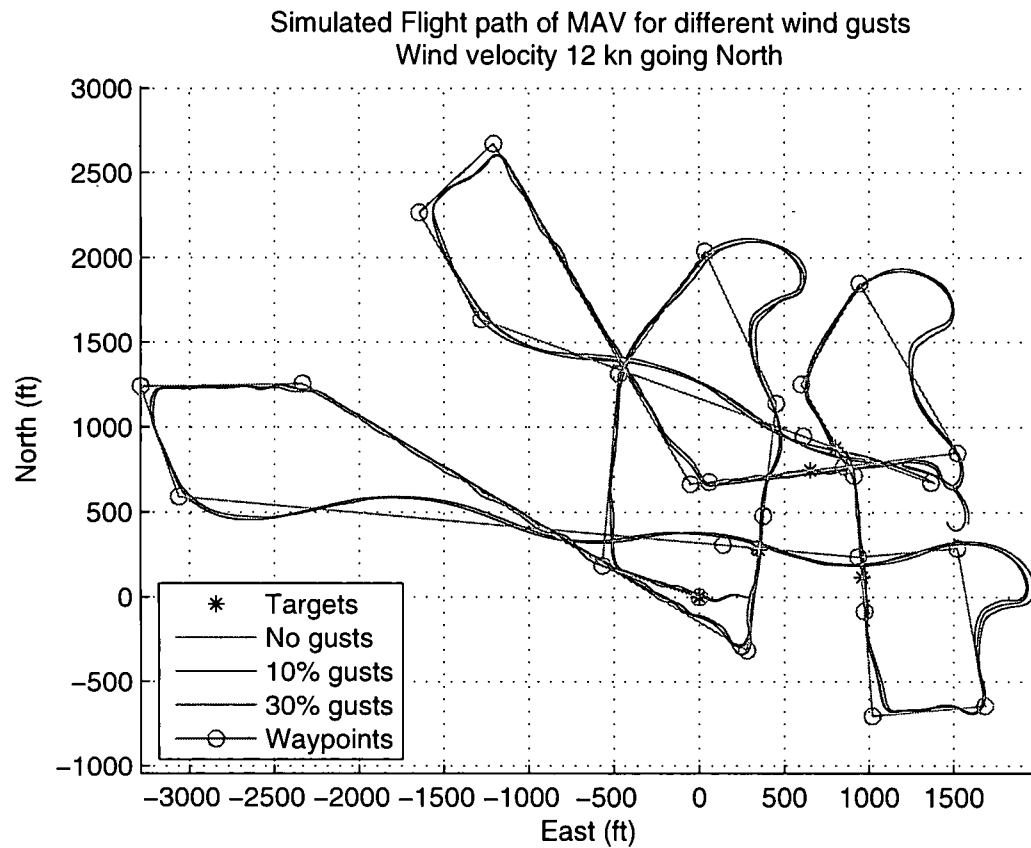


Figure 12: Simulated MAV flight over DGS paths using simulated settling lengths plus the minimum segment length with different wind gust magnitudes. The straight lines represent the desired path while the curves represent different trials.

### 3.2 Radial Basis Network

Neural Networks have been used to approximate functions in a variety of fields. Layered structures are often chosen for function approximation since their organization eases the training process. The input to the first layer neurons is a weighted sum of the test point. The neuron has some function used to process the input to create an output value. This first set of outputs may be used as inputs to the next neuron layer and so on, or to compose the final output.

Neuron functions, called activation functions, can be grouped by whether the activation function has only a local effect, a one-sided effect or a global effect. Using a linear function, the function would have a global effect, meaning a function change would significantly affect the entire spectrum. The hyperbolic tangent plus one has a local area of change with little change at extremes, but the weight applied to it will affect the right hand extreme making it a one-sided activation function. Radial basis curves have a local area of significance and diminish to insignificance at high and low values. Because they have only a local effect, they can be intuitively applied to function approximation. A set of radial basis functions with centers spread across the domain and normalized at each location, can be weighted to match the local output value and have little effect on the output at distant points. This is shown in Figure 13 where a sine function is approximated by weighting the sum of six neuron functions across the domain.

The structure used is shown in Figure 14. The input ( $x$ ) is the vector of parameters specifying the test point. The centers ( $c$ ) are vectors evenly spread across the domain. Test point dimensions are scaled by a matrix  $b$  to be of the same order of magnitude and to accentuate output sensitivity to an input dimension. For simplicity, all off diagonal elements of  $b$  are set

to zero and  $b$  is constant for each network. To train the network, the coefficients  $a$  are scaled to reduce error across the set of training data. The rest of the parameters remained constant through training.

The concept of this method is that as the network is trained to the data, it conforms to behavior underlying the input-output relation. The relation is given in Section 2.1, but is too complicated to use with the piecewise-linear settling length function. By training the network to the effect of these constraints in training data, the network could interpolate the behavior to new points. This is a reasonable assumption since some correlations can be noted by looking at graphs of components of the test point and the output, the intermediate headings. For the case of five segments ( $n = 4$ ), there are three intermediate headings:  $\theta_1, \theta_2$  and  $\theta_3$ . Plots of these variables against the dimensions of the test point in Figure 15 show all three having an important relation to  $\theta_n$  and minor relations to  $\theta_0$  and the other variables. Better relations were sought in higher dimension graphs, but could not be found. Training the network should uncover relations difficult to see. Preprocessing was done to aid network training. Since the outputs are module  $2\pi$  and the network imposes a discontinuity, the location of that discontinuity was chosen to be where few outputs would be. In addition, the outputs were detrended with a input to improve accuracy.

To generate overall good behavior, the error across the domain is minimized. In practice this is often done through training the network to a set of known points called the training set. However, reducing the error for this subset of the domain may cause large errors at other points in the domain; a phenomenon called over-training. To prevent over-training, the weights are trained to one set of data. The history of weights is used to find the least squared error to a

second set of data called the validation set. The set of weights that minimizes error over the validation set is chosen.

One network is created and trained for each output dimension. Training the complete network may take a long period of time (10-20 hours), but only needs to be done once and can be done autonomously. To test a specific point, the values at that location for every basis function are calculated and normalized. The weighted sum is the output. This process can be done very quickly (about five points/second). The disadvantage is that complexity scales to the power of the input dimension, and linearly in output dimension. This is opposite most other methods. For each input dimension, one dimension has to be added to the array of basis functions. The size of the basis array determines training time which can quickly prove intractable. For this method, seven basis functions were spaced across the five input dimensions resulting in 16,807 basis functions. At least twelve neurons in a dimension were desired to refine estimator behavior. That number of neurons was infeasible due to variable size limits in MATLAB.

The outputs were used directly in a linear optimization, but this produced very poor results. About half the paths had no solution. DGS likely found solutions against constraints. When these points were used to train, perturbations of the output had a good chance of falling outside of the valid area. As seen in Figure 6, DGS is not guaranteed to converge to the same minima. Consider two test points near each other but converged to minima in different regions. The network training seeks to put the output near both regions and ends up missing both. Even though there is a deterministic relation of what the optimal angles are for a test point, those

angles are not known. Only the output of DGS is known which may appear nondeterministic due to converging to different regions.

To get better results, the network outputs were used as starting points in a nonlinear staged optimization. For the staged optimization, a set of intermediate segment headings  $(\theta_1, \dots, \theta_{n-1})$  were used as the starting point of a nonlinear optimization of headings which had a linear suboptimization of segment lengths. The reason for staging the optimization is that the dimension of the nonlinear optimization is reduced and the effectiveness of linear programming can be used where applicable.

Radial basis network results used in a staged optimization are shown in Figure 16. The nonlinear optimization added about a minute of time to solve each path. A significant share of test points were invalid, but less than before and overall costs were much lower also. The cost change average was infinite for the mean and 235 second for the median (about twice the DGS cost). These results could be improved slightly by using other local search methods. Improvement would be limited due to the fundamental issue of having multiple regions of outputs close in the domain and not enough neurons to capture the behavior.

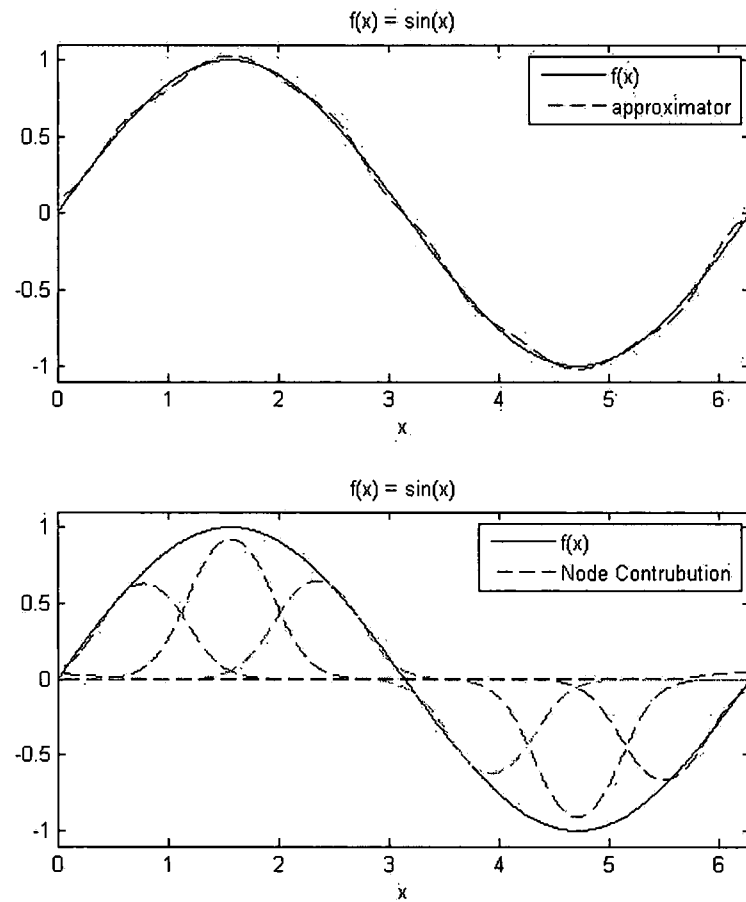


Figure 13: The top plot shows how well the function is approximated with a radial basis network while the bottom plot shows the individual neuron contributions.



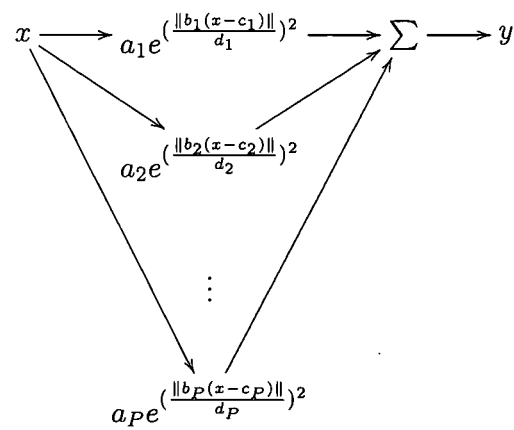


Figure 14: Neural network structure is a set of radial basis functions spaced across the domain of  $x$ . Training data is used to choose the neuron weights  $a$ .

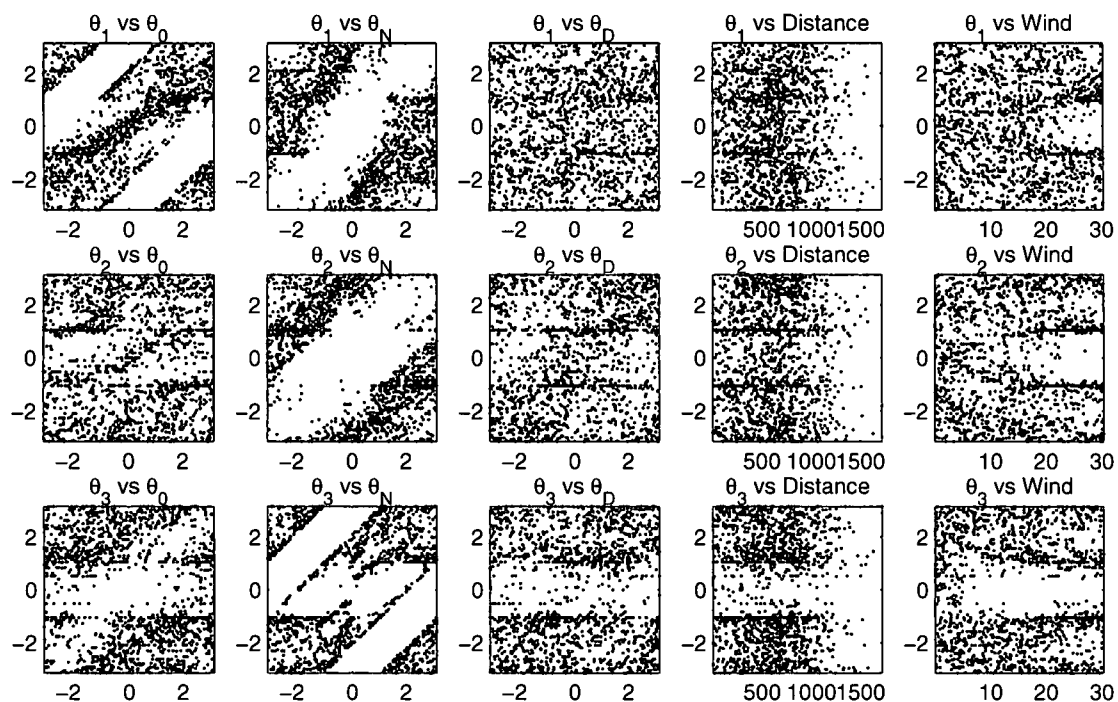


Figure 15: Plots show the output ( $\theta_1, \theta_2$  and  $\theta_3$ ) vs the parameters of the test points. The network seeks to emulate the relation of these test points. Note that though some relation exists, it is not a function in any individual parameter.

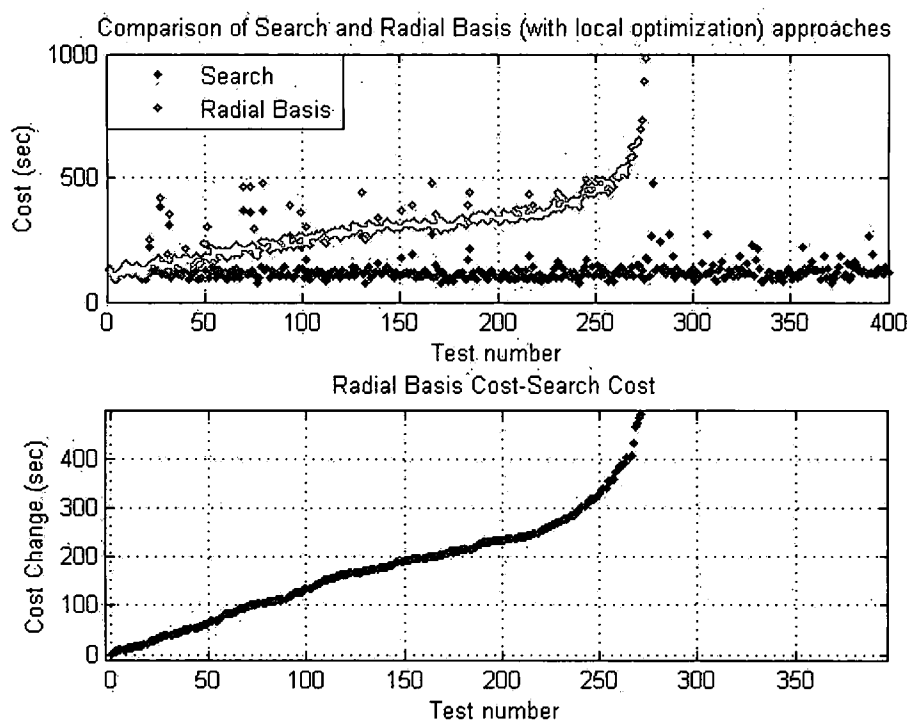


Figure 16: Results of comparison set with radial basis function compared to DGS (cost change mean=infinity, median=235 sec). Some test points had no solution. The following methods have better results.

### 3.3 Stochastic Optimization

Stochastic optimization involves either noisy inputs or algorithm decisions being made randomly. This application generates intermediate headings randomly through probability distribution functions of  $y$  created by the proximity of the test point,  $x$ , to  $x_p$  where  $x_p$  is one of the  $P$  points with known optimal intermediate angles  $y_p$ . The test point is a vector forming the parameters of the request: starting, finishing and displacement heading, displacement distance and wind speed. One probability distribution function is created for each intermediate segment,  $m$  where  $m \in \{1, \dots, n-1\}$ . The probability weight from test point  $x_p$  to output  $y_m$  is given by:

$$pw_{p,m} = e^{(\|W_m(x-x_p)\|)} / SUM_m \quad \forall p \in \{1, \dots, P\}, m \in \{1, \dots, n-1\},$$

$$SUM_m = \sum_{p=1}^P (e^{(\|W_m(x-x_p)\|)}). \quad (12)$$

$W_m$  is a matrix that weights the importance of the input dimensions to that output dimension. This weighting matrix can perform important linear transformations to improve sensitivity. In this case, all off diagonal elements were zero so  $W_m$  simply scales the input dimensions. The probability weights are used to create discrete probability distributions:

$$PDF_m(y) = \sum_p (pw_{p,m}) \quad \forall p \in \{p : y_{p,m} \leq y\}, \quad (13)$$

where  $y_{p,m}$  is the  $m^{\text{th}}$  intermediate angle in the  $p^{\text{th}}$  test point. In other words, the probability of choosing an intermediate angle less than  $y$  is equal the the sum of the weights of the test points with the  $m^{\text{th}}$  intermediate angle less than  $y$ . A continuous distribution is created by adding the point  $PDF_m(y_{min}) = 0$  and then linearly interpolating between points. This adds a left bias to the distribution, but has negligible effect for large data sets and simplifies the

calculations. Points with very small proximities,  $e^{(\|W_m(x-x_p)\|)} \leq 10^{-6}$  are ignored from  $SUM_m$  and  $PDF_m$  primarily for numeric reasons.

The random path is created by selecting  $n - 1$  random points,  $rand_m$ , with uniform distribution from  $[0, 1)$  and then finding  $y$  such that the continuous  $PDF_{y,m} = rand_m$ . This random path can be used directly or indirectly. The exact angles can be used to find the cost using linear programming to find the segment lengths. Indirectly, the output set can be used as an initial point for a staged nonlinear optimization of angles with an embedded linear optimization for segment lengths. This optimization is described in Section 3.2.

For example, suppose 15 points were taken as shown in Figure 17 in the top left corner. These points were randomly generated and are used to express an unknown function for a one dimension input to a one dimension output. Consider a set of test points spaced along the input domain. For each test point, the resulting  $PDF$  is generated. Three examples of  $PDF$ 's for different test points are shown on the right of Figure 17. A set of distributions from a set of test points is meshed to form a surface showing the input dynamics of  $PDF$ , placed in the bottom left corner. To use the chart, an input value, representing the point of interest, would be selected defining one  $PDF$  (similar to ones on right). Then uniformly random numbers between zero and one would be generated. Each generated number, for a specific distribution function, would correspond to one output value. Each output would be assigned a cost. In the path planning case it would be from (9). Over the set of random numbers, the output with the lowest cost would be selected as the best output.

There are two stages in this stochastic optimization: generating  $PDF$ , the overall distribution function; and evaluating paths. Generating  $PDF$  must be done every time a new test

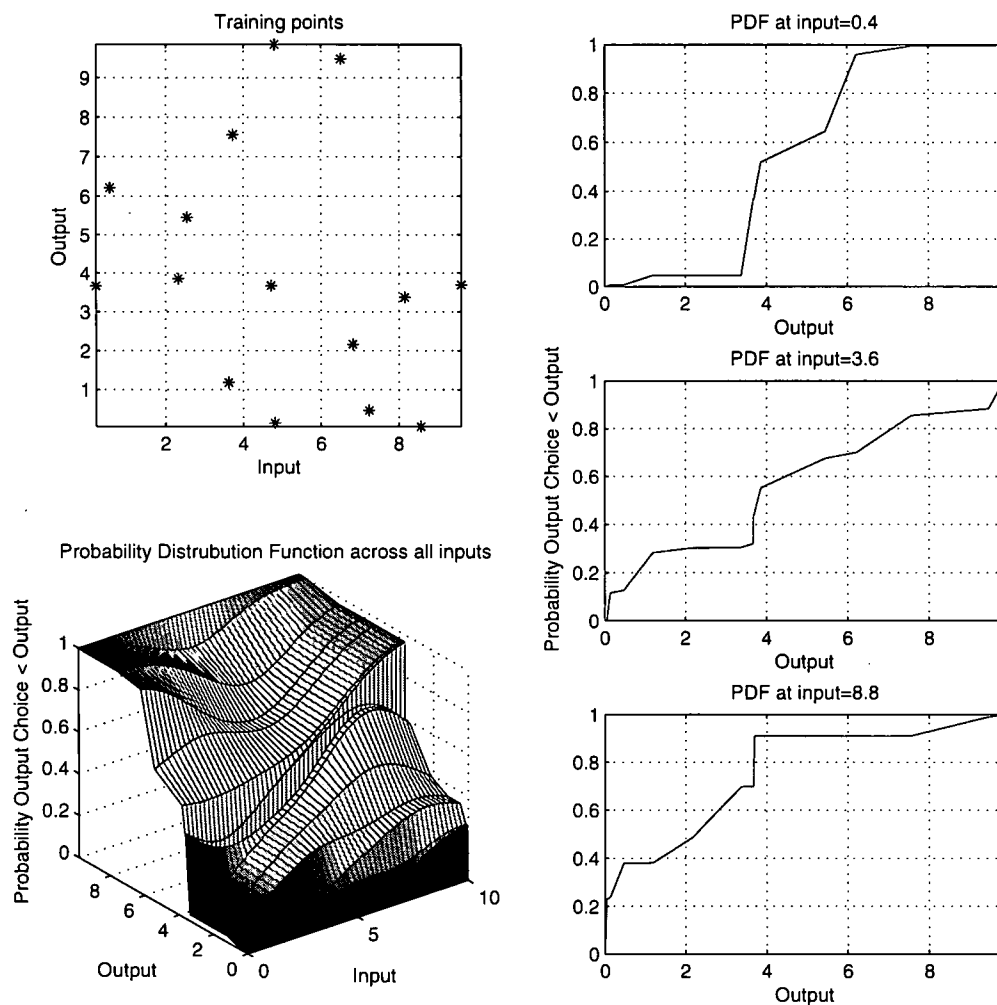


Figure 17: Example of generating probability distribution function. Test points in top left corner represent an unknown functional relationship. For a specific input value, an individual probability distribution function is built (PDF). Three different examples are shown on the right. A surface of distribution functions across all inputs showing PDF response to input changes is in the bottom left corner.

point is given. Computational time scales linearly with  $pw$  complexity, training set size, and output dimensions. The computational cost of generating  $PDF$  can be offset by reusing  $PDF$  to generate many paths. The computational cost of evaluating paths scales linearly in number of paths to test but for reasonable numbers of paths to test, generating  $PDF$  is noticeably longer. For direct use, the computation time scales near linearly in output dimension due to the linear programming. For indirect use, computational time scales exponentially in output dimensions due to the nonlinear programming. This method also includes the computational cost of generating a good training set for each set of MAV parameters or other constraints. Though generating the training set may be time consuming, fortunately this can be done autonomously. Stochastic results should not be used to increase the training set, since it biases the search away from the optimal paths to merely good paths.

As seen in Figure 18, results of this stochastic optimization have been good for the direct method. Using a reasonable size of test paths (200) and optimizing the parameters, output costs increases averaged about 90 seconds by mean and 70 seconds by median with an average of 45 seconds to compute. Some, but very few, paths had high costs. Using the indirect method also produced good results, but the computational time increased to the point that a global search could have been done in lieu of a stochastic optimization.

There is some room for improvement with this method to decrease cost or decrease computation time. Cost could be decreased by using a cost based termination criterion. Paths would be generated until the cost approaches the predicted cost. Computational time can be decreased by simplifying the proximity function, decreasing the training size and incorporating a Tabu

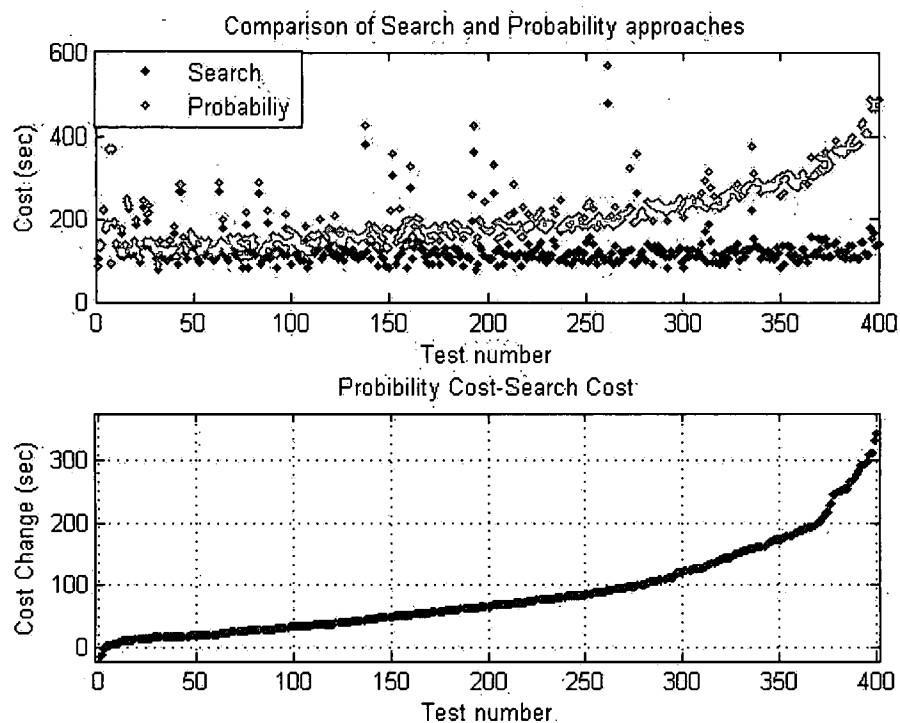


Figure 18: Results of comparison set with stochastic optimization compared to DGS (cost change mean=87 seconds, median=67 seconds). Cost is higher than DGS for the stochastic method, but stays close for most cases. Computation time is a fourth to a fifth that of DGS.

condition to restrict retesting similar paths. These predictions are likely, based on the method and examination of results but cannot be guaranteed without testing.



### 3.4 Dynamic Programming

Dynamic programming is a method to reduce computation complexity of optimizations. It consists of considering only optimal solutions to subproblems and works naturally for discrete systems. For this problem, dynamic programming found the turning pattern with shortest time to turn between two headings regardless of ending displacement. In addition, by not restricting the displacement, a lower bound on turning times can be generated. The motivating concept is that a pattern yielding quick turning times would have an advantage over arbitrary turning patterns. Since the settling lengths are piecewise linear and minima will occur at linear boundaries, the paths should be generated on the grid defining settling lengths. For this problem, settling length linear boundaries are starting and finishing angles in  $10^\circ$  increments and wind speed in 2 knot increments.

The process of finding optimal turns is to find the optimal turn progression for the first increment to the last increment by building from smaller increment turns. For example to find the quickest turn to the fifth increment ( $50^\circ$ ) only five progressions need to be tested:  $10^\circ$  and then a  $40^\circ$  turn; the best  $20^\circ$  progression and then a  $30^\circ$  turn; the best  $30^\circ$  progression and then a  $20^\circ$  turn; the best  $40^\circ$  progression and then a  $10^\circ$  turn; and a  $50^\circ$  turn. These five progressions made into paths are shown in Figure 19. All combinations of subangles produces 16 combinations to test, but those other combinations use subcombinations that have been shown suboptimal for lower segment turns. This optimal turning progression is generated for every starting orientation to every finishing orientation for every wind speed forming a database.

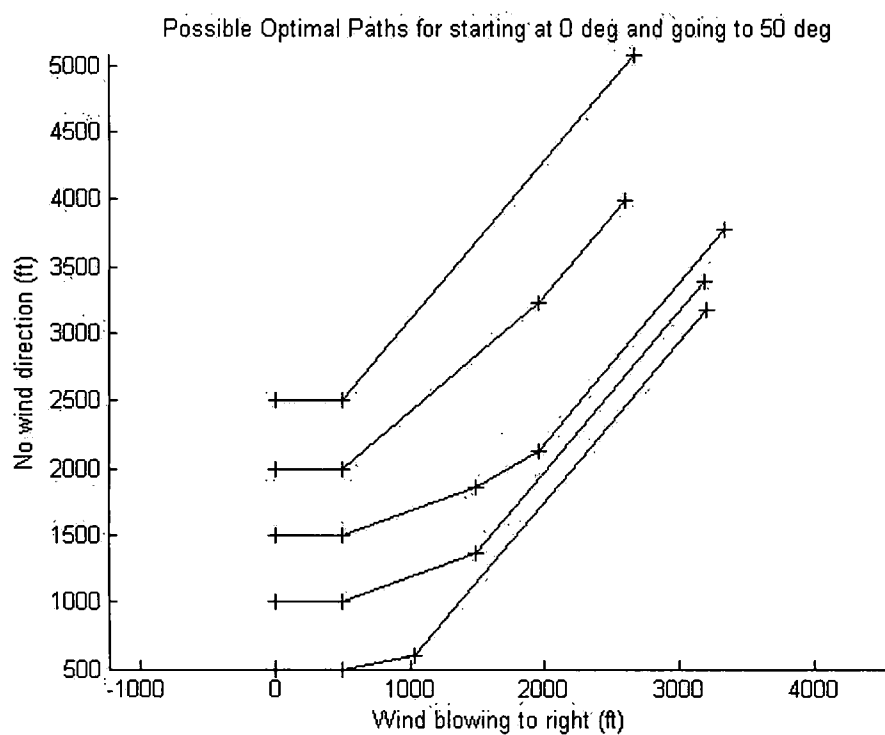


Figure 19: Paths considered for optimal turn progression of  $50^\circ$  to the left from flying with the wind.

To write the formula mathematically, some variables and notation should be declared. Since the settling lengths are a function of angles with respect to the wind, the initial angle ( $r$ ) must be known. Turns are in grid increments,  $\delta$ , and the aggregate change is  $\delta I$  to a final heading of  $r + \delta I$ . The aggregate turn is broken into turns composing a progression of turn increments,  $turn(r, r + \delta I) = [a, b, \dots, g]$  where  $a + b + \dots + g = I$ . All but the last turn is an optimal turn progression for a smaller increment,  $turn(r, r + \delta I) = [turn(r, s), I - (s - r)/\delta]$ . For a direct turn of  $\delta I$ ,  $s = r$  and no turn progression is needed to be added to the final turn. The time it takes to settle from orientation  $r$  to  $s$  is  $Cost_{r,s}$ . Using  $\arg \min_i (f(i))$  as the argument,  $i$ , that minimizes  $f(i)$ , the final turn increment in the progression is,

$$turn_{end}(r, r + \delta I) = \arg \min_{i \in [1, I]} (Cost_{r, r + \delta i} + Cost_{r + \delta i, r + \delta I}). \quad (14)$$

So, the turn progression is

$$turn(r, r + \delta I) = [turn(r, r + \delta turn_{end}), turn_{end}]. \quad (15)$$

The number of turns in the optimal path will depend on the length of  $turn(r, r + \delta turn_{end})$  and not on an artificial bound. Dynamic programming simplifies the search to the point that an arbitrary number of segments can be considered, where before the number of intermediate angles had to be known to size the output dimension. Intuitively, paths with no limit on number of turns should allow for more flexible and better paths.

Geometric constraints restrict some turn progressions from reaching all finish locations. Any segment can be lengthened without violating a constraint, so the possible finish locations lie outside the path. If the path were to contain segments that form a positive and negative basis in both cartesian directions, any point can be reached. A path that satisfies this condition

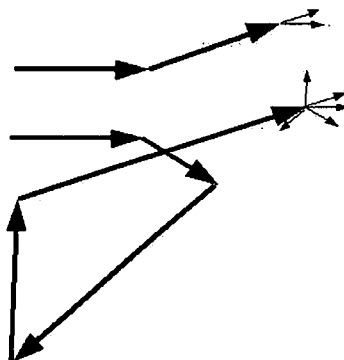


Figure 20: Viable paths are possible in the range reached by extending the segments. Direct turns (top) have a limited range while long turns ( $> 180^\circ$ , bottom) are viable for the entire space.

would be a long turn, meaning that the MAV's heading would sweep an angle larger than  $180^\circ$ . For example, to turn  $20^\circ$  clockwise, a short turn would be turning  $20^\circ$  to your right (clockwise) and a long turn would be turning  $340^\circ$  to your left (counter-clockwise). The math supports this as shown in Appendix A. It is shown graphically in Figure 20.

The quickest turn was normally a short turn and was not guaranteed to satisfy displacement constraints (2). To ensure that a possible route was found, the quickest long turn was also stored. With the three dimension grid on which settling lengths are built (starting heading, finishing heading and wind speed), every test point will be surrounded by at most eight points. Points that lie on a grid plane, line or point will be surrounded by less, but additional points were added to keep the method uniform. Each point was used to create a short-turn path and a long-turn path by adjusting the initial and final headings of the built paths to fit the desired headings. This adjustment should only add a minor impairment since perturbations from the grid will be small due to small grid size. The other option would be to add a segment for getting

on and off the path. These segments would be at least the minimum segment length. Often, the minimum segment length is longer than the incremental length so the latter path would be longer. With the headings known and a specific displacement, the segment lengths and costs would be found through linear programming. Of the set of sixteen paths tested, the lowest cost path would be selected.

This method consistently produced paths with lower cost (Figure 21). The advantage of an unrestricted number of turns allowed some paths to be significantly better than the Global Search method. This was especially true for paths with high DGS cost ( $> 175$  sec). Path costs increased 105 seconds by mean but only 47 seconds by median. Since the method always produces a path, it lends itself to producing a very long but finite path. This was the case and a very small number of extraordinary costs significantly skewed the mean. The method was extremely fast taking 0.15 seconds on average to generate a path.

One disadvantage of this method is that it offers little room for improvement. One possible method would be to take points further and further away from the test point. Another could be incorporating these turn progressions as the turns of another method. For example the global search or stochastic optimization could be done for a three turn path. Then those turns could be replaced with an equivalent composite turn from the dynamic programming library of turn progressions.

The other approach to improving this method would involve relaxing the constraint that all turns are in the same direction. A reason to challenge this heuristic is that for Dubin's car systems, most of the paths involve turns in differing directions. If this heuristic were completely relaxed, then solving the optimal turn progression would be NP-hard. The constraint can be

partially relaxed, allowing one or two changes in path direction, and remain solvable though much more difficult than the current problem. Hopefully this relaxation would address the issue causing the exorbitant cost of a few paths.

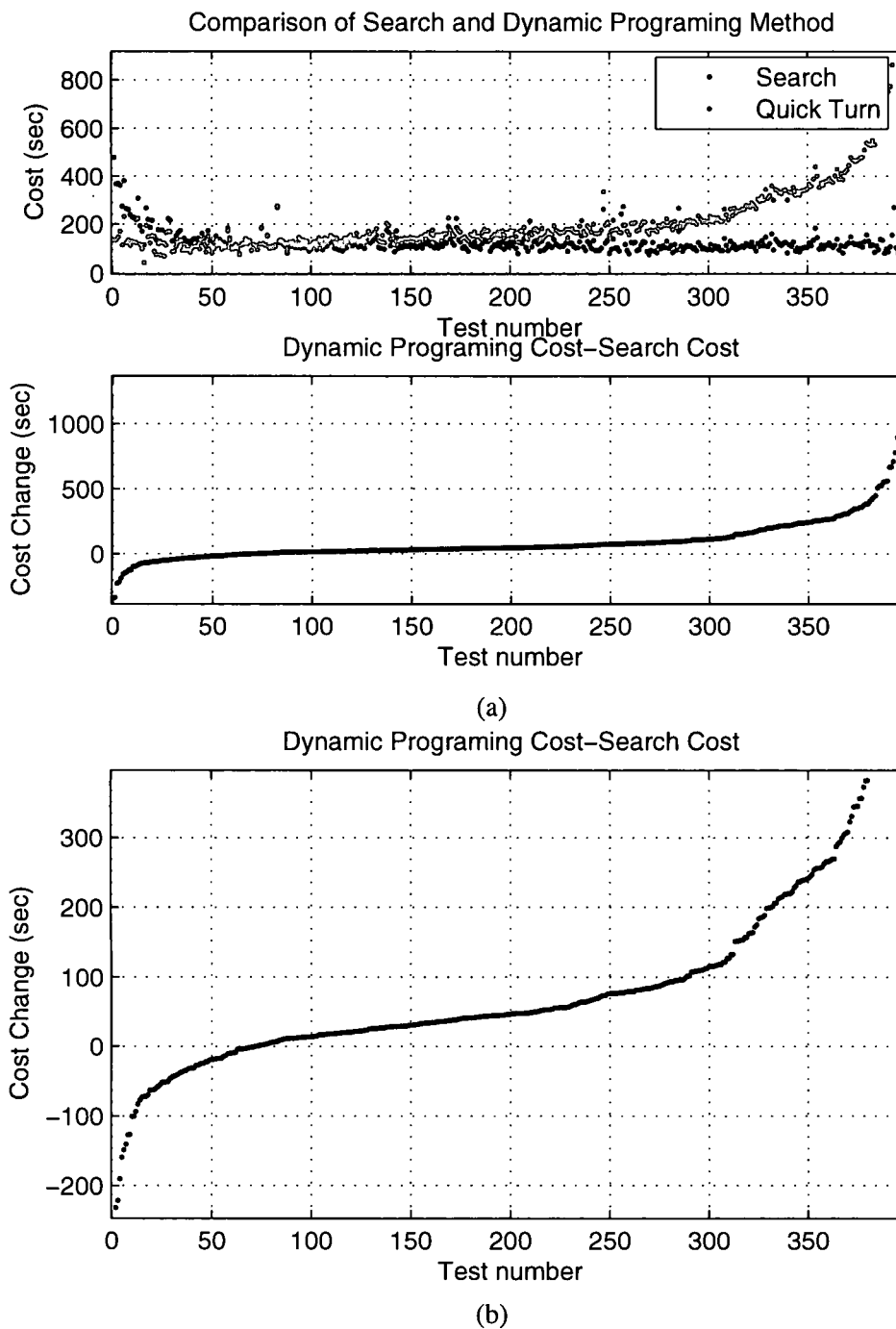


Figure 21: Results of comparison set with dynamic programming compared to DGS (cost change mean=105 sec median=47 sec), area of interest enlarged in (b). Results are normally very good, but a few cases had very large costs. Dynamic programming allowed more turns. Using more turns allowed lower costs than DGS, especially when DGS had high costs (> 175 sec). Computation was very fast (about 0.15 sec).

## **CHAPTER IV**

### **CONCLUSION**

The asymmetric settling length constraint adds significant complexity to the path optimization problem. No global optimization method seems feasible, though four near optimal methods can be used. The discretized global search (DGS), stochastic optimization, and dynamic programming methods all work well and have different advantages.

The neural network does not work well and had poor computation time. The primary reason is the training data, created from DGS, borders constraints and may have two regions of output for close inputs. Having an input dimension of five, precluded a narrow spacing of neurons due to the required number of neurons. Because the deterministic nature of the network was causing the problems, a stochastic method was created to address these issues. The stochastic method was generally able to generate paths similar to the training data. The cost did increase but stayed near the DGS cost.

The lowest cost results often came from the DGS. Dynamic programming normally had lower costs for test points with high DGS cost ( $> 175$  sec), see Figure 21. The primary disadvantage of DGS was its computation time due to the number of points to evaluate. Even with doing iterations of refinements, DGS took the longest to compute for a comparable number of segments. This could be mitigated if the number of points tested could be reduced without



raising the cost. When considering computation time, the probability method and dynamic programming method had low costs and processed quickly. The best method was the dynamic programming; it runs quickest (0.15 sec) and normally has a lower cost than the probability method. The cases which the probability method will likely do better can be discerned by the cost (about 25% of test points, see Figure 22). In other words, dynamic programming can be run and if the cost is above a threshold, a probability search will be done. Running the probability method first will not be as effective since there is no clear boundary of when dynamic programming will be better (see Figure 23). Example results of this hybrid approach are shown in Figure 24. Note that the results skewing the mean of dynamic programming costs high are replaced and the mean is near the median. The key parameter of this approach is the threshold value. Changing the threshold varies the average path cost and mean computation time (see Figure 25). The threshold may be set higher to preserve fast average computation times or lower to improve average path costs.

The results of this thesis are based on the assumptions of the analysis. First, the result time is the predicted time. This assumes that the MAV flies exactly on the path at the prescribed speed. As can be seen from Appendix B, as the MAV settles it follows a longer path than the waypoint path. Next, the controller performance is assumed to be satisfactory. Some of the paths the controller created had abrupt turns that may be unsatisfactory and require infinite settling lengths associated with it. Finally, singularity must be avoided. This means that initial and final headings must be unique and there must be some wind. The matrix inversions to generate the particular solution (6) and null solutions (7) require matrix inversion. Changing the format of the solutions can be done but would require more complicated analysis. The

calculations are fairly robust and perturbations from this singularity should have sufficient accuracy. Requiring nonzero wind speed is solely for scaling the cost. Changing the equations for zero wind is trivial.

The significance of this research was analyzing path planning as a discrete event optimization, developing a novel stochastic optimization method and applying the stochastic method and dynamic programming method to the path planning problem. Analysis shows that optimal, straight-line waypoint paths do not necessarily follow Dubin's path approximations for general settling length requirements. The novel stochastic optimization allows for approximation of a widely fluctuating relationship. Dynamic programming allows for generation of very complex paths by using a heuristic to limit the search.

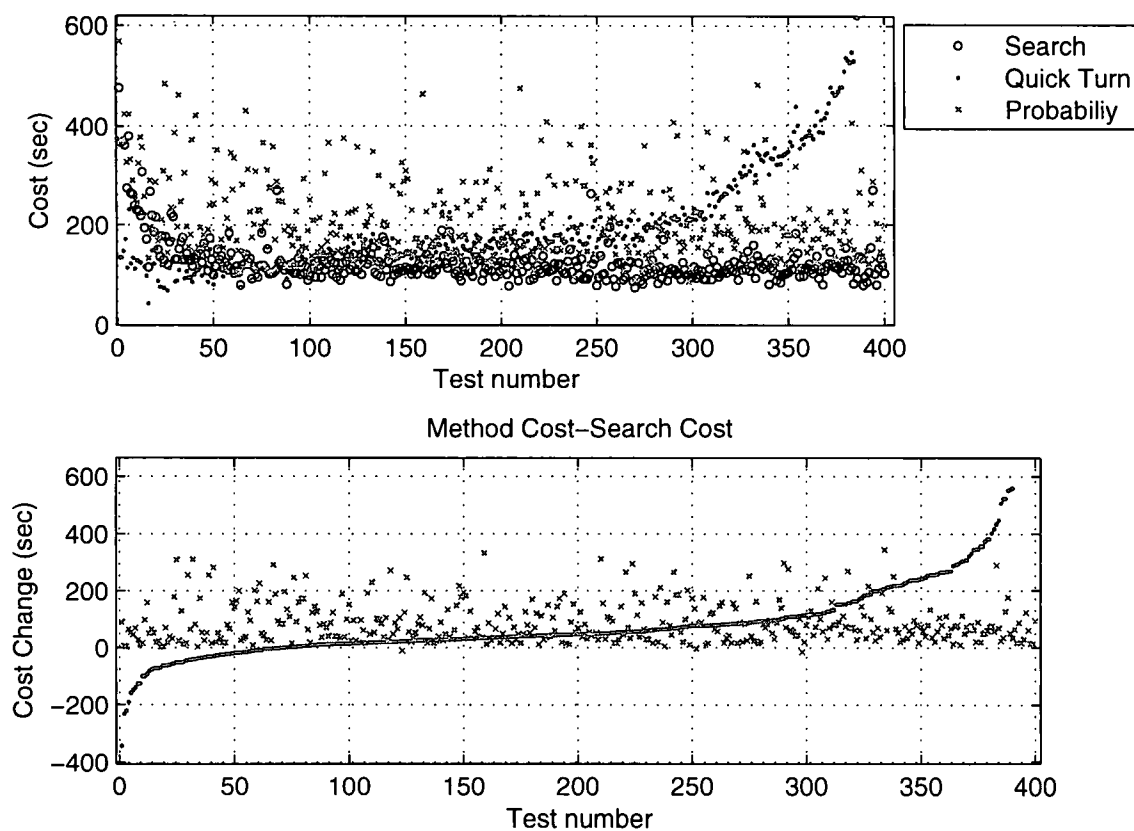


Figure 22: This plot is a comparison of the results of the different methods sorted by ascending cost change from the dynamic programming method. Though the cost ascends quite high, the dynamic programming method often has the best cost. For the cases when the cost is over 125 seconds, the probabilistic method often has better results than dynamic programming.

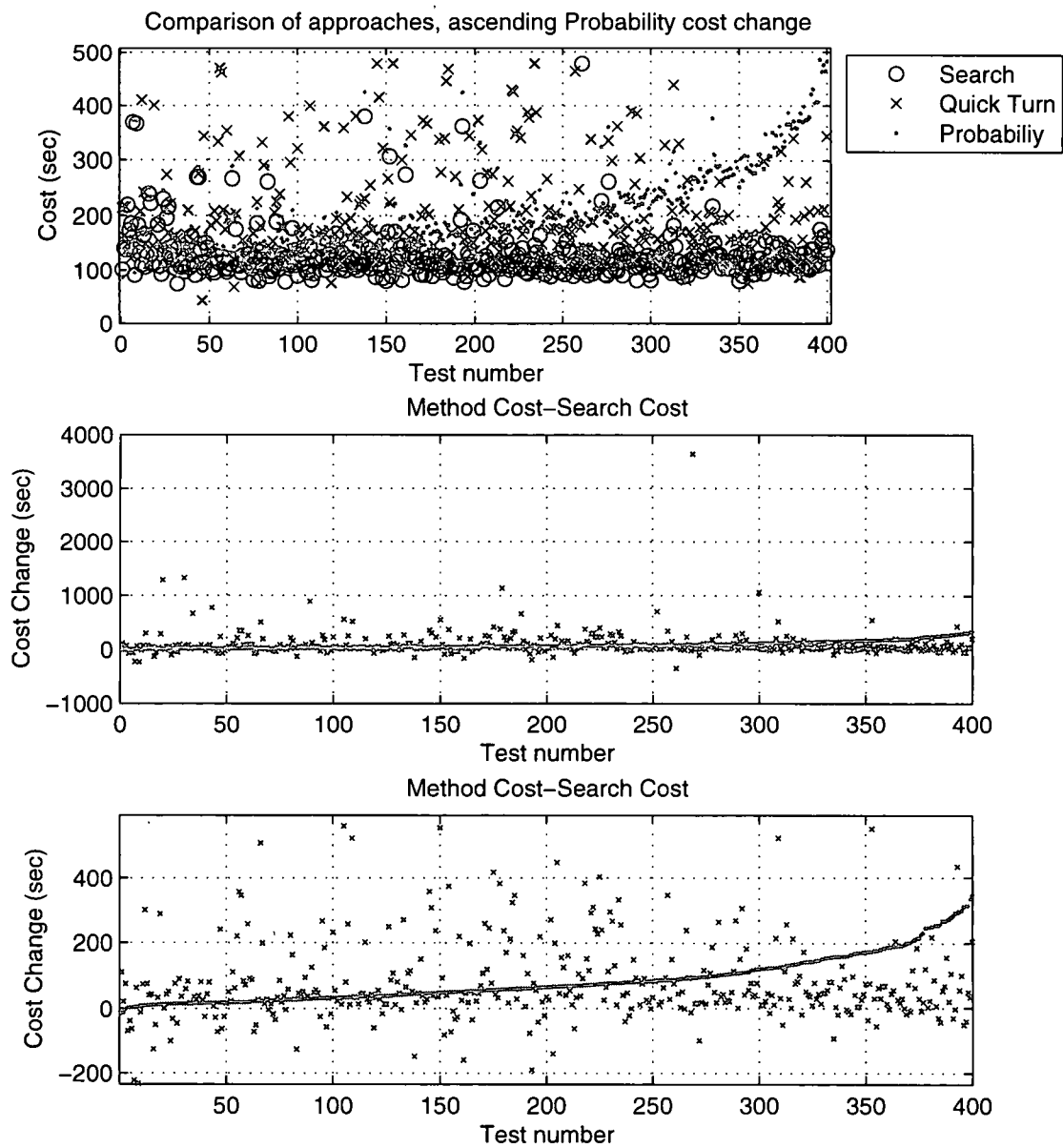


Figure 23: Here the results are sorted by ascending cost change from the probabilistic method. No order is seen among the other methods.

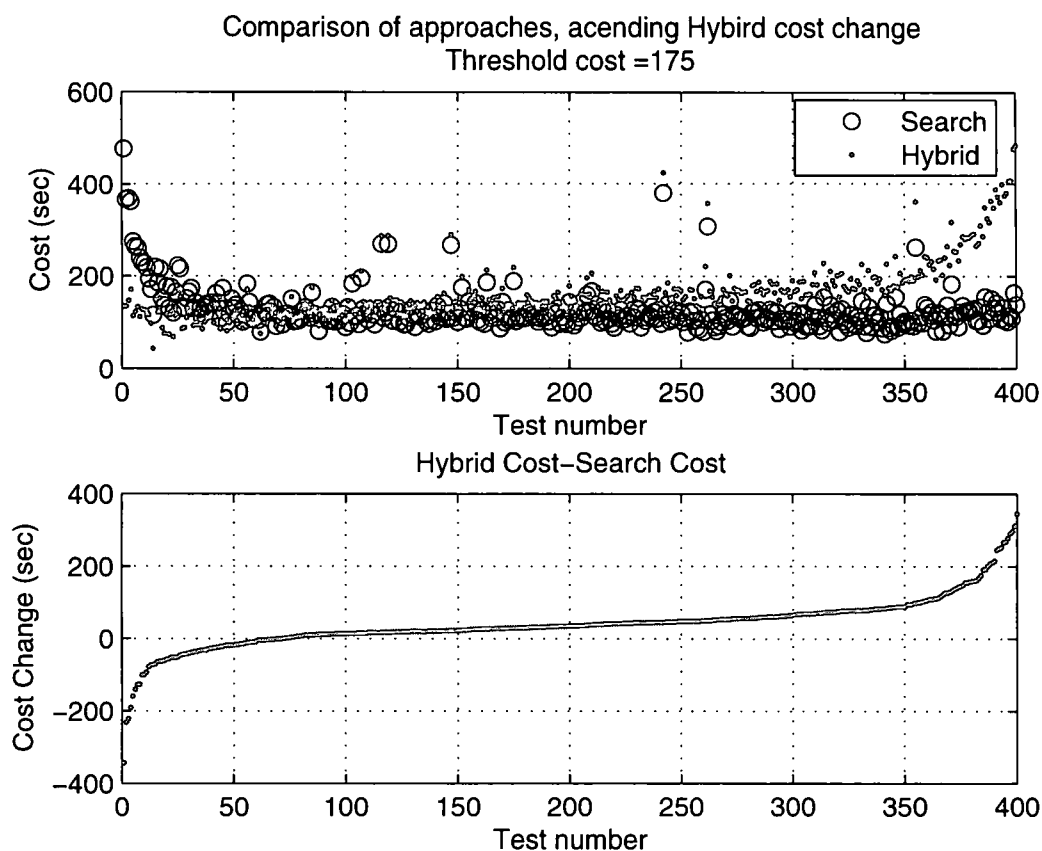


Figure 24: Results of comparison set with hybrid method compared to DGS (threshold=175 sec, cost change mean=105 sec and median=47 sec). Results are consistently very good, and the few cases of very large costs with dynamic programming are now normal. Computation remained much faster the DGS (about 15 sec total computation time).

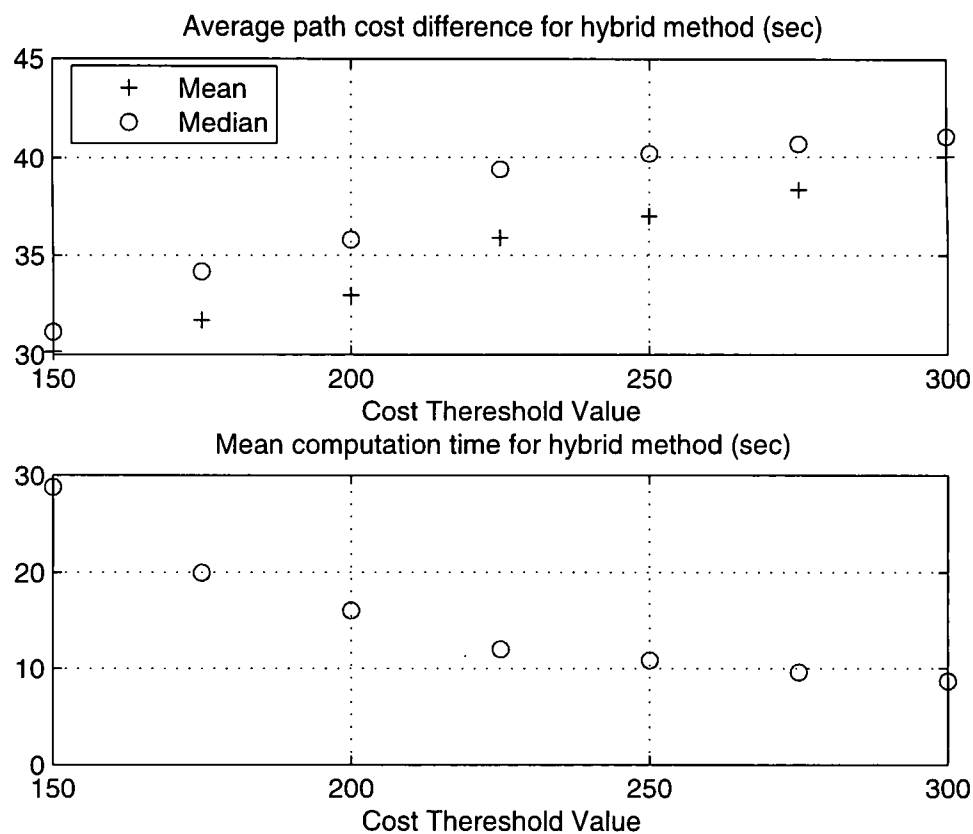


Figure 25: Results of comparison set with hybrid method compared to DGS (threshold=175 sec, cost change mean=105 sec and median=47 sec). Results are consistently very good, and the few cases of very large costs with dynamic programming are now normal. Computation remained much faster the DGS (about 15 sec total computation time).

## APPENDIX A

### Proof of Feasible Solution for Long Turns

Two sets of constraints limit the feasibility of a solution. First, the intermediate lengths must be longer than their settling lengths from (3),

$$g(\theta_i, \theta_{i-1}) \leq d_i = \beta_i \quad \forall i \in \{1, \dots, n-1\}. \quad (16)$$

Next, the initial segment length must be greater than or equal to zero,

$$0 \leq \frac{1}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_n - \theta_d) & -\sin(\theta_n - \theta_1) & \dots & -\sin(\theta_n - \theta_{n-1}) \end{bmatrix} \begin{bmatrix} d \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}$$

$$\frac{\sin(\theta_n - \theta_d)}{\sin(\theta_n - \theta_0)} d \geq \frac{1}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_n - \theta_1) & \dots & \sin(\theta_n - \theta_{n-1}) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}$$

$$c_{n,d}d \geq c_{n,1}\beta_1 + \dots + c_{n,n-1}\beta_{n-1} \quad (17)$$

where  $c_{n,i} = \sin(\theta_n - \theta_i) / \sin(\theta_n - \theta_0) = \sin(\theta_i - \theta_n) / \sin(\theta_0 - \theta_n)$ . Finally, the last segment must be longer than its settling length,

$$g(\theta_n, \theta_{n-1}) \leq \frac{\begin{bmatrix} \sin(\theta_d - \theta_0) & -\sin(\theta_1 - \theta_0) & \dots & -\sin(\theta_{n-1} - \theta_0) \end{bmatrix}}{\sin(\theta_n - \theta_0)} \begin{bmatrix} d \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}$$

$$\frac{\sin(\theta_d - \theta_0)}{\sin(\theta_n - \theta_0)} d - g(\theta_n, \theta_{n-1}) \geq \frac{1}{\sin(\theta_n - \theta_0)} \begin{bmatrix} \sin(\theta_1 - \theta_0) & \dots & \sin(\theta_{n-1} - \theta_0) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}$$

$$c_{0,d}d - g(\theta_n, \theta_{n-1}) \geq c_{0,1}\beta_1 + \dots + c_{0,n-1}\beta_{n-1} \quad (18)$$

with  $c_{0,i} = \sin(\theta_i - \theta_0) / \sin(\theta_n - \theta_0)$ .

The first set of constraints in (16) is unbounded on the positive side. The second set of constraints only constrain one side of each  $\beta_i$ , but the side depends on the sign of  $c_{n,i}$  for (17) and  $c_{0,i}$  for (18). If  $c_{n,i} < 0$ , then the coefficient of  $\beta_i$  in (17) is negative and  $\beta_i$  is still unbounded in the positive direction. For (18), the equivalent requirement of guaranteeing unbounded  $\beta_i$  is  $c_{0,i} < 0$ . These conditions must be met for a single  $\beta_i$ . Even if all other coefficients form two sided constraints, the unconstrained  $\beta_i$  can be made arbitrarily large so that the positive and negative constraints of another  $\beta$  do not overlap.

Now the concept of short ( $< 180^\circ$ ) and long ( $> 180^\circ$ ) turns comes into play. Consider that in a modulus of  $2\pi$ , each angle can be reached by a clockwise turn or a counterclockwise turn. If  $\sin(\theta)$  is positive, a short turn would be clockwise and a long turn would be counterclockwise. If  $\sin(\theta)$  is negative, a short turn would be counterclockwise and a long turn would be clockwise. So for an angle change of  $\theta_n - \theta_0$ , a short or long turn can be done and still reach the same final orientation. Note that for  $c_{n,i}$  and  $c_{0,i}$  to be negative, the sines must have opposite signs. Since the argument of the sines can be written with respect to the same angle, it implies that the short turn to one of the angles is in the opposite direction as the short turn to the other. In other words, one angle lies to the left of the reference angle and the other to the right. This constraint is shown graphically in Figure 26. In terms of linear algebra, vectors constrained to a positive direction must have a minimum of three vectors to form a basis of two cartesian dimensions. The third segment must be unique to the first two and therefore must have a component in the opposite cartesian direction to each of the first two segments.



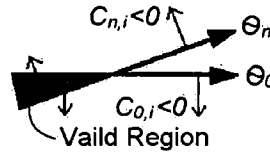


Figure 26: This shows the region for  $\theta_i$  that guarantees a valid solution.

Constraining an angle  $\theta_i$  to lie in the region guaranteeing validity, means that this segment lies in the opposite direction of the short turn to the final heading. It is fair to assume that all turns in a path follow the same direction (clockwise or counterclockwise). This is a heuristic for finding optimal solutions. If it were to be guaranteed, more would have to be known of the settling length function,  $g(\cdot, \cdot)$ . Taking a series of turns that lie in the opposite direction of the short turn of the final heading will cause the MAV to sweep an angle greater than  $180^\circ$ . This path is considered one long turn though all individual heading changes are interpreted as short turns due to the underlying control.

That any long turn path creates a valid solution can be seen by considering the directed segment (positive vector) example. To guarantee a solution, the basis (segment headings) must cover the range space. To cover the range space, some segment or combination of segments must have a component opposite to the first two segments. In the equations, this means that  $c_{n,i} < 0$  and  $c_{0,j} < 0$  for  $i, j \in \{1, \dots, n-1\}$ , so the coefficient constraint need not be met by the same angle.

This condition is merely sufficient and not necessary. It only becomes necessary if all turns are required to be in the same direction and the displacement distance is arbitrary. With those additional constraints, long turns are the only paths that will meet constraints. Also inherit in

this proof is that all angles are unique. Repeated angles cause some of the statements to become redundant or singular. In all, using long turn paths provides a nonintuitive solution that assists in finding optimal paths by guaranteeing a valid path.

## APPENDIX B

### Settling Behavior

The simulations of the MAV's turning characteristics were done at two knot increments and heading increments of  $10^\circ$ . The MAV's airspeed was 24 knots and wind speed ranged from 0 to 18 knots. Symmetry about the wind direction only required initial headings of  $0^\circ$  to  $180^\circ$  with respect to the wind direction to be simulated. Turns were done from  $120^\circ$  counterclockwise to  $120^\circ$  clockwise. Plots of a few initial headings and wind speeds are shown in Figure 27 to 29. For turns that did not settle in the simulation, either estimates of the settling length were made or the settling length was set to a large value. For turns beyond the range tested, settling lengths were set to large values that linearly increased with turn magnitude. Simulation results were linearly interpolated in heading and wind speed to form a continuous settling length function. This settling length function,  $g(\cdot, \cdot)$  is shown for a few wind speeds in Figure 30 to 32. To improve path quality, a minimum length was added to all settling lengths. This is in addition to the settling lengths shown.

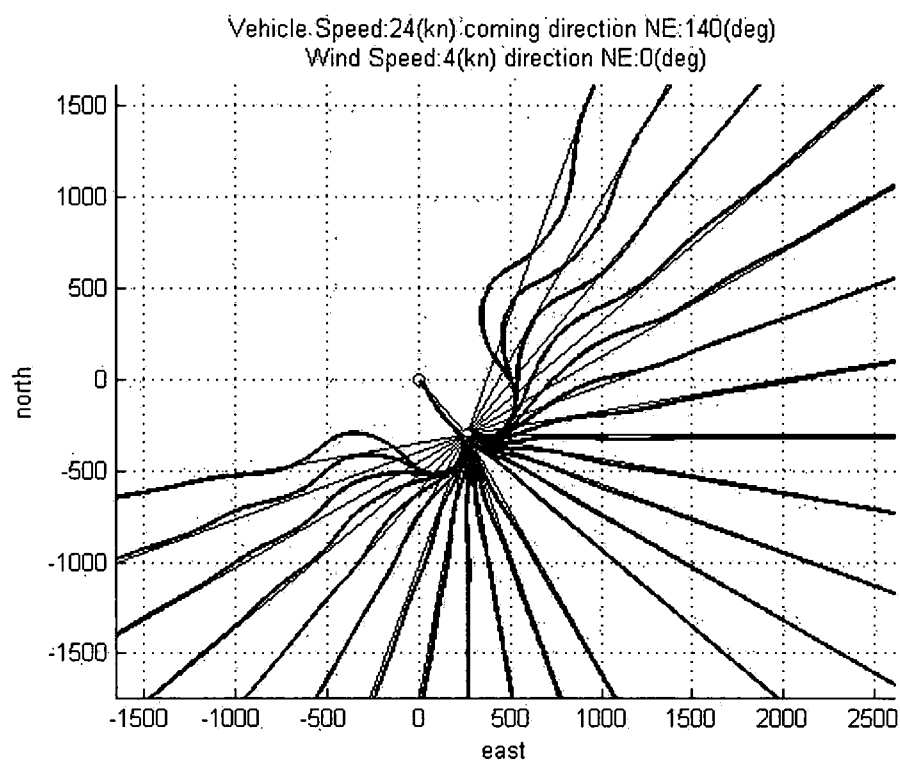
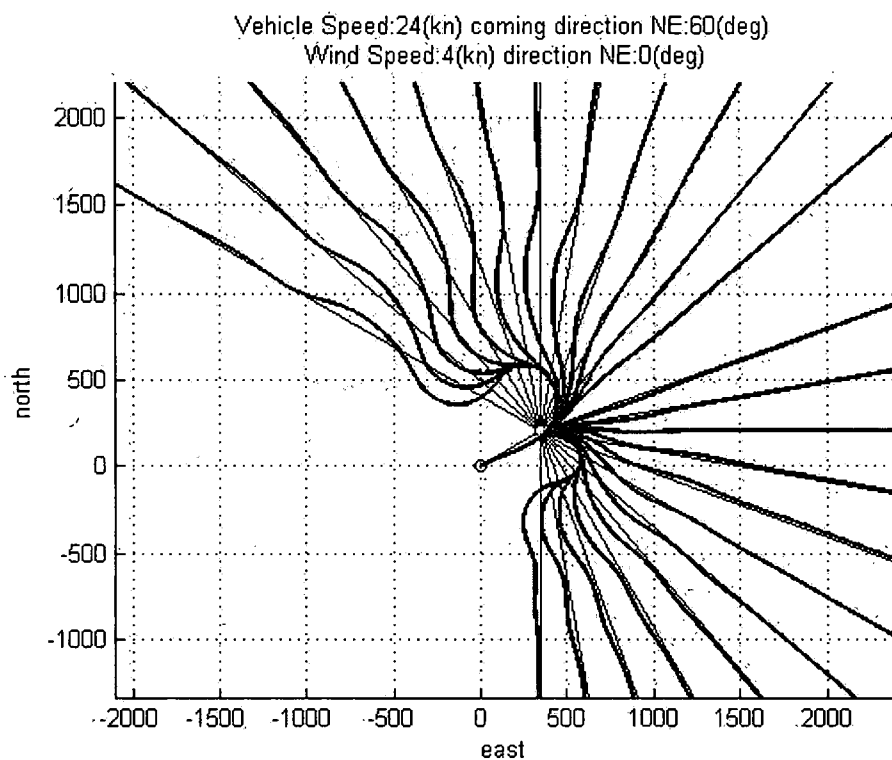


Figure 27: Settling behavior for wind speed of 4 knots.

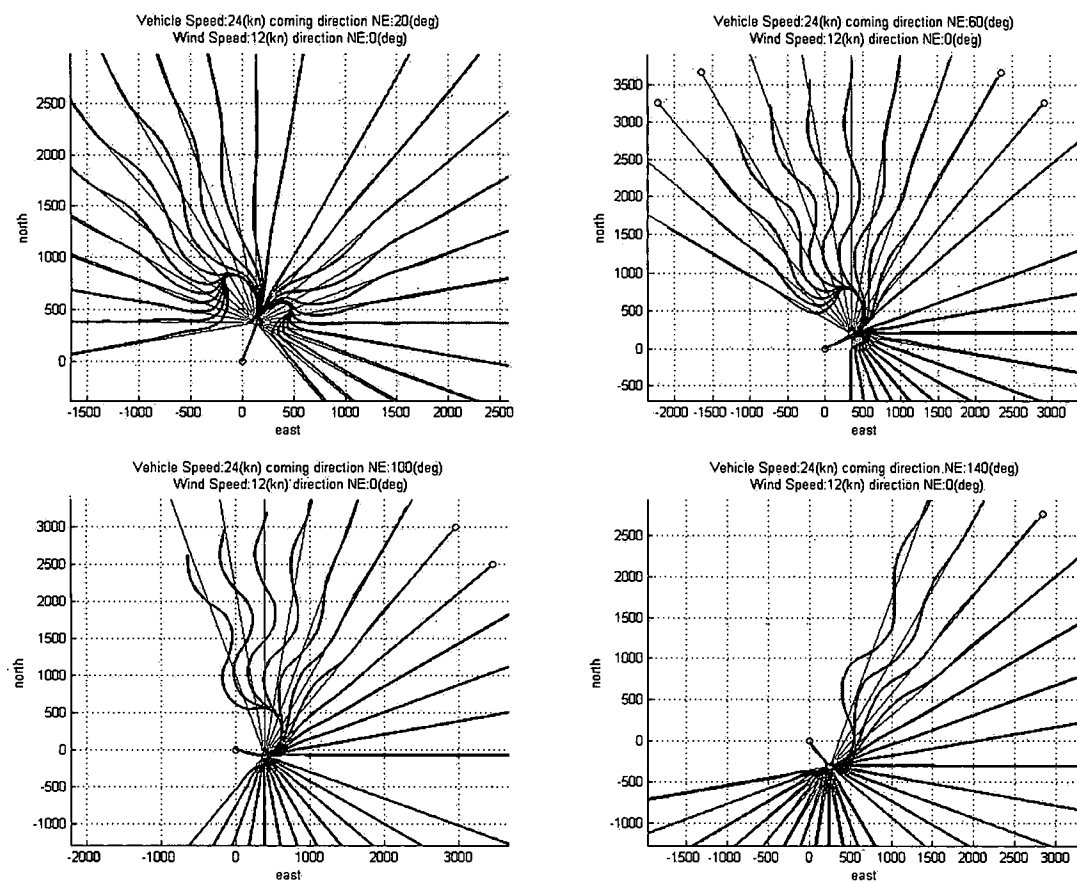


Figure 28: Settling behavior for wind speed of 12 knots.

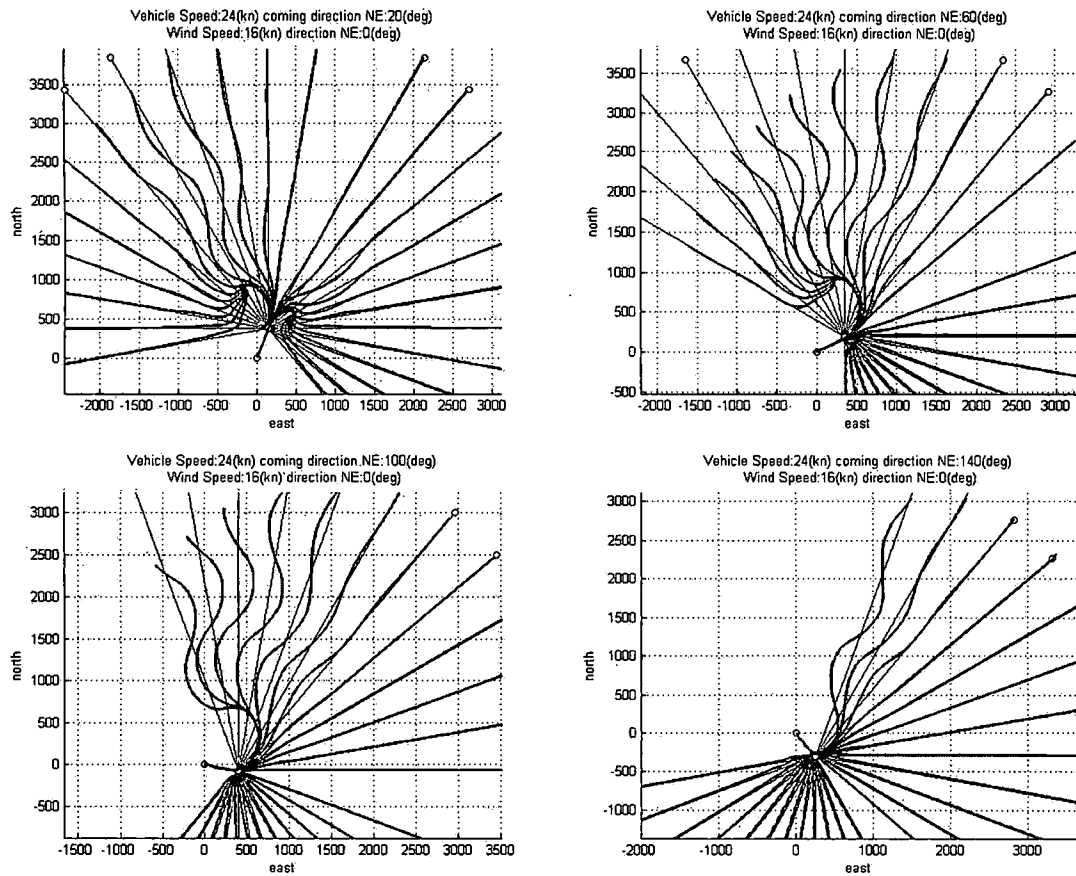


Figure 29: Settling behavior for wind speed of 16 knots.

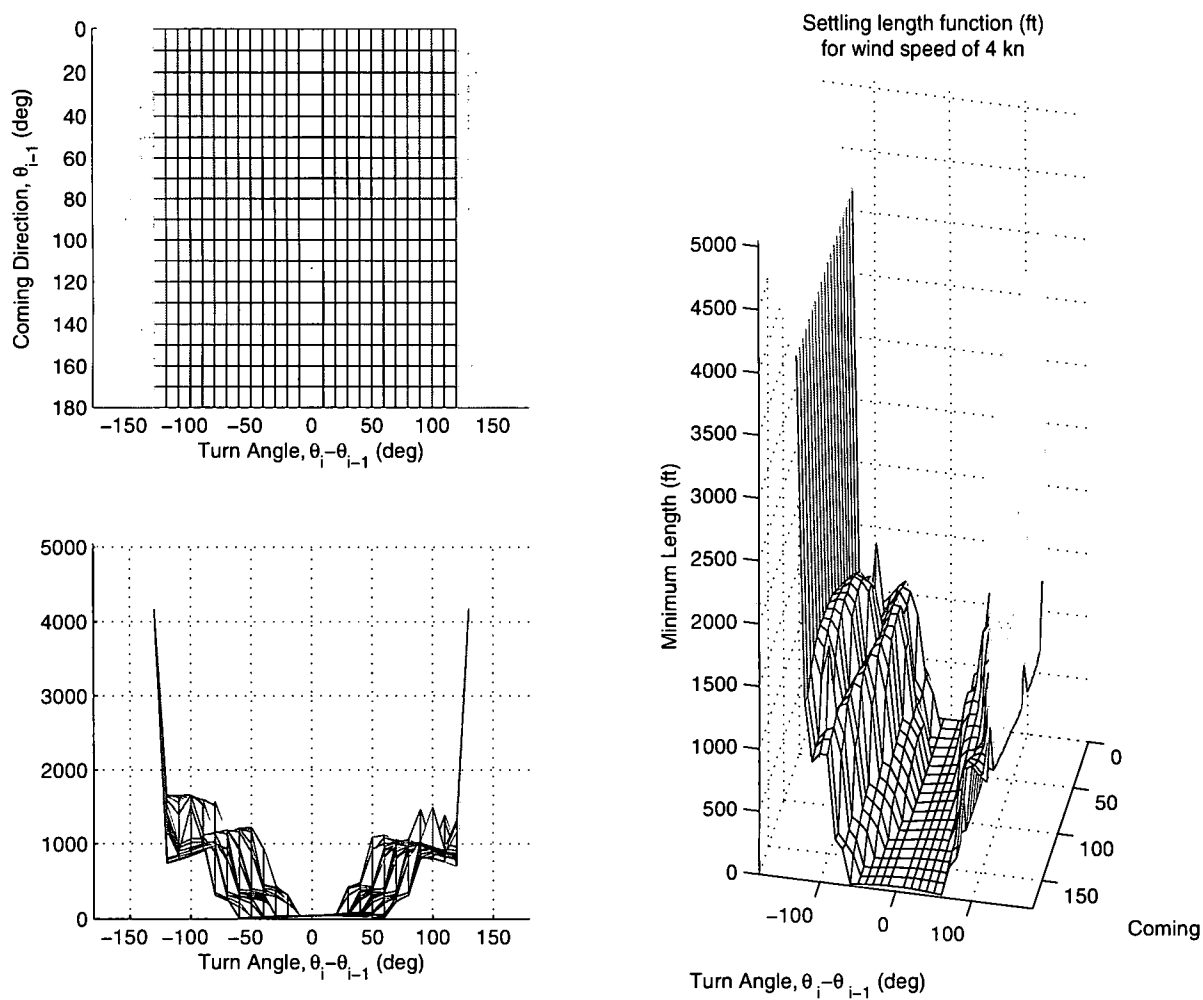


Figure 30: Settling length function for wind speed of 4 knots.

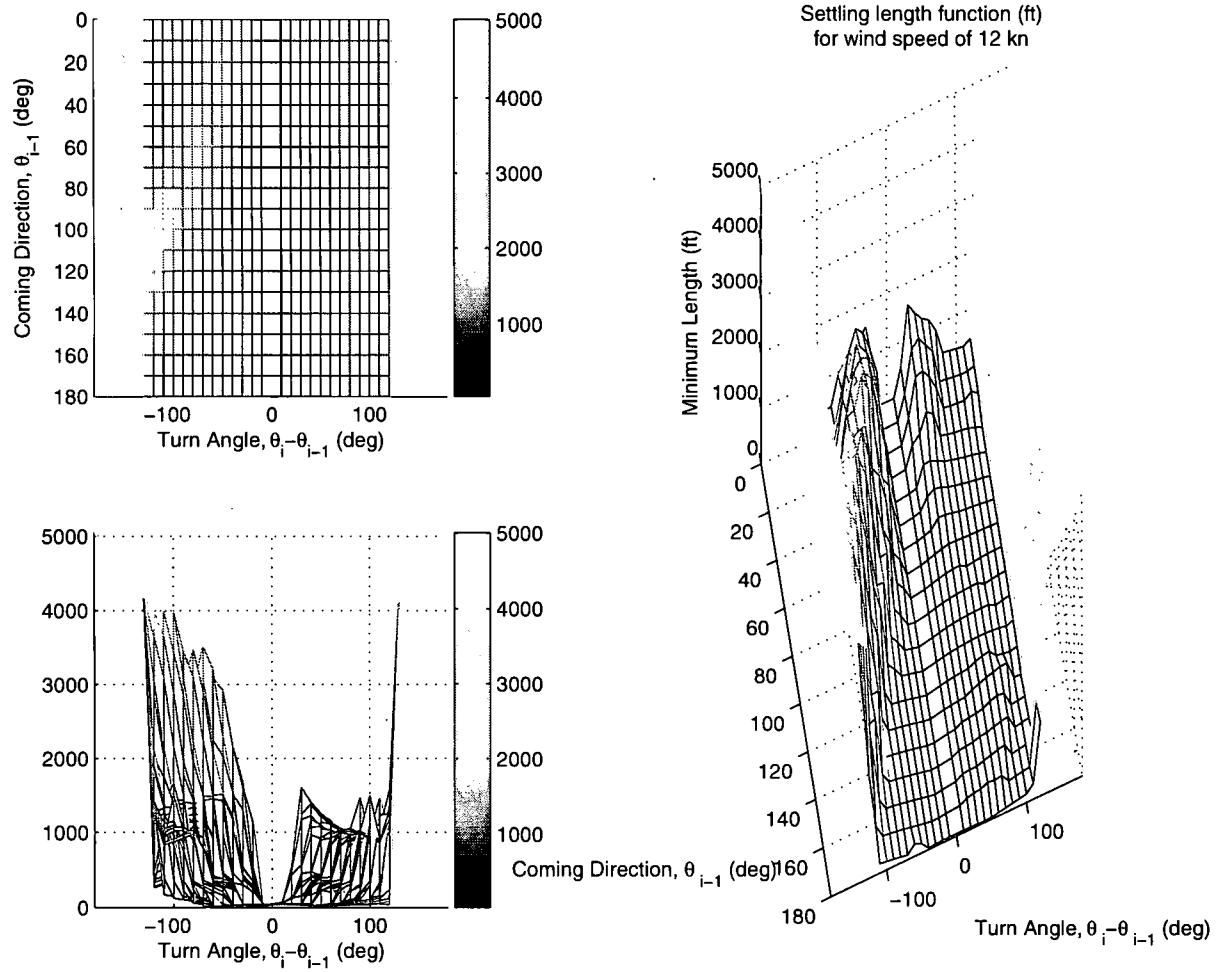


Figure 31: Settling length function for wind speed of 12 knots.



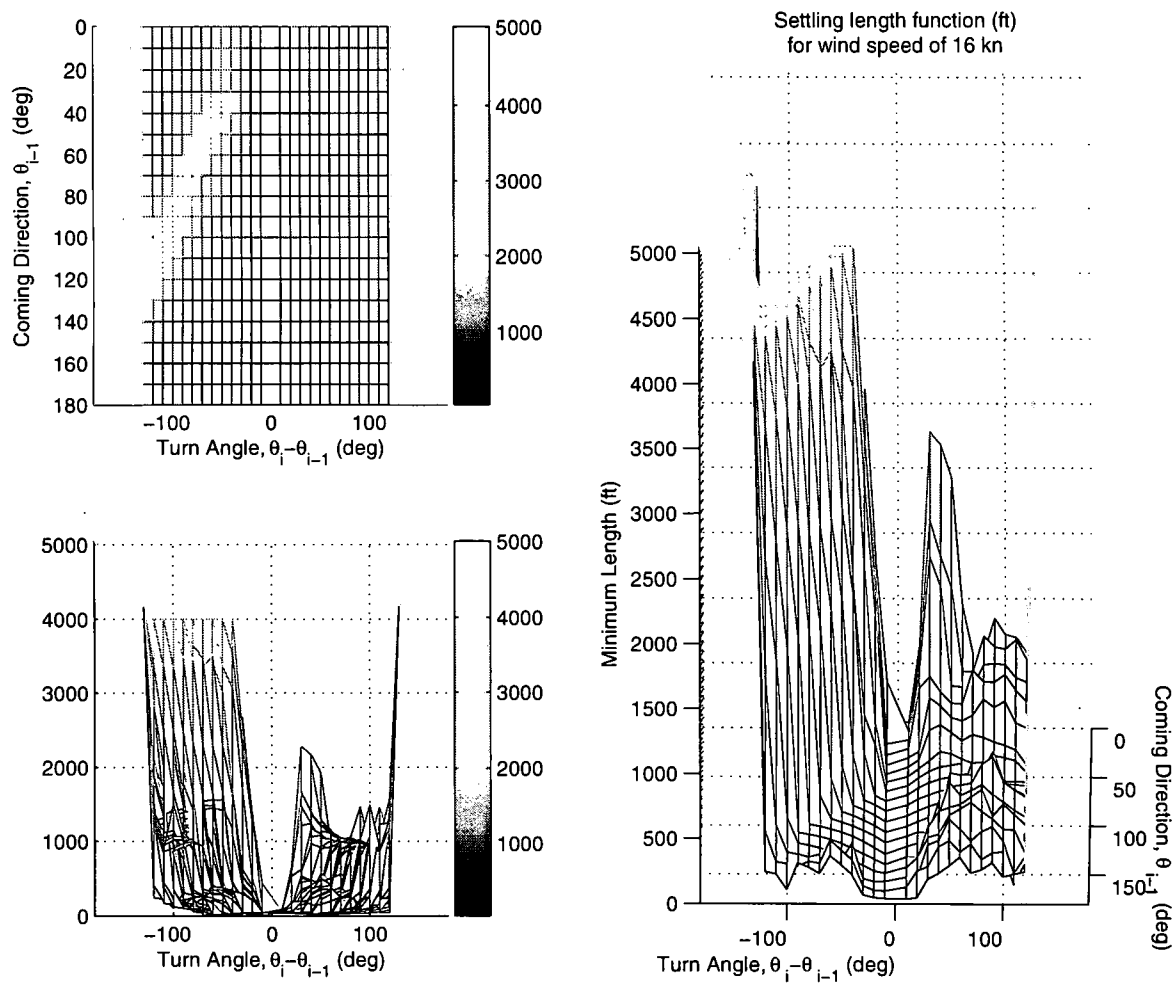


Figure 32: Settling length function for wind speed of 16 knots.

## BIBLIOGRAPHY

- [1] A. Ryan, "A mode-switching path planner for UAV-assisted search and rescue," Master of Science in mechanical engineering, University of California, Berkeley, Spring 2003.
- [2] N. Ceccarelli, J. J. Enright, E. Fazzoli, S. J. Rasmussen, and C. J. Schumacher, "Micro UAV path planning for reconnaissance in wind," in *Proceedings of American Control Conference*, pp. 5310–5315, Jul 2007.
- [3] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and perscribed initial and terminal positions and tangents," *Americian Journal of Mathematics*, vol. 79, pp. 497–516, Jul 1957.
- [4] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerical Math*, vol. 1, pp. 269–271, 1959.
- [5] R. C. Prim, "Shortest connection networks and some genieralizations," *Bell Sys. Tech. Journal*, vol. 36, pp. 1389–1401, 1957.
- [6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for heuristic determination of minimum path cost," in *IEEE Trans. Sys. Sci. Cybern.*, vol. 4, pp. 100–107, 1968.
- [7] M. Shanmugavel, *Path Planning of Multiple Autonomous Vehicles*. Phd., Cranfield University, 2007.
- [8] I. H. Whang and T. W. Hwang, "Horizontal waypoint guidance design using optimal control," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 1116–1120, Jul 2002.
- [9] Y. Zheng and U. Özgüner, "Path routing with switch-back avoidance for autonomous vehicles," in *Proceedings of the 45th IEEE Conference on Decision and Control*, IEEE, Dec 2006.
- [10] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," *American Control Conference*, p. 6, Jun 2006.
- [11] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for small unmanned air vehicles," in *Proceedings of American Control Conference*, p. 7, ACC, Jun 2006.

R002593572

- [12] J. Osborne and R. Rysdyk, "Waypoint guidance for small UAVs in wind," *IEEE Conference on Decision and Control*, pp. 709–714, Dec 2006.
- [13] T. G. McGee, S. Spry, and J. K. Hedrick, "Optimal path planning in a constant wind with a bounded turning rate," in *Proc. of the AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA, 2005.
- [14] J. Sellen, "Direction weighted shortest path planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, (Nagoya, Japan), pp. 1970–1975, IEEE, May 1995.