3-23-2018

Shortest Path across Stochastic Network with Correlated Random Arcs

Stephanie M. Boone

Shortest Path Across Stochastic Network with Correlated Random Arcs

THESIS

Stephanie M. Boone, 1st Lt, USAF

AFIT-ENC-MS-18-M-109

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENC-MS-18-M-109

SHORTEST PATH ACROSS STOCHASTIC NETWORK WITH CORRELATED RANDOM ARCS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Stephanie M. Boone, B.S.

1st Lt, USAF

22 March 2018

AFIT-ENC-MS-18-M-109

SHORTEST PATH ACROSS STOCHASTIC NETWORK WITH CORRELATED RANDOM ARCS

THESIS

Stephanie M. Boone, B.S.
1st Lt, USAF

Committee Membership:

Lt Col A. J. Geyer
Chair

Dr. R. R. Hill
Reader

Abstract

This paper introduces a new approach to identify the shortest path across a stochastic network with correlated random arcs utilizing nonparametric samples of arc lengths. This approach is applied to find optimal aircraft routes that minimize expected fuel consumption for a given airspeed utilizing predicted wind output from numerical weather prediction (NWP) ensemble models. Results from this new methodology are then compared to the current fuel minimization route planning method that utilizes deterministic NWP wind data for arc lengths. Comparisons are also made to other previously proposed alternative fuel minimization methodologies that utilize mean and median wind data calculated from NWP ensemble wind data.

Acknowledgements

I would like to thank my husband for his sacrifice and support, without whom this would not be possible. In addition, I would like to thank my faculty research advisor, Lt Col Geyer, and committee member, Dr. Hill, for their guidance, patience, and support in development of this research. I would also like to thank Dr. Baker, Capt Boone, and Col Reiman for their subject matter expertise and guidance in development of this research. Lastly, I would like to thank my classmates for the countless study groups, late nights in the COA, and for making the stressful times more enjoyable.

Stephanie M. Boone

Table of Contents

Page

List of Figures

List of Tables

List of Acronyms

557th WW	557th Weather Wing
ACFP	Advanced Computer Flight Planner
AFB	Air Force Base
AMC/A3W	AMC Director of Weather
AMC	Air Mobility Command
APSP	*a posteriori* Shortest Path
ATC	Air Traffic Control
CCP	chance-constrained programming
CIF	Cost Index Flying
DCP	dependent-chance programming
DoD	Department of Defense
ft	foot
GFS	Global Forecast System
IID	independently and identically distributed
Klbs	kilopounds
knots	nautical miles per hour
LP	Linear Programming
m/s	meters per second

MAJCOM Major Command

MIF Mission Index Flying

MSL mean sea level

NCEP National Centers for Environmental Prediction

NL Newfoundland and Labrador

NM nautical miles

NOAA National Oceanic and Atmospheric Administration

NWP numerical weather prediction

RMSE root mean square error

SPP Shortest Path Problem

TAS True airspeed

WARP Worldwide Aeronautical Flight Planner

WS wind speed

SHORTEST PATH ACROSS STOCHASTIC NETWORK WITH CORRELATED RANDOM ARCS

I. Introduction

1.1 Overview

With the increased scrutiny on government spending, Air Mobility Command (AMC) has been looking for ways to reduce costs. Fuel has become the largest contributor to aircraft operating costs. As the biggest consumer of aircraft fuel in the Department of Defense (DoD), significant savings could come from more efficient flight planning [23]. According to Lt Col Vince Zabala, AMC's fuel efficiency program manager, energy costs for the Air Force total nearly $6.8 billion annually, with about 86 percent of that cost spent on aviation fuel [26]. AMC consumes approximately 56 percent, more than all other Major Command (MAJCOM)s combined. If improvements can be made to significantly reduce fuel consumption, AMC could potentially save millions of dollars. In fact, Heseltine [19] determined that $28M a year could be saved if the command saved as little as $200 per sortie.

1.2 Background

The culture in AMC surrounding fuel-efficiency has changed in recent years, but there is still room for improvement. By identifying and utilizing fuel-efficient routes, fuel consumption can be minimized throughout the MAJCOM. While a lot of factors have an impact on fuel efficiency in flight, winds aloft play a large role during long-haul flights. Accurate wind forecasts are vital to ensuring fuel efficiency during

flight planning; Inaccurate forecasts may result in over- or under-estimating the fuel necessary, which translates into wasted money [20].

AMC contractors currently use deterministic numerical weather prediction (NWP) models for aircraft route planning. Deterministic NWP models utilize a single forecast to estimate weather predictions, whereas ensemble models utilize an independently and identically distributed (IID) sample of forecast models to make predictions. These different forecasts (ensemble members) are generated by running multiple simulations with slightly different initial conditions and/or various perturbations of models [36]. The intent is that these model variations represent the range of uncertainty associated with initial weather conditions and yield a range of potential forecasts [14].

Krishnamurti *et al.* [27] compared ensemble models with their deterministic counterparts and found that the ensemble models illustrated superior forecasting skill over all of the individual models inspected. In the last 30 years, many experts have proposed replacing the traditional deterministic forecast with the ensemble mean forecast [32, 41, 46]. In the last 20 years, ensemble mean forecasts have consistently been found to outperform deterministic forecasts on average [4, 8, 13, 45, 46]. With this consistent improvement upon the traditional deterministic forecasts, the use of ensemble forecasting has become routine [17]. Most recently, Homan [20] used ensemble mean forecasts to predict fuel burn for long range flights and found that ensemble means generally provided more accurate estimates over the deterministic model.

1.3 Motivation

While the use of ensemble NWP data may be routine, it is not common practice in AMC. Therefore, its introduction may provide added value in aircraft routing and fuel estimates. To identify any added value, a technique must be developed that will leverage the uncertainty that is accounted for in the ensemble NWP data.

The stochastic shortest path algorithm is widely used in route planning when there is uncertainty in the model. Unfortunately, however, ensemble NWP output values are highly correlated within ensemble members while randomized but independent between members. Furthermore, the randomized error between ensemble members is nonparametric. These features combine to make creating a stochastic network difficult.

Chapter 2 reviews current methodologies for solving the discrete and probabilistic shortest path problems. In Chapter 3, a new methodology is presented that identifies the shortest path across a stochastic network with correlated random arcs which addresses some limitations of current methodologies. In Chapter 4, this new approach is applied toward the optimal routing of AMC aircraft with respect to minimizing fuel usage and compared to current practices. Finally, Chapter 5 concludes with key insights gained from this research and propose efforts to further this research, as well as additional applications of this new methodology.

II. Literature Review

2.1 Overview

This chapter discusses the stochastic Shortest Path Problem (SPP) and three current methodologies for determining the shortest path: the expected shortest path, the most shortest path, and the α-shortest path. The goal of the classic (deterministic) SPP is to find the quickest, cheapest, and/or the most reliable route between two points [1]. These problems are very common when dealing with transportation, routing, and communication networks. However, the discrete SPP is not always the most realistic, particularly when the arc lengths are uncertain. For instance, the optimal routing of an aircraft between two points will be highly affected by winds; predictions of which are highly probabilistic in ensemble NWP models. In situations such as these where there is significant uncertainty in the network, the classic SPP is far from sufficient [48]. The robust, or stochastic, SPP varies from its classic counterpart wherein the length of each arc is associated with a probability distribution [6]. Several different models have been proposed when solving this type of problem. Three of these models are the expected shortest path, the most shortest path, and the α-shortest path. For comparison, the Linear Programming (LP) formulation for the deterministic SPP is shown in (1).

2.2 Stochastic SPP

The three stochastic shortest path models discussed below are variations of their deterministic counterpart. The objective function varies with each model, but the constraints from the deterministic model remain constant in each variation, as does its notations. A is the set of all arcs (i, j) in the network and x_{ij} defines the arc from node i to node j, where $1 \leq i, j \leq n$. If the arc (i, j) is in the path, then x_{ij}=1 and

0 otherwise. c_{ij} is the cost of traversing, or the length of, the arc (i, j).

$$\begin{cases} \min \ \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{subject to :} \\ \quad \sum_{(1,j)\in A} x_{1j} - \sum_{(j,1)\in A} x_{j1} = 1, \\ \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = 0, \ 2 \leq i \leq n-1, \\ \quad \sum_{(n,j)\in A} x_{nj} - \sum_{(j,n)\in A} x_{jn} = -1, \\ \quad x_{ij} \in 0, 1, \ \forall (i,j) \in A. \end{cases} \tag{1}$$

Expected Shortest Path

The expected shortest path finds the path with the shortest expected length between two nodes; that is, the path that is shortest on average [43]. Murthy and Sarkar [34] found that finding the expected shortest path reduces to the discrete SPP where the arc costs are replaced by their expected values. There has been extensive research in the development of formulas to calculate these solutions. Davis and Prieditis [11] developed a closed-form approximation, building upon the recursive method developed by Kulkarni [28]. They determined the expected shortest path when arcs are independent and exponentially distributed. Davis and Prieditis [11] also found that their same formula gives a close approximation when the arcs are uniformly distributed. Ji [24] identified a general linear formulation for identifying the expected shortest path as (2).

The only difference between the expected shortest path formulation (2) and the deterministic case (1), is the objective function. Instead of minimizing the cost to get from the source node to the terminus node, the goal is to minimize the expected cost and identify the path that is shortest on average. $E\Big[\sum_{(i,j)\in A} \xi_{ij}x_{ij}\Big]$ is the expected shortest path and ξ_{ij} is the arc length, with the associated probability distribution,

from nodes 1 to n.

$$\begin{cases} \min\ E\Big[\sum_{(i,j)\in A}\xi_{ij}x_{ij}\Big] \\ \text{subject to :} \\ \qquad \sum_{(1,j)\in A}x_{1j}-\sum_{(j,1)\in A}x_{j1}=1, \\ \qquad \sum_{(i,j)\in A}x_{ij}-\sum_{(j,i)\in A}x_{ji}=0,\ 2\le i\le n-1, \\ \qquad \sum_{(n,j)\in A}x_{nj}-\sum_{(j,n)\in A}x_{jn}=-1, \\ \qquad x_{ij}\in 0,1,\ \forall(i,j)\in A. \end{cases} \tag{2}$$

The Most Shortest Path

The most shortest path model determines the path that has the highest probability of being faster than some requirement T_0 [24]. Using the dependent-chance programming (DCP) concepts outlined by Liu [29], Ji [24] developed the following DCP model for the most shortest path shown in (3). The most shortest path formulation contains the same constraints as the deterministic model, again the only variability is in the objective function.

$$\begin{cases} \max\ Pr\Big\{\sum_{(i,j)\in A}\xi_{ij}x_{ij}\le T_0\Big\} \\ \text{subject to :} \\ \qquad \sum_{(1,j)\in A}x_{1j}-\sum_{(j,1)\in A}x_{j1}=1, \\ \qquad \sum_{(i,j)\in A}x_{ij}-\sum_{(j,i)\in A}x_{ji}=0,\ 2\le i\le n-1, \\ \qquad \sum_{(n,j)\in A}x_{nj}-\sum_{(j,n)\in A}x_{jn}=-1, \\ \qquad x_{ij}\in 0,1,\ \forall(i,j)\in A. \end{cases} \tag{3}$$

The α-Shortest Path

The α-shortest path identifies the path that minimizes some time constraint $\bar{T}$ with a confidence level of at least α [24]. Leveraging the chance-constrained programming (CCP) concepts developed by Charnes and Cooper [10] and Liu [30], Ji [24] developed a model for determining the α-shortest path shown in (4). The deterministic formulation in (1) is augmented with an additional constraint and new objective function.

$$\begin{cases} \min \bar{T} \\ \text{subject to :} \\ \quad Pr\Big\{ \sum_{(i,j)\in A} \xi_{ij} x_{ij} \leq \bar{T} \Big\} \geq \alpha, \\ \quad \sum_{(1,j)\in A} X_{1j} - \sum_{(j,1)\in A} X_{j1} = 1, \\ \quad \sum_{(i,j)\in A} X_{ij} - \sum_{(j,i)\in A} X_{ji} = 0,\ 2 \leq i \leq n-1, \\ \quad \sum_{(n,j)\in A} X_{nj} - \sum_{(j,n)\in A} X_{jn} = -1, \\ \quad X_{ij} \in 0, 1,\ \forall (i,j) \in A. \end{cases} \tag{4}$$

These and many other methodologies assume arc lengths to be independently distributed to simplify models and reduce computational complexity [22]. While this assumption may be necessary to simplify a problem and its computational complexity, it is extremely limiting to ignore such a strong characteristic of the network and dampens the strength of the result. For this reason, newer methodologies have been introduced that account for arc correlation.

Correlated Arcs

There are many instances where arc lengths are not only uncertain, but they are correlated. For example, groups of nodes or links in a specific region of the net-

work may be correlated and, in turn, adjacent links and nodes are also affected [15]. Therefore, when determining the shortest path, prior choices will inform the decision making for the duration of the path construction, *i.e.* identifying the shortest path. Fan *et al.* [15] outline a formulation for solving the shortest path problem with correlated arcs. The arcs are considered congested or not congested, affected or not affected. Conditional probabilities are associated with each arc; that is, the probability that node i is affected given that node $i-1$ is affected. Fan *et al.* [15] present two formulations to identify the expected shortest path between two nodes depending on the type of network: node-based or link-based congestion. These networks are described by where the congestion may occur, at the nodes as in the node-based approach or along the arcs as in the link-based approach.

Eq. (5) is the formulation when node-based congestion is present. Where u_{ij} = the lowest expected travel time from *uncongested* node i to node j and v_{ij} = the lowest expected travel time from *congested* node i to j, where $i = 1, 2, ..., N-1$ and $j = 2, 3, ..., N$. The α_{ij} is the probability that if node i is uncongested, then node j is uncongested and β_{ij} is the probability that if node i is congested then j is congested. The t_{ij} and τ_{ij} are the expected arc lengths from (i, j) under uncongested and congested conditions respectively.

$$\begin{aligned} u_i &= \min_{j \neq i} \{t_{ij} + \alpha_{ij} u_j + (1 - \alpha_{ij}) v_j\}, \quad i = 1, 2, ..., N-1 \\ v_i &= \min_{j \neq i} \{\tau_{ij} + \beta_{ij} v_j + (1 - \beta_{ij}) u_j\}, \quad i = 1, 2, ..., N-1 \\ u_N, v_N &= 0 \end{aligned} \tag{5}$$

Eq. (6) shows the formulation when link-based congestion is present. The same notation from the node-based formulation also apply here. However, there are additional variables that need to be defined. That is,

$$\lambda_{ij} = 1 - \beta_{ij},$$

$p_{ij}(\tau)d\tau$ = the probability that traveling (i,j) requires time between τ and $\tau + d\tau$ given that the arc traversed to node i was *uncongested*, and

$q_{ij}(\tau)d\tau$ = the probability that traveling (i,j) requires time between τ and $\tau + d\tau$ given that the arc traversed to node i was *congested*.

$$\begin{aligned} u_i &= \min_{j \neq i}\{t_{ij} + \alpha_{ij}u_j + (1-\alpha_{ij})v_j\}, \quad i = 1, 2, ..., N-1 \\ v_i &= \min\{\tau_{ij} + \lambda_{ij}u_j + (1-\lambda_{ij})v_j\}, \quad i = 1, 2, ..., N-1 \\ u_N, v_N &= 0 \end{aligned} \tag{6}$$

where:

$$t_{ij} = \int_0^\infty \tau p_{ij}(\tau)d\tau \text{ and}$$

$$\tau_{ij} = \int_0^\infty \tau q_{ij}(\tau)d\tau.$$

While the formulations proposed address correlated arcs, they are contingent on conditional probabilities, which are difficult for the AMC problem. In addition, Fan *et al.* [15] requires assumptions about the distribution of probabilities across each arc which is not possible in the AMC problem.

The Mean Ensemble Model

Homan [20] compared the fuel burn estimates of a mean ensemble model to those of the deterministic approach currently in use today. Specifically, the study compared the fuel loads planned using a deterministic model forecast to those using three different ensemble mean forecasts across five aircraft and five pre-determined routes. The +00 hour forecast was used as the 'truth' source for a given date/time for the previous forecasts at the same date/time. The 'true' fuel burn was compared to the estimates

for the previous forecasts to calculate a fuel burn error. The results suggested that the use of ensemble means generally provided more accurate estimates.

While this approach might provide a better fuel point estimate than the deterministic approach currently in use, there is still a great deal of data that is not being utilized. By averaging the ensembles, data that could provide additional insight toward fuel planning is lost. More accurately, winds at point A and time 1 are correlated to winds at point B upstream at time 0. Correlation information within each ensemble member is lost, therefore averaging across the ensembles may not be the best approach.

2.3 Conclusion

While these examples are just a subset of the many ways to approach the stochastic SPP, many of these approaches only work when the arcs are independent. Of the formulations that allow for correlation, there are other limitations that do not fully address the AMC problem. A new methodology is outlined in the next chapter that determines the shortest path across a correlated random network without assuming independence of arc lengths or a distribution for the arc costs.

III. Methodology

3.1 Introduction

This chapter outlines the methodology for the *a posteriori* Shortest Path (APSP) approach proposed in this research and for the case study provided. The AMC application is described in detail to show how the data were gathered, how the network was constructed, and how this model differs from previous approaches. Lastly, information is provided showing which statistical techniques were applied to the results to gain further insight. Figure 1 provides an overview of this methodology.

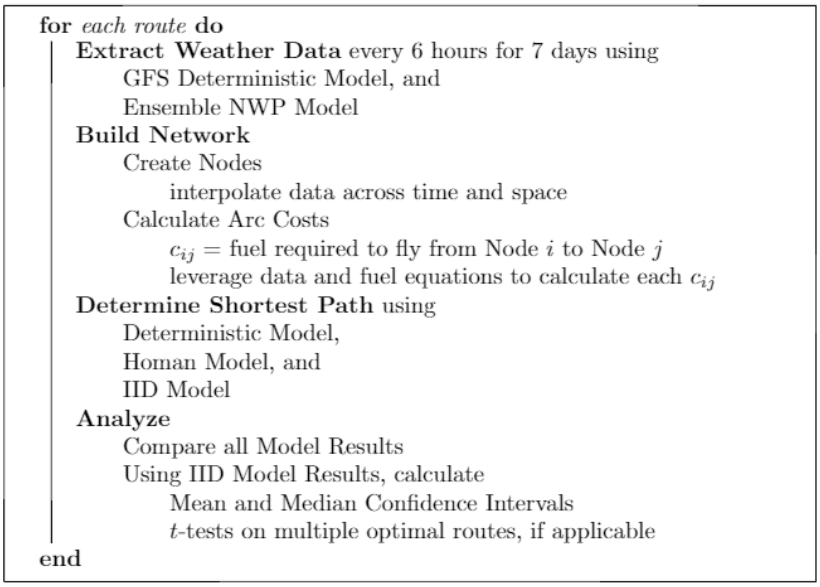

Figure 1. Outline of Methodology

Ensemble NWP

NWP leverages current weather observations and computer models to forecast future weather [38]. Current AMC models utilize NWP deterministic forecasts as opposed to the readily available ensemble forecasts. A deterministic forecast is a single member of an ensemble model initialized without random perturbations [16].

In other words, this type of model utilizes one forecast, whereas ensemble forecasts utilize multiple forecasts for weather prediction.

The *a posteriori* Shortest Path

Unlike current methodologies where the arcs are probabilistic, the stochastic nature of this network is analyzed after-the-fact. Using structural factoring, the complex network is broken down into k subnetworks [18]. Each subnetwork is then solved as the discrete SPP, formulation in (1), to obtain optimal solution(s) [6]. These k optimal solutions are then analyzed using nonparametric statistics, *i.e.* kernel smoothing, to obtain descriptive statistics and other relevant analyses on the k solutions.

3.2 AMC Routing Practices

AMC is the largest single consumer of fuel in the DoD. As such, their focus has shifted toward a more fuel-efficient culture and a great deal of work has been done to identify more fuel-efficient practices.

Previous Work

Mirtich [33] introduced the concept of Cost Index Flying (CIF). This is a program now used by commercial airlines to balance the cost of time and the cost of fuel. The USAF has since adopted this program and renamed it Mission Index Flying (MIF). Weather data is leveraged when determining aircraft routing, and in the last few years, research has improved the current routing practices. Homan [20] compared ensemble mean and deterministic forecasts for route planning. Homan's results suggested that ensemble mean forecasts outperform deterministic forecasts. That is, ensemble mean forecasts provide more accurate fuel burn estimates which could result in less reserve fuel being carried. However, unlike Mirtich, Homan's research has yet to be adopted.

Current Methodology

This new APSP methodology arose in an effort to provide better fuel estimates to AMC. AMC currently utilizes the Advanced Computer Flight Planner (ACFP) system to route cargo and ensure aerial refueling operations [40]. This is done by optimizing routes with respect to fuel consumption, subject to aircraft performance with wind and temperatures aloft and air traffic control and diplomatic constraints [20]. The 557th Weather Wing (557th WW) provides ACFP with the weather data necessary for this optimization scheme.

Weather data are extracted from a single forecast, one-degree NWP model at six-hour increments, from six to 96 hours, for each waypoint along the route [3]. These data consist of wind data for each latitude/longitude pair at each of the 4 atmospheric pressure levels [39]. Note that additional weather data are available but were not used in this analysis.

Wind data are presented with U- and V- components in meters per second (m/s). These components are the East/West and North/South components of the wind, respectively. Positive U-component indicate that the wind is traveling West to East. Positive V-component indicates that the wind is traveling from South to North. These components are utilized to determine the wind speed, direction, and angle.

Atmospheric Pressure Levels are provided in millibars. These pressure levels, when combined with temperature at a given point, translate to the altitude above mean sea level (MSL).

At the core of ACFP is the Worldwide Aeronautical Flight Planner (WARP). WARP serves to leverage advanced search techniques to produce routes that minimize fuel burn [40]. According to the AMC Director of Weather (AMC/A3W), routes are broken down into segments, legs, and sublegs within WARP (Fig. 2) [3]. Segments lie between two points. That is, if a route has four points, then that route would have

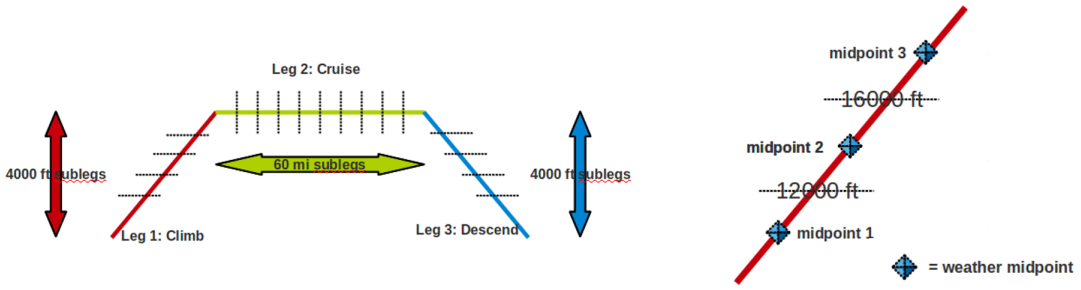

Figure 2. Segments, Legs, and Sublegs Example [3]

three segments. Legs lie between two navigational points along the route. Legs are then divided into sublegs inside WARP. If the length of a cruise leg is greater than 60 miles, then WARP divides the leg into sublegs such that all sublegs are less than or equal to 60 miles. During climb, sublegs are divided into 4000-foot (ft) increments. Weather at each subleg is determined at the midpoint of that subleg and the average of these subleg midpoints within each leg is reported as the weather for that leg. [3]

To get the weather data for a specific point (latitude, longitude, altitude, and time), WARP interpolates the weather data from nearby, known points, specifically from lower altitude and earlier time to higher altitude and later time. For example, if a specific point is 75% through a time slot, then the temperature and winds will be representative of 75% of the change in the temperature or wind, respectively.

3.3 The AMC Application

Data Gathering

Two MATLAB scripts were written to extract NWP data directly from National Oceanic and Atmospheric Administration (NOAA) using *nctoolbox* developed by Schlining *et al.* [44] [21]. Wind data, temperature, pressure levels, and model step times are provided at each latitudinal and longitudinal coordinate. Given the latitudinal and longitudinal coordinates, the U- and V- wind components are extracted in

6-hr increments across 20 ensemble members. Given the latitudinal and longitudinal coordinates, the U- and V- wind components are extracted in 3-hr increments across 1 deterministic forecast.

Building the Network

To investigate the problem at hand, three models were developed using different NWP data sets: a deterministic model, IID ensemble model, and Homan (mean ensemble) model. Each of these shortest path models are identical, the only difference being how the initial weather conditions are input into the model.

The weather data for the deterministic model are provided with one forecast in 3-hr increments with 1-degree resolution. These weather data from NOAA are extracted for three pressure altitudes converted to altitude MSL in standard atmosphere: 25K, 30K, and 35K feet. Operators will not change altitude in 5K-ft increments, therefore linear interpolation over time and space is used to calculate weather data in 1K-feet increments between 25K and 35K feet and hourly increments of time.

The weather data for the IID and Homan models are provided with 20 ensemble members in 6-hr increments with 1-degree resolution. Therefore, the network is separated into 20 subnetworks, where subnetwork i leverages the weather data at ensemble i. These weather data from NOAA are extracted at the same three pressure levels as in the deterministic forecast and interpolated in the same manner. To validate model comparisons, described in detail later in Section 3.3, the weather data for all models begins with the +06 hour forecast.

Each subnetwork consists of $11n + 1$ nodes (i) and $11(11n - 20)$ arcs (j), where n is the number of equidistant legs along the great circle route from the source node s to the terminus node t. The weather data are provided from a 1 degree Global Forecast System (GFS) model, therefore a weighted average based on the location of

actual node with respect to the nearest 1 degree nodes was applied to determine the weather data at each node.

Ng *et al.* [37] found that travel time and fuel savings for initial climb and final descent are negligible when compared to those during cruise. For this reason, the take-off and landing portions of flight are ignored; that is, node 1 and node $11n + 1$ are forced to be 25k feet.

According to the C-17 Fact Sheet, the average cruising speed is 450 nautical miles per hour (knots) [12]. Therefore, this research assumes that the C-17 maintains a constant airspeed of 450 knots during cruise. To calculate the effect that winds have on fuel efficiency, that is, the headwind, tailwind, and crosswind, the wind speed and direction and the aircraft heading is first calculated. Weather data are provided in U- and V- components in m/s, therefore the wind speed (m/s) was calculated using the Pythagorean Theorem as (7).

$$WS = \sqrt{U^2 + V^2} \tag{7}$$

The wind direction was calculated using MATLAB's *atan2d* function, (8), to calculate the wind direction in degrees. MATLAB's *atan2d(y,x)* function returns the four-quadrant arctangent of y/x [31].

$$\text{Wind Direction} = atan2d(-U, -V) \tag{8}$$

The aircraft heading from A to B, without any wind effects, was calculated using the *atan2d* function as (9) [47].

$$\text{AC heading}_{i-1} = mod(atan2d(Y, X), 360) \tag{9}$$

where:

$X = cos\theta_A sin\theta_B - sin\theta_A cos\theta_B cos\Delta_L$,
$Y = cos\theta_B sin\Delta_L$,
$L =$ Longitude, and
$\theta =$ Latitude.

The aircraft corrected heading, due to winds, was calculated using MATLAB's *driftcorr* function which takes AC_heading, True airspeed (TAS), Wind_Dir, and wind speed (WS) as inputs and returns the aircraft's corrected heading (AC_heading_corr), ground speed (in knots), and the correction angle (in degrees) due to winds. The distance between neighboring nodes were small enough to be assumed linear; therefore Pythagorean Theorem was again used to determine the straight-line distance between nodes.

Reiman [42] developed regression models on flight data from performance manuals to estimate fuel consumption for the C-17, C-130, and C-5. These models were utilized to determine the path that required the least amount of fuel to traverse. These models were broken down into climb, cruise, and descent. For the purposes of this problem, only the C-17 data are provided in the tables.

The regression model for calculating the fuel and distance required to climb in (10) and their respective βs are shown in Table 1. The descent model for calculating the the fuel and distance required for descent in (11) with the βs in Table 2.

$$\phi_C = \beta_0 + \beta_1\alpha + \beta_2\alpha^2 + \beta_3\alpha^3 + \beta_4\omega + \beta_5\omega^2 + \beta_6\omega^3 + 10^{-6}\beta_7\alpha^2\omega^3 + 10^{-6}\beta_8\alpha^2\omega^3 \quad (10)$$

$$\phi_D = \beta_0 + \beta_1\omega + \beta_2\omega^2 + \beta_3\alpha + \beta_4\alpha\omega \quad (11)$$

where:

ϕ_C = Fuel to Climb in Klbs or Distance to Climb in NMs
ϕ_D = Fuel to Descend in Klbs or Distance to Descend in NMs
α = Altitude in Thousands of Feet
ω = Aircraft Gross Weight in Klbs at Climb/Descent Start

Table 1. Climb ϕ_C Regression Terms [42]

	Fuel	Dist
β_0	-4.7054	-51.504
β_1	0.2869	2.0961
β_2	-0.0070	-0.0282
β_3	7.1E-05	0.0003
β_4	0.0267	0.3363
β_5	-5.9E-05	-0.0008
β_6	4.8E-08	6.9E-07
β_7	6.7E-05	0.0003
β_8	-2.1E-07	1.7E-05

Table 2. Descent ϕ_D Regression Terms [42]

	Fuel	Dist
β_0	0.2574	-16.382
β_1	0.0005	0.1278
β_2	-8.5E-7	-1.7E-4
β_3	0.0108	1.3919
β_4	3.2E-5	0.0036

Finally, the cruise model for calculating the fuel consumed during cruise in (12) with the respective β values in Table 3.

$$\omega_{ff} = -\frac{B}{3A} - \frac{1}{3A}\sqrt[3]{\frac{1}{2}[2B^3 - 9ABC + 27A^2D + \sqrt{(2B^3 - 9ABC + 27A^2D)^2 - 4(B^2 - 3AC)^3}]} - \frac{1}{3A}\sqrt[3]{\frac{1}{2}[2B^3 - 9ABC + 27A^2D - \sqrt{(2B^3 - 9ABC + 27A^2D)^2 - 4(B^2 - 3AC)^3}]} \quad (12)$$

where (all weights in Klbs):

$$A = \frac{\beta_4}{3}$$
$$B = (\frac{\beta_3}{2} + \beta_4(\omega_{op} + \omega_{frc} + \omega_{fah} + \omega_p) + \frac{\beta_5}{2}\alpha$$
$$C = \beta_0 + \beta_1\alpha + \beta_2\alpha^2 + \beta_3(\omega_{op} + \omega_{frc} + \omega_{fah} + \omega_p) + \beta_4(\omega_{op} + \omega_{frc} + \omega_{fah} + \omega_p)^2) + \beta_5\alpha(\omega_{op} + \omega_{frc} + \omega_{fah} + \omega_p)$$
$$D = -\delta$$
α = Altitude in Thousands of Feet

δ = Distance in NMs
ω = Aircraft Gross Weight
ω_{frc} = Reserve/Contingency Fuel Weight
ω_{op} = Operating Weight
ω_{fah} = Alternate/Holding Fuel Weight
ω_p = Payload Weight
ω_{ff} = Cruise Fuel Weight
f = Fuel Consumed
$= \omega_{op} + \omega_{frc} + \omega_{fah} + \omega_p + f$

Table 3. Cruise ϕ_D Regression Terms [42]

	Fuel
β_0	31.735
β_1	0.9897
β_2	-0.0043
β_3	-0.0642
β_4	5.8E-05
β_5	-0.0011

A binary integer programming model is formulated to determine the shortest path, with respect to fuel consumed, between two points. This model utilizes the great circle route between the two points, and optimizes the cruising altitude to minimize fuel consumption along that route. This model is the deterministic model in (1) where c_{ij} is calculated utilizing the regression models developed by Reiman [42], code is provided in Appendix B and C.

The Different Models

This subsection identifies the differences between the three models developed for this analysis. Each model utilizes the same network(s), however the weather data is implemented into each model differently. Only one approach, the IID Model, actually uses the new *a posteriori* methodology.

Each of the models are applied to four operationally relevant routes as identified by a subject matter expert, a C-17 Instructor Pilot [7]. The following routes are used:

KSUU-PHNL Travis Air Force Base (AFB), CA to Honolulu, HI

KTCM-CYQX McChord AFB, WA to Gander Newfoundland and Labrador (NL)

KTCM-KCHS McChord AFB, WA to Charleston AFB, SC

KRIV-KWRI March Air Reserve Base to Joint Base McGuire-Dix-Lakehurst, NJ

Deterministic Model

The purpose of the deterministic model was to replicate current AMC routing practices. AMC currently uses deterministic forecasts for aircraft routing, therefore a single forecast was used to determine the optimal path: NOAA's deterministic 1°GFS model forecast. These weather data were utilized to identify the route that would minimize fuel burn; see Appendix A for the MATLAB code for the deterministic model. This network does not utilize the *a posteriori* approach proposed because there is no uncertainty accounted for in the model. Therefore, no subnetworks are analyzed.

Homan Model

The Homan model is a recreation of the mean ensemble model introduced by Homan [20]. This model is similar to the deterministic model in that it does not utilize the *a posteriori* approach developed and reverts to the discrete SPP. However, unlike the deterministic model, the Homan model leverages the ensemble data. Instead of using one deterministic forecast, the average of all of the ensembles is input into the network for the mean model, code is provided in Appendix B. However, there are

issues with this model due to the inter-correlation between wind values within each ensemble member.

IID Model

The IID model leverages the *a posteriori* approach. Twenty subnetworks are developed, one for each ensemble. The optimal path for each subnetwork is then saved. With these 20 optimal paths, up to 20 unique routes are identified. Each unique route is then re-ran through each of the subnetworks again, code provided in Appendix C. This provided 20 estimates of fuel consumption for each unique route. Descriptive statistics were then applied to determine which routes were statistically significantly better (consume less fuel) than others across all ensembles and to develop confidence intervals on the fuel estimates.

Model Comparison

The accuracy of each of the three models is calculated using the root mean square error (RMSE) of the fuel burn estimates. The model estimates are compared to a *truth* value in order to calculate the fuel burn error. This *truth* is calculated using the Deterministic Model across the +00 hour forecasts. The +00 hour forecast for a specific date/time is the initialization of the deterministic model and is therefore the closest thing to the true conditions at each GFS model run time. This method of comparison was used by Homan [20] and is a common technique for NWP researchers [25]. The RMSE is calculated as (13) where N is the sample size, FB_{truth} is the true fuel burn, and FB_{est} is the estimated fuel burn for a given model. $N = 1$ for the Deterministic and Homan models and $N = 20$ for the IID model.

$$FB_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(FB_{est} - FB_{truth})_i^2} \tag{13}$$

Statistical Analysis Techniques

The aforementioned descriptive statistics allow a better characterization of fuel usage across flights. Through t-tests, any statistically significant differences between the optimal route for each subnetwork can be identified. This will identify the overall optimal route(s) and, if multiple routes are identified, potentially provide the user with route options that consume statistically equivalent amounts of fuel.

Using the estimated fuel consumption for each ensemble will also result in fuel usage confidence intervals. These intervals provide the user with more fidelity when calculating the amount of fuel required for a mission. The ensemble data can provide the user with the $(1-\alpha)\%$ confidence interval around the mean fuel usage. This is shown in (14) where μ is the true mean, $\bar{x}$ is the sample mean, n is the number of ensembles (20), and σ is the sample standard deviation.

$$\mu \in \bar{x} \pm t_{1-\frac{\alpha}{2},n-1}\frac{\sigma}{\sqrt{n}} \tag{14}$$

Confidence intervals around the median fuel usage can also be calculated by determining the values of j and k such that $P(X_{(j)} \leq x_p \leq X_{(k)}) = 1-\alpha$ after sorting the data from smallest to largest [9]. This is shown in (15) where n is the number of ensembles (20) and q is the proportion (0.5). Therefore, the 95% confidence interval of the median when $n = 20$ is always between X_6 and X_{15}.

$$\begin{aligned} j &= \lceil nq - t_{1-\frac{\alpha}{2},n-1}\sqrt{nq(1-q)}\rceil \\ k &= \lfloor nq + t_{1-\frac{\alpha}{2},n-1}\sqrt{nq(1-q)}\rfloor \end{aligned} \tag{15}$$

Comparing the different models and the unique routes identified in the IID model is done using t-tests, more specifically the paired t-test. Because fuel estimates depend on the specific forecasts used and are not independent, the paired t-test is the most appropriate test for detecting statistically significant differences in the fuel estimates.

The paired t-test tests the null hypothesis, $H_0 : \mu_2 - \mu_1 = d_0$, versus the alternative, $H_1 : \mu_2 - \mu_1 \neq d_0$, where d_0 is the difference to detect and μ_i is the true mean of group i. For this study, $d_0 = 0$ because the goal of the test is to identify if there is a difference between the true means. Let $\bar{d}$ and s_d be the sample mean and standard deviation of the differences, respectively, and n be the number of observations, then the critical value, t_0 is calculated as (16) [5]. The statistic, t_0, is then tested against the test statistic to determine if there is a statistically significant difference between the means. If $|t_0| \geq t_{1-\alpha/2,n-1}$, then there is enough evidence to identify a statistically significant difference between the means with
$(1 - \alpha)\%$ confidence.

$$t_0 = \frac{\bar{d} - d_0}{s_d/\sqrt{n}} \tag{16}$$

IV. Analysis

4.1 Introduction

For the deterministic and mean models, only one route and fuel point estimate is calculated. However, with the IID model, 20 different estimates of fuel usage and at least one route is identified. To show the differences between the models, seven consecutive days of weather data, ranging from 28 January to 5 February 2018, were extracted for each route of interest every six hours. This resulted in 27 deterministic and ensemble weather forecasts for each of the routes.

4.2 KSUU-PHNL

The RMSE of fuel burn estimates for the KSUU-PHNL route for the IID, Deterministic, and Homan models are shown in Figure 3. The RMSE for the Homan and IID models are nearly identical, whereas the RMSE using the Deterministic model is much larger in all but two cases.

The Homan model point estimates are well contained in both mean and median confidence intervals generated by the IID model, as seen in Figure 4. However, the fuel estimate yielded by the deterministic model is outside of the 95% mean confidence interval 27/27 times. When outside the confidence bounds, the deterministic model under- or over-estimates between 21.85 and 4,414.44 pounds of fuel at each time interval. The deterministic model under-estimated and over-estimated 27 times. Figure 4 shows the offsets between the current deterministic model with all other models, including the 95% mean and median confidence intervals calculated using the IID model.

The large spread between the deterministic model and Homan model is verified by the paired t-tests. These tests show that there is a statistically significant difference

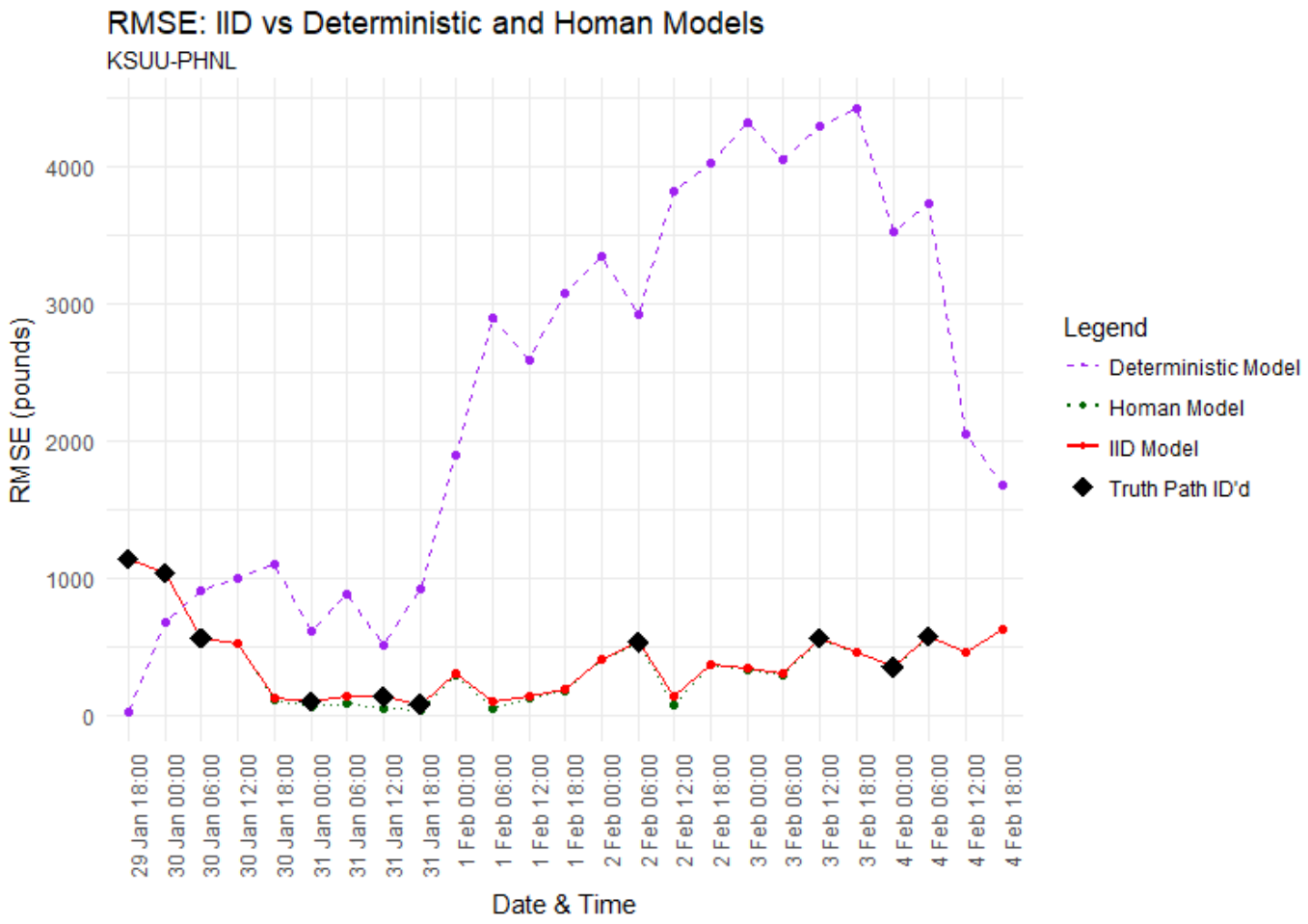

Figure 3. KSUU-PHNL: RMSE for IID, Deterministic, and Homan Models

between the deterministic and Homan models, with a p-value of 3.69×10^{-8}.

Aside from the ability to generate confidence intervals around the mean and median, there is another advantage to the IID model. Of the 27 timesteps, 13 found multiple routes across all ensembles. These timesteps and their results are shown in Table 4. In 11 scenarios, only two unique routes were identified, and three routes were identified in the other two scenarios. 13 of the 17 total route comparisons performed identified a statistically significant difference between the means. So, depending on the day and time, multiple alternative routes could be suggested that will not use statistically significant more fuel.

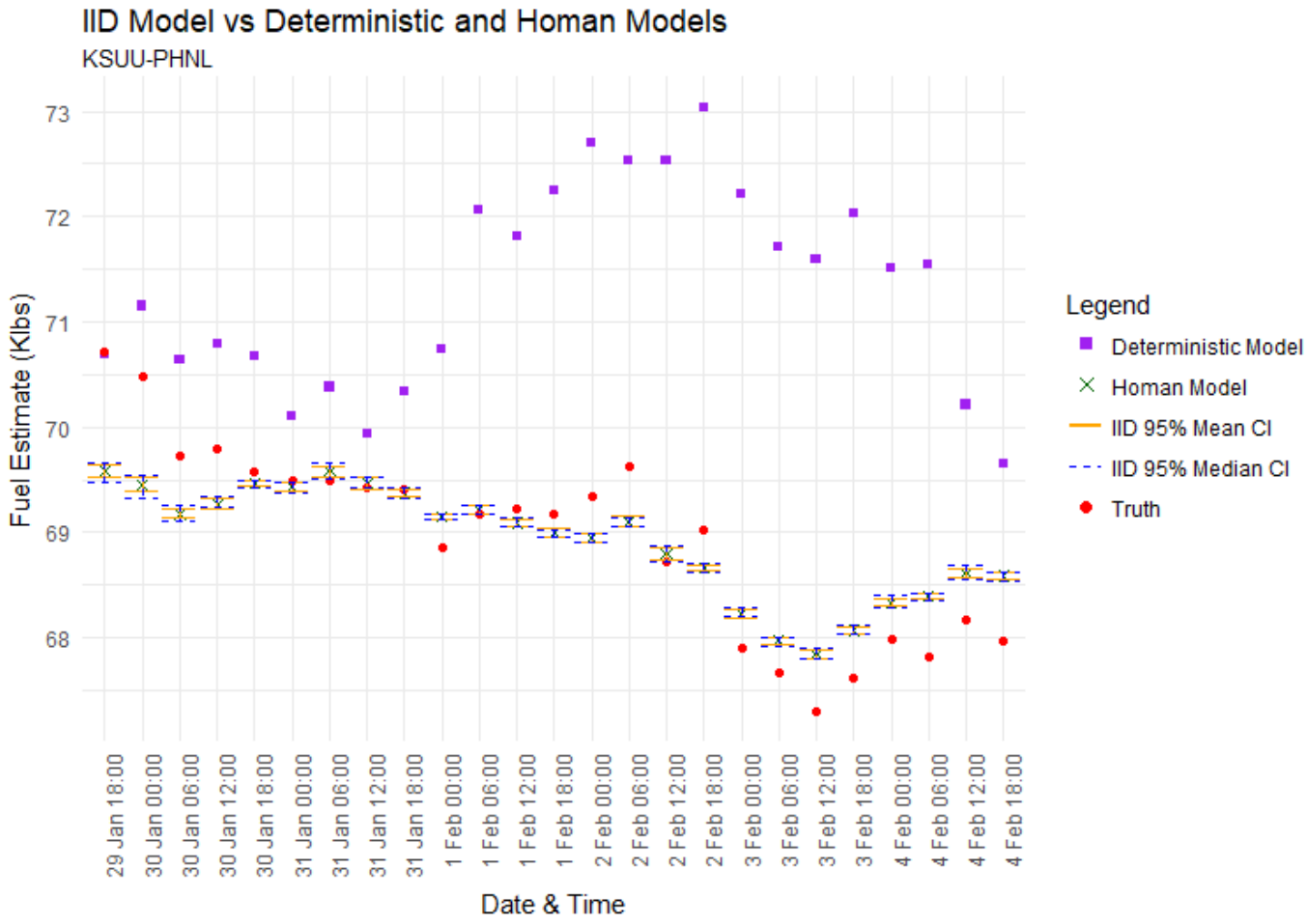

Figure 4. KSUU-PHNL: All model comparisons

For each date/time, the optimal routes identified by each model were compared to the optimal path of the *true* forecast. In 10 of 25 comparisons, the IID model identified the *true* optimal path (Figure 5). The Deterministic and Homan models did not identify the *true* path in any of the scenarios inspected. Figure 3 shows which date/times each model identified the *true* optimal path.

4.3 KTCM-CYQX

The RMSE results for the KTCM-CYQX route for the IID, Deterministic, and Homan models are shown in Figure 6. The RMSE for the Homan and IID models are

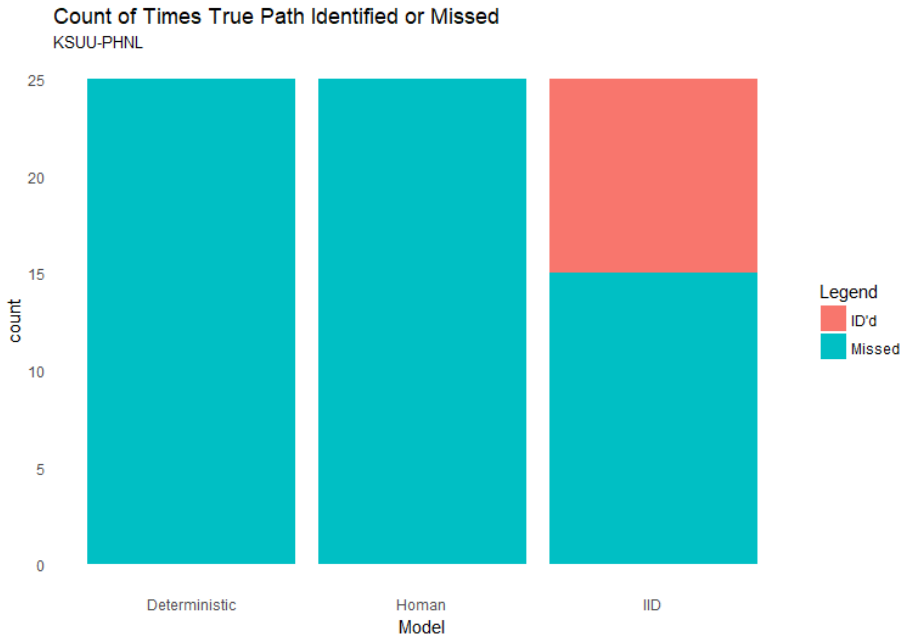

Figure 5. KSUU-PHNL: True Path Comparisons

Table 4. KSUU-PHNL: Unique Route Comparisons

Date & Time	Number of Routes	Comparison	p-value	Result
31 Jan 0600	3	$\mu_1 = \mu_2$	2.02×10^{-6}	$\neq$
		$\mu_1 = \mu_3$	2.1×10^{-6}	$\neq$
31 Jan 1200		$\mu_2 = \mu_3$	5.67×10^{-6}	$\neq$
	3	$\mu_1 = \mu_2$	0.038	$\neq$
		$\mu_1 = \mu_3$	6.72×10^{-4}	$\neq$
		$\mu_2 = \mu_3$	3.85×10^{-8}	$\neq$
31 Jan 1800	2	$\mu_1 = \mu_2$	0.151	$=$
2 Feb 0600	2	$\mu_1 = \mu_2$	5.00×10^{-4}	$\neq$
2 Feb 1200	2	$\mu_1 = \mu_2$	0.052	$=$
2 Feb 1800	2	$\mu_1 = \mu_2$	0.117	$=$
3 Feb 1200	2	$\mu_1 = \mu_2$	1.45×10^{-10}	$\neq$
3 Feb 1800	2	$\mu_1 = \mu_2$	0.021	$\neq$
4 Feb 0000	2	$\mu_1 = \mu_2$	2.67×10^{-4}	$\neq$
4 Feb 0600	2	$\mu_1 = \mu_2$	0.140	$=$
4 Feb 1800	2	$\mu_1 = \mu_2$	4.03×10^{-6}	$\neq$
5 Feb 0000	2	$\mu_1 = \mu_2$	0.015	$\neq$
5 Feb 0600	2	$\mu_1 = \mu_2$	6.91×10^{-8}	$\neq$

nearly identical, whereas the RMSE using the Deterministic model is much larger in all but three cases.

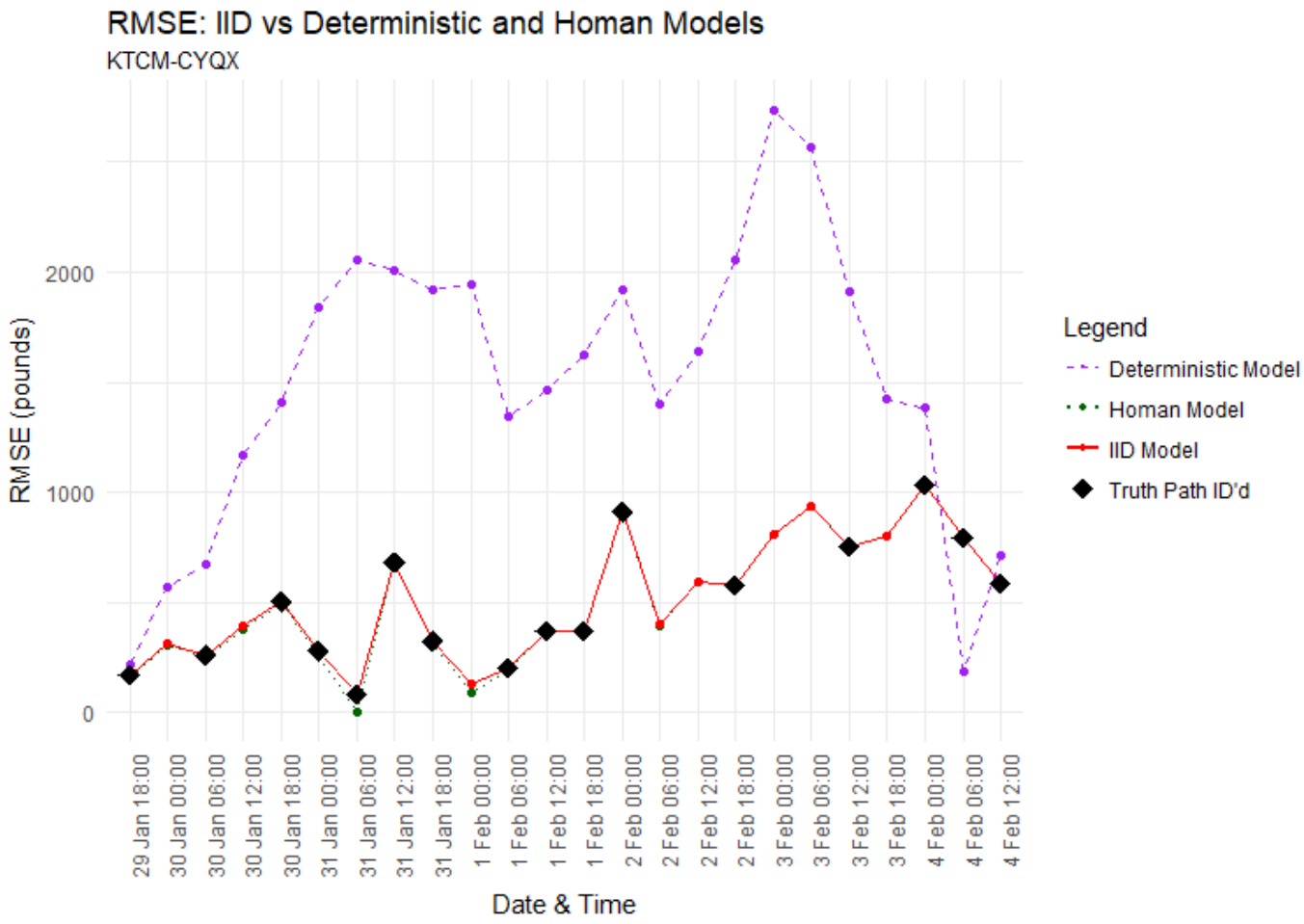

Figure 6. KTCM-CYQX: RMSE for IID, Deterministic, and Homan Models

The Homan model point estimates are well contained in the mean and median confidence intervals generated by the IID model, as seen in Figure 7. However, the fuel estimate yielded by the deterministic model is outside of the 95% mean confidence interval 24/24 times. When outside the confidence bounds, the deterministic model under- or over-estimates between 186.25 and 2,731.89 pounds of fuel at each time interval. The deterministic model under-estimated and over-estimated 24 times. Figure 7 shows the offsets between the current deterministic model with all other

models, including the 95% mean and median confidence intervals calculated using the IID model.

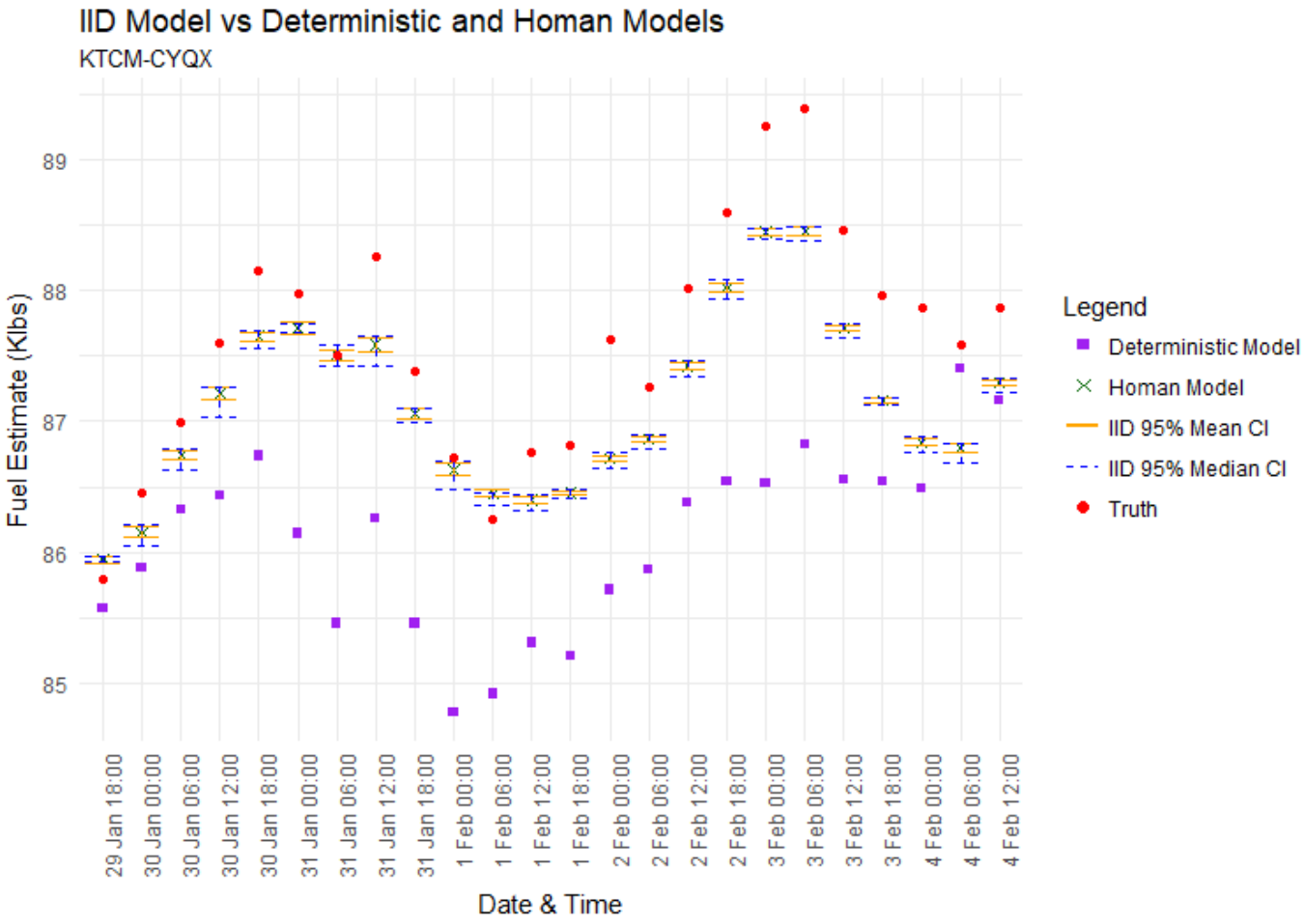

Figure 7. KTCM-CYQX: All model comparisons

The large spread between the deterministic model and Homan model is verified by the paired t-tests. These tests show that there is a statistically significant difference between the deterministic and Homan models, with a p-value of 8.01×10^{-9}.

Aside from the ability to generate confidence intervals around the mean and median, there is another advantage to the IID model. Of the 27 timesteps, 14 found multiple routes across all ensembles. These timesteps and their results are shown in Table 4. In six scenarios, only two unique routes were identified and the number of

routes identified in the other eight scenarios ranged from three to seven. A total of 29 of the 47 total route comparisons performed identified a statistically significant difference between the means. Depending on the day and time, multiple alternative routes could be suggested that will not use statistically significant more fuel.

For each date/time, the optimal routes identified by each model were compared to the optimal path of the *true* forecast. In 16 of 25 comparisons, the IID model identified the *true* optimal path (Figure 8). The Deterministic and Homan models did not identify the *true* path in any of the scenarios inspected. Figure 6 shows which date/times each model identified the *true* optimal path.

Figure 8. KTCM-CYQX: True Path Comparisons

Table 5. KTCM-CYQX: Unique Route Comparisons

Date & Time	Number of Routes	Comparison	p-value	Result
29 Jan 1800	4	$\mu_1 = \mu_2$	3.83×10^{-5}	$\neq$
		$\mu_1 = \mu_3$	1.84×10^{-9}	$\neq$
		$\mu_1 = \mu_4$	3.35×10^{-8}	$\neq$
		$\mu_2 = \mu_3$	7.99×10^{-10}	$\neq$
		$\mu_2 = \mu_4$	2.66×10^{-8}	$\neq$
		$\mu_3 = \mu_4$	3.36×10^{-7}	$\neq$

Table 5. (continued)

Date & Time	Number of Routes	Comparison	p-value	Result
30 Jan 0000	2	$\mu_1 = \mu_2$	2.87×10^{-7}	$\neq$
30 Jan 0600	2	$\mu_1 = \mu_2$	0.057	$=$
30 Jan 1200	3	$\mu_1 = \mu_2$	6.60×10^{-7}	$\neq$
		$\mu_1 = \mu_3$	2.11×10^{-6}	$\neq$
		$\mu_2 = \mu_3$	1.03×10^{-3}	$\neq$
30 Jan 1800	7	$\mu_1 = \mu_2$	5.65×10^{-2}	$=$
		$\mu_1 = \mu_3$	0.124	$=$
		$\mu_1 = \mu_4$	0.602	$=$
		$\mu_1 = \mu_5$	0.996	$=$
		$\mu_1 = \mu_6$	0.023	$\neq$
		$\mu_1 = \mu_7$	0.019	$\neq$
		$\mu_2 = \mu_3$	0.771	$=$
		$\mu_2 = \mu_4$	0.337	$=$
		$\mu_2 = \mu_5$	0.638	$=$
		$\mu_2 = \mu_6$	0.027	$\neq$
		$\mu_2 = \mu_7$	0.022	$\neq$
		$\mu_3 = \mu_4$	0.318	$=$
		$\mu_3 = \mu_5$	0.602	$=$
		$\mu_3 = \mu_6$	0.028	$\neq$
		$\mu_3 = \mu_7$	0.023	$\neq$
		$\mu_4 = \mu_5$	0.124	$=$
		$\mu_4 = \mu_6$	0.019	$\neq$
		$\mu_4 = \mu_7$	0.015	$\neq$
		$\mu_5 = \mu_6$	0.023	$\neq$
		$\mu_5 = \mu_7$	0.019	$\neq$
		$\mu_6 = \mu_7$	0.124	$=$
31 Jan 0000	3	$\mu_1 = \mu_2$	0.046	$\neq$
		$\mu_1 = \mu_3$	0.027	$\neq$
		$\mu_2 = \mu_3$	0.083	$=$
31 Jan 0600	2	$\mu_1 = \mu_2$	2.93×10^{-6}	$\neq$
31 Jan 1200	3	$\mu_1 = \mu_2$	8.37×10^{-5}	$\neq$
		$\mu_1 = \mu_3$	3.86×10^{-6}	$\neq$
		$\mu_2 = \mu_3$	3.37×10^{-6}	$\neq$
31 Jan 1800	2	$\mu_1 = \mu_2$	6.33×10^{-6}	$\neq$
1 Feb 0600	2	$\mu_1 = \mu_2$	0.527	$=$
1 Feb 1800	2	$\mu_1 = \mu_2$	1.10×10^{-3}	$\neq$
2 Feb 1800	3	$\mu_1 = \mu_2$	0.401	$=$
		$\mu_1 = \mu_3$	0.458	$=$
		$\mu_2 = \mu_3$	0.671	$=$
3 Feb 1200	2	$\mu_1 = \mu_2$	0.079	$=$
4 Feb 1200	2	$\mu_1 = \mu_2$	3.56×10^{-8}	$\neq$

4.4 KTCM-KCHS

The RMSE results for the KSUU-PHNL route for the IID, Deterministic, and Homan models are shown in Figure 9. The RMSE for the Homan and IID models are nearly identical, whereas the RMSE using the Deterministic model is much larger in all scenarios.

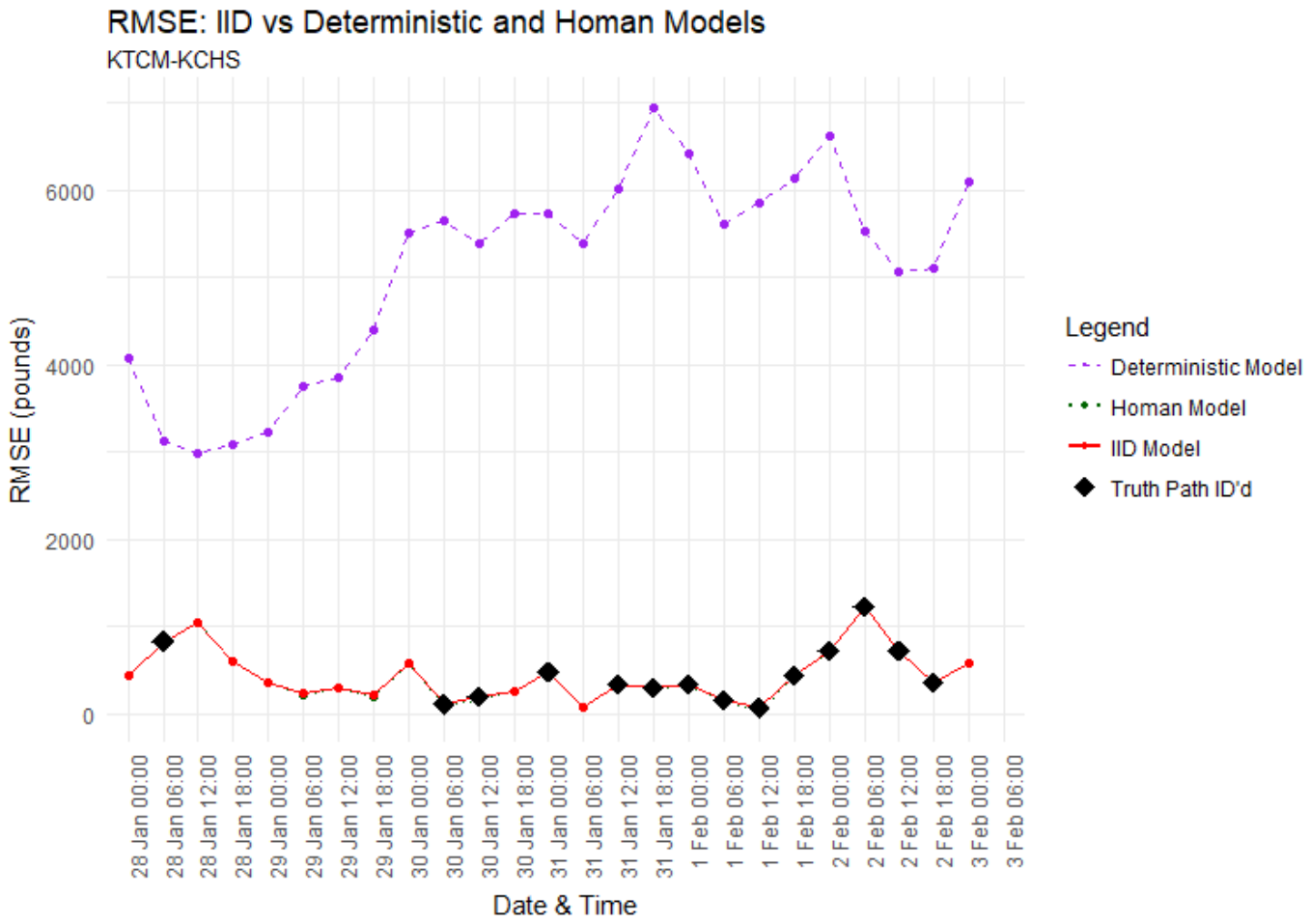

Figure 9. KTCM-KCHS: RMSE for IID, Deterministic, and Homan Models

The Homan model point estimates are well contained in both mean and median confidence intervals generated by the IID model, as seen in Figure 10. However, the fuel estimate yielded by the deterministic model is outside of the 95% mean confidence interval 25/25 times. The deterministic model over-estimates the *true* fuel estimate

between 2,981.81 and 6,940.05 pounds of fuel at each time interval. Figure 10 shows the offsets between the current deterministic model with all other models, including the 95% mean and median confidence intervals calculated using the IID model. The Homan model overlaps the mean confidence interval in every scenario.

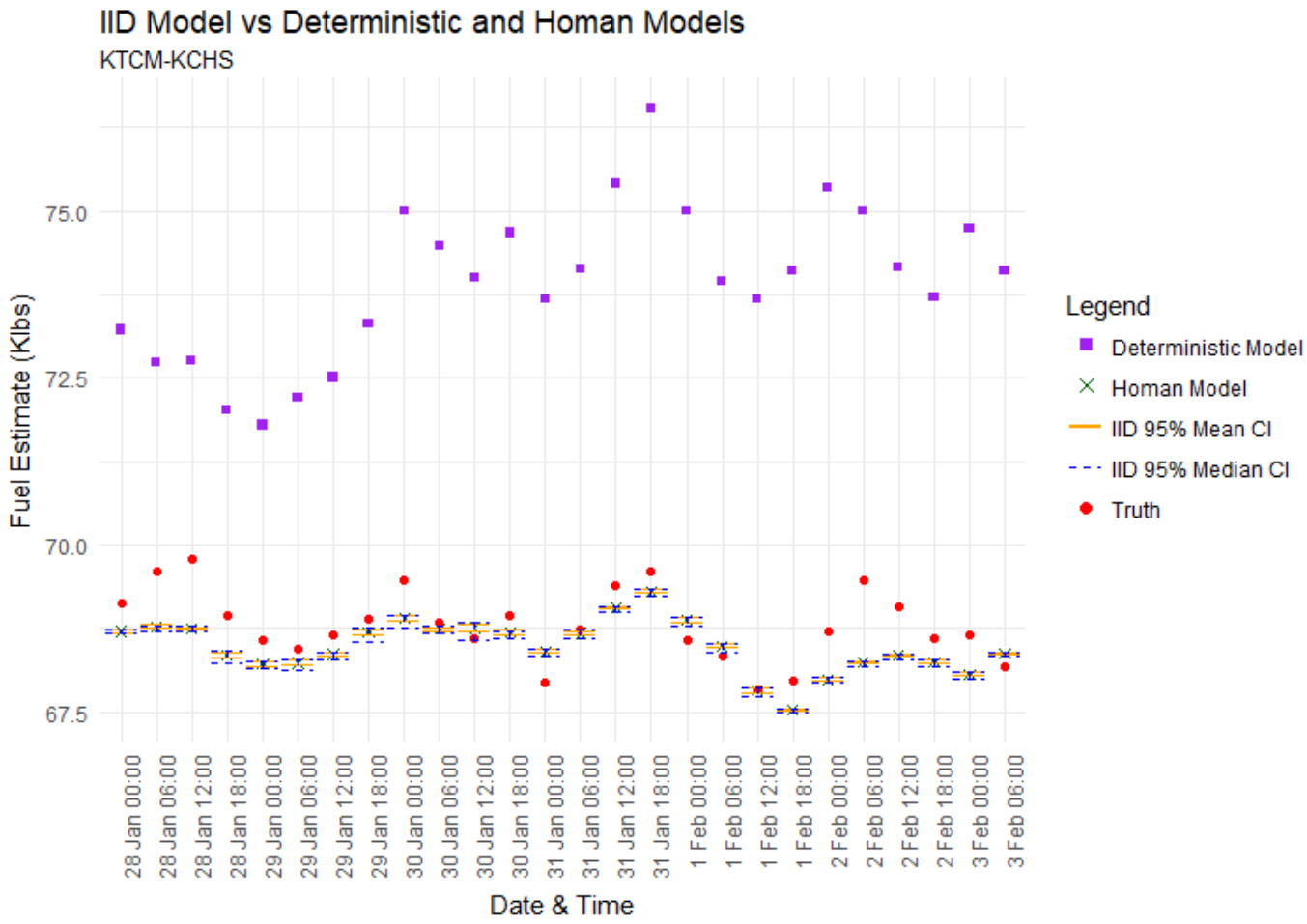

Figure 10. KTCM-KCHS: All model comparisons

The large spread between the deterministic model and Homan model is verified by the paired t-tests. These tests show that there is a statistically significant difference between the deterministic and Homan models, with a p-value of 5.01×10^{-21}.

Aside from the ability to generate confidence intervals around the mean and median, there is another advantage to the IID model. Of the 26 timesteps, 13 found

multiple routes across all ensembles. These timesteps and their results are shown in Table 6. In 11 scenarios, only two unique routes were identified, and three routes were identified in the other two scenarios. All 17 total route comparisons performed identified a statistically significant difference between the means. Depending on the day and time, multiple alternative routes could be suggested that will not use statistically significant more fuel.

For each date/time, the optimal routes identified by each model were compared to the optimal path of the *true* forecast. In 15 of 25 comparisons, the IID model identified the *true* optimal path (Figure 11). The Deterministic and Homan models did not identify the *true* path in any of the scenarios inspected. Figure 9 shows which date/times each model identified the *true* optimal path.

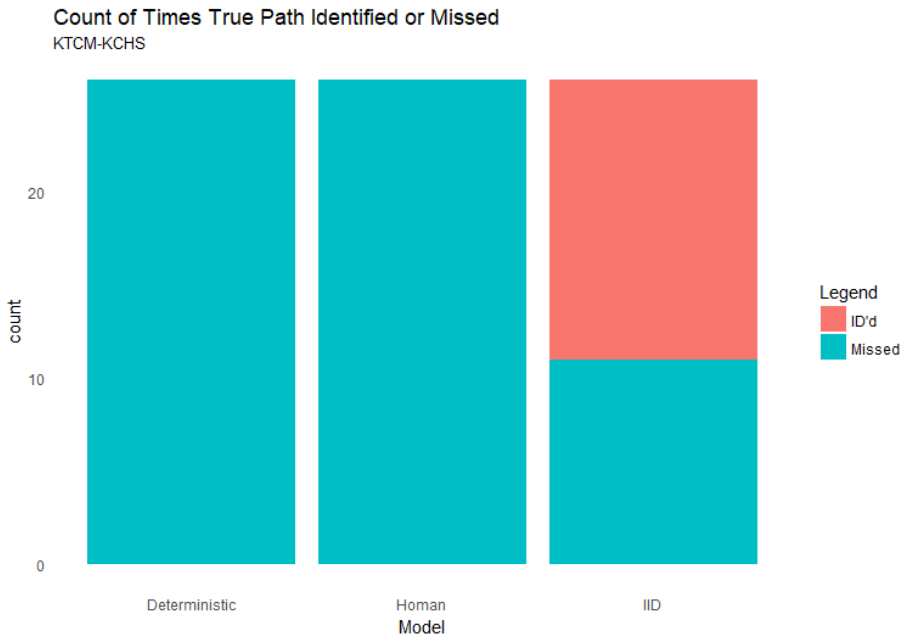

Figure 11. KTCM-KCHS: True Path Comparisons

Table 6. KTCM-KCHS: Unique Route Comparisons

Date & Time	Number of Routes	Comparison	P value	Result
28 Jan 0000	3	$\mu_1 = \mu_2$	2.36×10^{-6}	$\neq$
		$\mu_1 = \mu_3$	1.10×10^{-5}	$\neq$
		$\mu_2 = \mu_3$	2.79×10^{-3}	$\neq$
28 Jan 0600	2	$\mu_1 = \mu_2$	7.27×10^{-7}	$\neq$
28 Jan 1200	2	$\mu_1 = \mu_2$	2.48×10^{-3}	$\neq$
28 Jan 1800	3	$\mu_1 = \mu_2$	4.46×10^{-3}	$\neq$
		$\mu_1 = \mu_3$	2.70×10^{-5}	$\neq$
		$\mu_2 = \mu_3$	7.78×10^{-7}	$\neq$
29 Jan 0000	2	$\mu_1 = \mu_2$	1.89×10^{-4}	$\neq$
29 Jan 0600	2	$\mu_1 = \mu_2$	1.03×10^{-2}	$\neq$
29 Jan 1200	2	$\mu_1 = \mu_2$	1.17×10^{-2}	$\neq$
29 Jan 1800	2	$\mu_1 = \mu_2$	1.59×10^{-7}	$\neq$
30 Jan 0600	2	$\mu_1 = \mu_2$	5.50×10^{-8}	$\neq$
31 Jan 0000	2	$\mu_1 = \mu_2$	5.08×10^{-5}	$\neq$
31 Jan 1200	2	$\mu_1 = \mu_2$	1.11×10^{-4}	$\neq$
1 Feb 0000	2	$\mu_1 = \mu_2$	6.45×10^{-6}	$\neq$
3 Feb 0600	2	$\mu_1 = \mu_2$	1.52×10^{-5}	$\neq$

4.5 KRIV-KWRI

The RMSE for the KRIV-KWRI route for the IID, Deterministic, and Homan models are shown in Figure 12. The RMSE for the Homan and IID models are nearly identical, and the RMSE of the Deterministic model varies with the other models.

The Homan model point estimates are well contained in both mean and median confidence intervals generated by the IID model, as seen in Figure 13. However, the fuel estimate yielded by the deterministic model is outside of the 95% mean confidence interval 24 of 26 times. The deterministic model under-estimates the *true* fuel burn between 235 and 1,707.09 pounds of fuel at each time interval. The deterministic model under-estimated 27 times. Figure 13 shows the offsets between the current deterministic model with all other models, including the 95% mean and median confidence intervals calculated using the IID model.

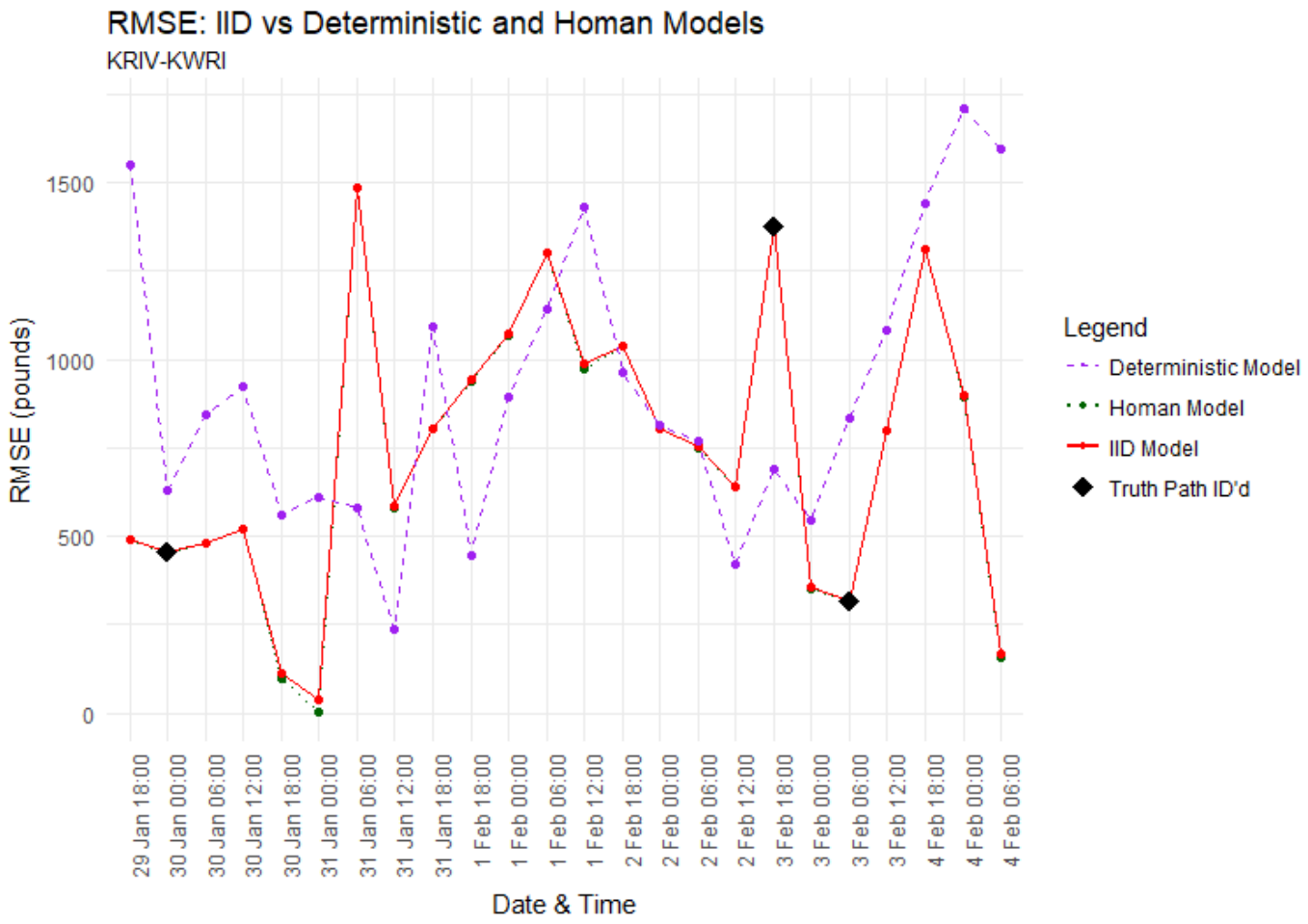

Figure 12. KRIV-KWRI: RMSE for IID, Deterministic, and Homan Models

The large spread between the deterministic model and Homan model is verified by the paired t-tests. These tests show that there is a statistically significant difference between the deterministic and Homan models, with a p-value of 2.35×10^{-2}.

Aside from the ability to generate confidence intervals around the mean and median, there is another advantage to the IID model. Of the 24 timesteps, 10 found multiple routes across all ensembles. These timesteps and their results are shown in Table 7. In five scenarios, only two unique routes were identifiedand the number of routes identified in the other five scenarios ranged from three to six. A total of 34 of the 42 total route comparisons performed identified a statistically significant dif-

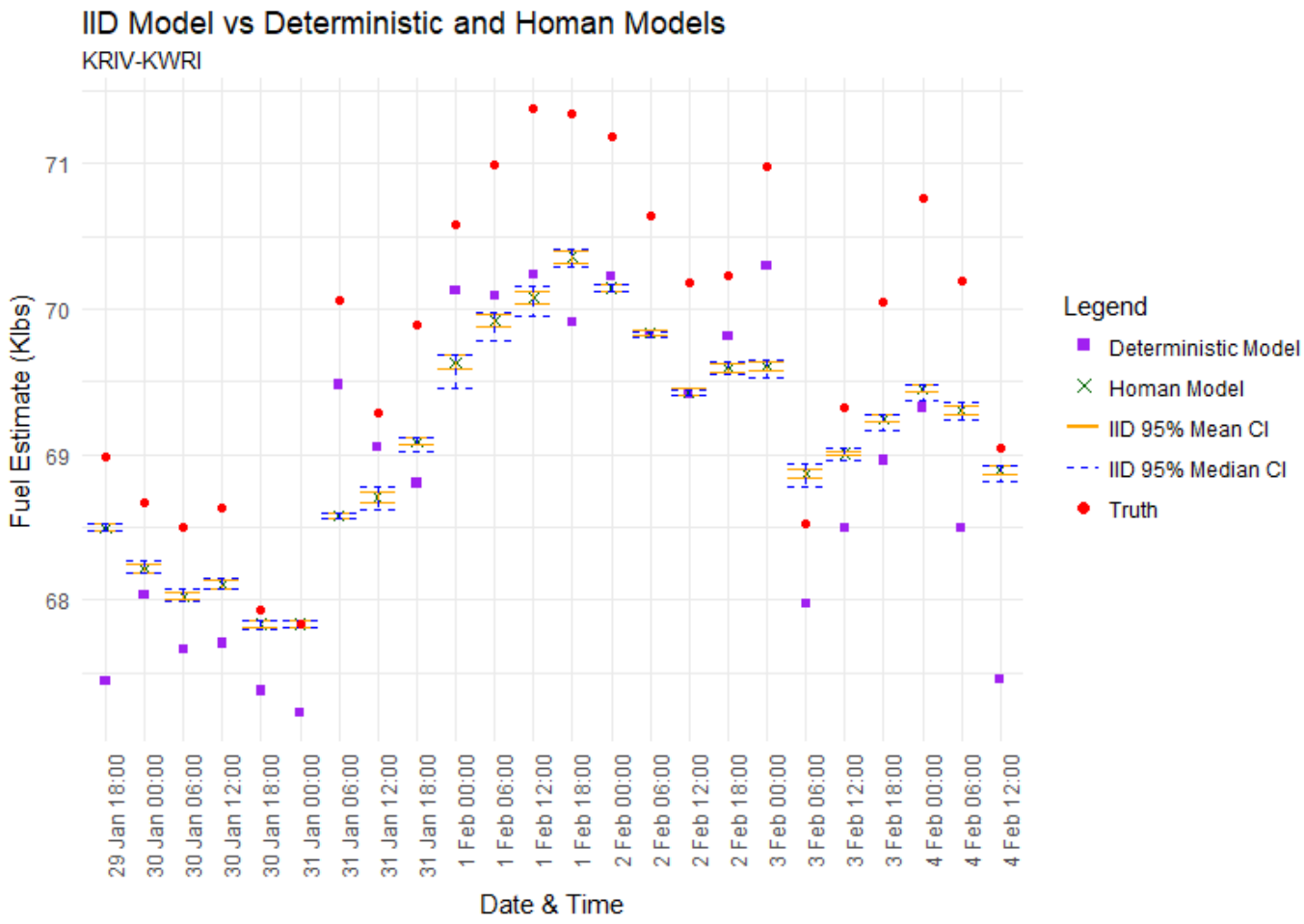

Figure 13. KRIV-KWRI: All model comparisons

ference between the means. So, depending on the day and time, multiple alternative routes could be suggested that will not use statistically significant more fuel.

For each date/time, the optimal routes identified by each model were compared to the optimal path of the *true* forecast. In 3/24 comparisons, the IID model identified the *true* optimal path (Figure 14). The Deterministic and Homan models did not identify the *true* path in any of the scenarios inspected. Figure 12 shows which date/times each model identified the *true* optimal path.

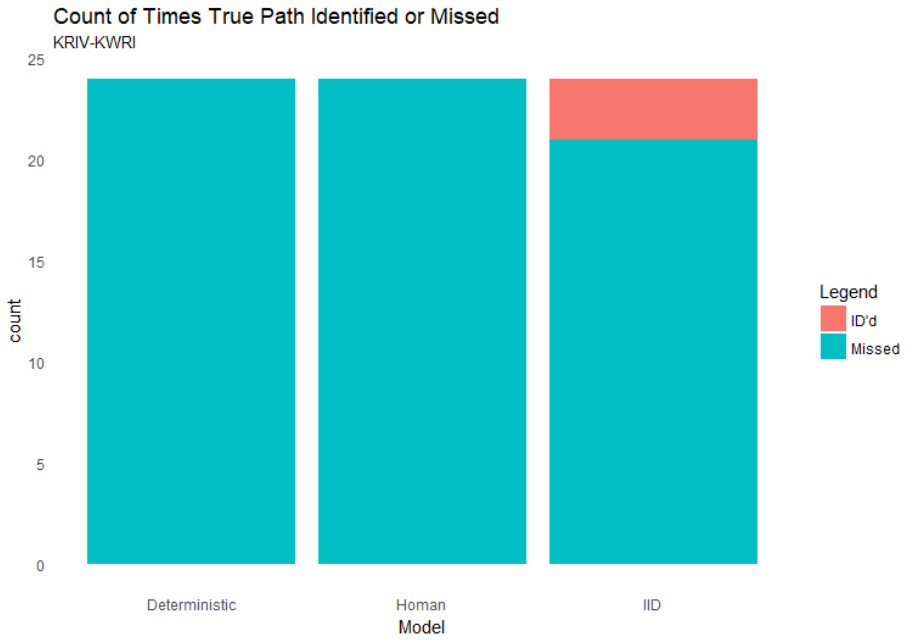

Figure 14. KRIV-KWRI: True Path Comparisons

Table 7. KRIV-KWRI: Unique Route Comparisons

Date & Time	Number of Routes	Comparison	p-value	Result
31 Jan 0600	3	$\mu_1 = \mu_2$	3.26×10^{-2}	$\neq$
		$\mu_1 = \mu_3$	0.236	$=$
		$\mu_2 = \mu_3$	9.90×10^{-6}	$\neq$
31 Jan 1200	2	$\mu_1 = \mu_2$	3.69×10^{-4}	$\neq$
31 Jan 1800	2	$\mu_1 = \mu_2$	0.269	$=$
1 Feb 1800	5	$\mu_1 = \mu_2$	2.81×10^{-8}	$\neq$
		$\mu_1 = \mu_3$	5.88×10^{-5}	$\neq$
		$\mu_1 = \mu_4$	0.035	$\neq$
		$\mu_1 = \mu_5$	0.236	$=$
		$\mu_2 = \mu_3$	0.299	$=$
		$\mu_2 = \mu_4$	0.148	$=$
		$\mu_2 = \mu_5$	0.034	$\neq$
		$\mu_3 = \mu_4$	3.22×10^{-4}	$\neq$
		$\mu_3 = \mu_5$	0.020	$\neq$
		$\mu_4 = \mu_5$	0.038	$\neq$

Table 7. (continued)

Date & Time	Number of Routes	Comparison	p-value	Result
1 Feb 0000	6	$\mu_1 = \mu_2$	4.10×10^{-5}	$\neq$
		$\mu_1 = \mu_3$	7.87×10^{-5}	$\neq$
		$\mu_1 = \mu_4$	2.00×10^{-4}	$\neq$
		$\mu_1 = \mu_5$	1.81×10^{-4}	$\neq$
		$\mu_1 = \mu_6$	2.02×10^{-4}	$\neq$
		$\mu_2 = \mu_3$	6.07×10^{-5}	$\neq$
		$\mu_2 = \mu_4$	1.95×10^{-4}	$\neq$
		$\mu_2 = \mu_5$	1.76×10^{-4}	$\neq$
		$\mu_2 = \mu_6$	1.97×10^{-4}	$\neq$
		$\mu_3 = \mu_4$	2.75×10^{-4}	$\neq$
		$\mu_3 = \mu_5$	2.46×10^{-4}	$\neq$
		$\mu_3 = \mu_6$	2.78×10^{-4}	$\neq$
		$\mu_4 = \mu_5$	6.90×10^{-9}	$\neq$
		$\mu_4 = \mu_6$	0.013	$\neq$
		$\mu_5 = \mu_6$	0.595	$=$
2 Feb 0000	2	$\mu_1 = \mu_2$	1.24×10^{-9}	$\neq$
3 Feb 1800	2	$\mu_1 = \mu_2$	2.69×10^{-4}	$\neq$
4 Feb 0600	3	$\mu_1 = \mu_2$	4.66×10^{-4}	$\neq$
		$\mu_1 = \mu_3$	0.011	$\neq$
		$\mu_2 = \mu_3$	1.01×10^{-4}	$\neq$
4 Feb 1200	2	$\mu_1 = \mu_2$	0.101	$=$
4 Feb 1800	4	$\mu_1 = \mu_2$	3.13×10^{-7}	$\neq$
		$\mu_1 = \mu_3$	5.27×10^{-7}	$\neq$
		$\mu_1 = \mu_4$	6.51×10^{-4}	$\neq$
		$\mu_2 = \mu_3$	0.949	$=$
		$\mu_2 = \mu_4$	5.08×10^{-4}	$\neq$
		$\mu_3 = \mu_4$	2.34×10^{-4}	$\neq$

V. Conclusions and Future Research

5.1 Conclusion

By incorporating ensemble NWP into the route planning, AMC can reduce the amount of excess fuel burned by poor forecasts. The Homan model performed well when compared to the IID but due to the inability to account for the inter-correlation within each ensemble member and the correlation across ensembles, this model is not ideal.

Of the three models discussed, the Deterministic model almost always over-estimated fuel burn compared to the IID and Homan models; sometimes up to almost 4,500 pounds of fuel. For one of the four routes inspected, the Deterministic model under-estimated the fuel burn up to 2,000 pounds. While these over-estimations of fuel consumption could result in up to 671 gallons of excess fuel, these estimates are only for a small subset of coast-to-coast routes that AMC flies regularly and only for the C-17 aircraft. When not over-estimating the fuel necessary, the Deterministic model under-estimated the fuel required up to almost 300 pounds of fuel. The average RMSE for the IID and Homan models across all routes investigated was roughly 512 and 501 pounds, respectively, while the average RMSE of the Deterministic model was almost 2,500 pounds of fuel. On average, the Deterministic model misses the *true* fuel burn by nearly 2,000 more pounds than the IID and Homan models. These severe inconsistencies in fuel burn estimates can make it difficult for appropriate route planning. The amount of wasted or insufficient fuel will add up quickly for longer, *i.e.* transoceanic, routes resulting in excess costs or dangerous situations.

Applying the APSP methodology in the IID model provides users with additional information and more fidelity. The deterministic and Homan models provide point estimates. The IID model provides a range of potential values which further aids in

flight planning. It has the ability to provide multiple routes that will not statistically change the amount of fuel used. Additionally, the IID model was the only model to ever identify the *true* optimal path during the testing period.

5.2 Future Research

This research only accounts for aircraft performance and weather in determining the optimal route. A more useful flight plan should also include route restrictions from Air Traffic Control (ATC) and relevant regulatory restrictions [2]. This application only accounts for the effects of wind on fuel consumption, what about other weather conditions? As a proof of concept, several simplifying assumptions were made: constant TAS, linearity between time, space, and points, and only looking at the great circle route. Building upon these assumptions would yield a more accurate tool for identifying optimal aircraft routing and estimating fuel consumption. Another interesting expansion of this application would be to look at lateral route deviations instead of just vertical deviations.

This research focused on shorter, coast-to-coast routes. While differences between the current methodology and the application of the APSP, a study should be conducted to investigate the differences between the models for longer flights. In addition, a study should also be conducted to identify a distribution of fuel by month, day, etc. This study could leverage the +00 hour forecasts from this analysis and the National Centers for Environmental Prediction (NCEP) GFS Historical Archive [35], which has weather data from 15 January 2015 to 21 February 2018.

AMC aircraft are large and can handle the affects of wind better than smaller aircraft. This technique could provide valuable insight to the small aircraft and drone communities, because they are heavily impacted by winds. This *a posteriori* approach could also apply to communication networks.

Appendix A. Matlab Code: Data Extraction

A Data Extraction Loop

Code/PullData.m

```
clear;
clc;

% Add the current directory and subdirectories to the path
addpath(genpath(pwd()));

% Get the current UTC
t1 = datetime('now','TimeZone','utc');

% Strip off the minutes and seconds
t1 = t1 - minutes(minute(t1)) - seconds(second(t1));

% Count back to the model time
while (mod(hour(t1),6) ~= 0)
    t1 = t1 - hours(1);
end

t2 = t1 - days(7); %go back 7 days

strt_trial = 100;
num_trial = 1;
stepsize = 100;
nlegs = 100;

% Pressure levels
levels = [400,350,300,250];  %pressure levels interested in

num_mem = 20; %number of ensemble members
TAS = 450; %constant TAS


routes = {'KSUU-PHNL','KTCM-CYQX','KTCM-KCHS', 'KRIV-KWRI'};

% lat/longs from airnav.com

latlongs_routes = [38.2645367, -121.9241315, 21.3178275, -157.9202627; %Travis lat/long, Honolulu lat\long
                   47.1376778, -122.4764750, 48.936944, -54.567778 ; %McChord Lat/Long, Gander lat/long (from skyvector)
                   47.1376778, -122.4764750, 32.8986389, -80.0405278; %McChord Lat/Long, %Charleston
                   33.8819433 -117.2590169, 40.0155833, -74.5916991]; %March Air Reserve Air base, JBMDL lat/long

[num_routes,~] = size(latlongs_routes);

```

```
43
44 while (t1 >= t2) % Loop back seven days
45
46     for i = 1:num_routes  %create all folders based on route names before pulling data
47
48
49         FolderName = sprintf('%s',routes{i}); %save in folder named after route
50         mkdir('../Thesis Docs/Data', FolderName)  %create new folder under data tab
51
52         LatA = latlongs_routes(i,1);
53         LongA = latlongs_routes(i,2);
54         LatB = latlongs_routes(i,3);
55         LongB = latlongs_routes(i,4);
56         latlongmat = gcwaypts(LatA,LongA,LatB,LongB,nlegs);
57         %% Pull Weather Data
58
59         % Hours in forecast
60         tot_dist = deg2nm(distance('gc',[LatA,LongA],[LatB,LongB])); %calculate total distance
               from pt A to pt B across great circle route in nautical miles
61         totalHrs = 2*ceil(tot_dist/TAS); % double time to travel across gc route and round up (in
               hours) --> amount of hours at least to pull weather data for
62
63         while (mod(totalHrs,6) ~= 0)   % Adjust the total hours to a multiple of 6 (model is in 6
               hour increments)
64             totalHrs = totalHrs + 1;
65         end
66
67         % Left/Right longitude
68             leftlon = mod(floor(min(latlongmat(:,2))),360);
69             rightlon = mod(ceil(max(latlongmat(:,2))),360);
70         % Top/Bottom latitude
71
72             toplat = ceil(max(latlongmat(:,1))); %% ceil(LatA);
73             bottomlat = floor(min(latlongmat(:,1))); %floor(LatB);
74 err = 1;
75 while err == 1
76         % Gets the winds and temps
77         [ ~, U, V, W, lat, lon, iso ] = Ensemble_Wind_Temp( t1,levels, ...
78                                     totalHrs, leftlon, rightlon, toplat, bottomlat );
79
80         if (isempty(U)  || isempty(V) || isempty(W) || ...
81                 isempty(lat) || isempty(lon) || isempty(iso))
82             disp(strcat('Route',32,routes{i},32,'download failed for ensemble GFS run
                    starting',32,datestr(t1)));
83             err = 1;
84             %return;
85         else
86             %file_name =
                    sprintf('%02i%02i_%02i%02i_Ensemble.mat',day(W(1)),month(W(1)),hour(W(1)),0);
87             if month(W(1)) == 2
88                 mnth = 'Feb';
89             else mnth = 'Jan';
90             end
91             file_name = sprintf('%02i%s_%02i%02i_Ensemble.mat',day(W(1)),mnth,hour(W(1)),0)
```

```
92              matfile = fullfile(pwd,FolderName, file_name);
93              save(matfile, 'U', 'V', 'lat', 'lon','iso');
94              err = 0;
95          end
96  end
97
98
99  while err == 0
100     [ ~, U, V, W, lat, lon, iso ] = Deterministic_Wind_Temp(t1, levels, ...
101                                     totalHrs, leftlon, rightlon, toplat, bottomlat );
102         if (isempty(U)  || isempty(V) || isempty(W) || ...
103                 isempty(lat) || isempty(lon) || isempty(iso))
104             disp(strcat('Route',32,routes{i},32,'download failed for deterministic GFS run
                    starting',32,datestr(t1)));
105             err = 0;
106             %return;
107         else
108             %file_name =
                    sprintf('%02i%02i_%02i%02i_Deterministic.mat',day(W(1)),month(W(1)),hour(W(1)),0);
109             file_name = sprintf('%02i%s_%02i%02i_Deterministic.mat',day(W(1)),mnth,hour(W(1)),0)
110             matfile = fullfile(pwd,FolderName, file_name);
111             save(matfile, 'U', 'V', 'lat', 'lon','iso');
112             err = 1;
113         end
114     end
115 end
116
117     t1 = t1 - hours(6);  % decrement model initialization time by 6 hours
118 end
```

B Ensemble Data Extraction

Code/Ensemble_Wind_Temp.m

```
1  function [ T, U, V, W, lat, lon, iso ] = Ensemble_Wind_Temp( t1, levels, ...
2                              totalHrs, leftlon, rightlon, toplat, bottomlat )
3  % This function retrieves the GFS 1 degree ensemble Numerical Weather
4  % Prediction (NWP) model wind and temperature output for the requested
5  % level(s) and time period(s).
6
7  % This function requires the nctoolbox package:
8  %                                  https://github.com/nctoolbox/nctoolbox
9  % Make sure to include the nctoolbox directory in the MATLAB path.
10
11 % Input variables:
12 % levels-isobaric pressure levels (1000,925,850,700,500,400 and/or 300 mb)
13 % totalHrs - total number of hours of forecast data needed
14 % leftlon - Western edge of longitude window (0 - 360)
15 % rightlon - Eastern edge of longitude window (0 - 360)
16 % toplat - Northern edge of latitude window (-90 - 90)
17 % bottomlat - Southern edge of latitude window (-90 - 90)
18
```

```
19 W = [];          % NWP model step times
20 lat = [];        % Latitudes in degrees (-90 - 90)
21 lon  = [];       % Longitudes in degrees (0 - 360)
22 iso = [];        % Isobaric pressure levels (translates to altitude MSL)
23 % Approximate altitude calculations from pressure can be found here:
24 % http://ww2010.atmos.uiuc.edu/(Gh)/wwhlpr/constant_pressure_surface.rxml
25
26 % Dimensions in order are time (entries in W), ensemble member (1 to 20),
27 % pressure level (entries in iso in millibars), latitude, and longitude
28 T = [];      % Temperature values in degrees Celsius
29 U = [];      % U component of winds in meters per second
30 V = [];      % V component of winds in meters per second
31 % Explanation for how to use U and V wind components can be found here:
32 % http://colaweb.gmu.edu/dev/clim301/lectures/wind/wind-uv.html
33
34 % Make sure top and bottom lat are correctly configured
35 if bottomlat > toplat
36     temp=bottomlat;
37     bottomlat = toplat;
38     toplat=temp;
39 end
40
41 if toplat > 90 || bottomlat < -90
42     return;
43 end
44
45 % Make sure left and right longitudes are correctly configured
46 if leftlon > rightlon
47     temp = rightlon;
48     rightlon = leftlon;
49     leftlon = temp;
50 end
51
52 if leftlon < 0 || rightlon > 360
53     return;
54 end
55
56 % Set up the pressure levels to ensure only standard levels are entered
57 levels = intersect(levels,[1000,975,950,925,900,850,800,750,700,650,600,...
58   550,500,475,450,400,350,300,250,200,150,125,100,70,50,30,20,10,7,5,2,1]);
59
60 % Set up nctoolbox
61 setup_nctoolbox;
62
63 % Count back to the model time
64 while (mod(hour(t1),6) ~= 0)
65     t1 = t1 - hours(1);
66 end
67
68 % Adjust the total hours to a multiple of 6 (model is in 6 hour increments)
69 while (mod(totalHrs,6) ~= 0)
70     totalHrs = totalHrs + 1;
71 end
72
```

```
73 t2 = t1-hours(12);
74
75 DownLoadError = 1;
76
77 while (DownLoadError && t1 >= t2)
78
79     % Store model data times
80     W = t1 + hours(0:6:totalHrs);
81
82     for j = 0:6:totalHrs
83
84         for i = 1:20
85
86             % The weather data URL
87             URL = 'http://nomads.ncep.noaa.gov/cgi-bin/filter_gens.pl?file=gep';
88             URL=strcat(URL,num2str(i,'%02i'),'.t',num2str(hour(t1),'%02i'));
89             URL=strcat(URL,'z.pgrb2f',num2str(j,'%02i'));
90             for k = 1:1:max(size(levels))
91                 URL=strcat(URL, '&lev_',num2str(levels(k)),'_mb=on');
92             end
93             URL=strcat(URL,'&var_TMP=on&var_UGRD=on&var_VGRD=on&subregion=&leftlon=',num2str(leftlon));
94             URL=strcat(URL,'&rightlon=',num2str(rightlon),'&toplat=',num2str(toplat), ...
95                 '&bottomlat=',num2str(bottomlat),'&dir=%2Fgefs.');
96             URL=strcat(URL,num2str(year(t1)));
97             URL=strcat(URL,num2str(month(t1),'%02i'));
98             URL=strcat(URL,num2str(day(t1),'%02i'));
99             URL=strcat(URL,'%2F',num2str(hour(t1),'%02i'),'%2Fpgrb2');
100            fileName = strcat(pwd(),'/winds_',num2str(j),'_',num2str(i),'.grib2');
101
102            % Download weather data
103            try
104                outfilename = websave(fileName,URL);
105            catch
106                if (j==0 && i == 1)
107                    t1 = t1 - hours(6);
108                    totalHrs = totalHrs + 6;
109                    DownLoadError = 1;
110                    break;
111                else
112                    return;
113                end
114            end
115
116            % Import weather data into MATLAB
117            nc = ncgeodataset(outfilename);
118
119            % Fill values for T, U and V
120            V(floor(j/6)+1,i,:,:,:) = nc{'v-component_of_wind_isobaric'}(:); %#ok<AGROW>
121            U(floor(j/6)+1,i,:,:,:) =  nc{'u-component_of_wind_isobaric'}(:); %#ok<AGROW>
122            T(floor(j/6)+1,i,:,:,:) =  nc{'Temperature_isobaric'}(:); %#ok<AGROW>
123            DownLoadError = 0;
124        end
125        if (DownLoadError == 1)
126            break;
```

```
127         end
128     end
129 end
130
131 if (DownLoadError == 1)
132    return;
133 end
134
135 lat = nc{'lat'}(:);
136 lon = nc{'lon'}(:);
137 iso = nc{'isobaric'}(:)/100;
138
139 for j = 0:6:totalHrs
140     for i = 1:20
141         outfilename = strcat(pwd(),'/winds_',num2str(j),'_',num2str(i),'.grib2');
142         % Delete the weather data file
143          delete(outfilename);
144          delete(strcat(outfilename,'.gbx9'));
145          delete(strcat(outfilename,'.ncx'));
146     end
147 end
148
149 end
```

C Deterministic Data Extraction

Code/Deterministic_Wind_Temp.m

```
1  function [ T, U, V, W, lat, lon, iso] = Deterministic_Wind_Temp( t1, levels, ...
2                              totalHrs, leftlon, rightlon, toplat, bottomlat )
3  % This function retrieves the GFS 1 degree deterministic Numerical Weather
4  % Prediction (NWP) model wind and temperature output for the requested
5  % level(s) and time period(s).
6
7  % This function requires the nctoolbox package:
8  %                                    https://github.com/nctoolbox/nctoolbox
9  % Make sure to include the nctoolbox directory in the MATLAB path.
10
11 % Input variables:
12 % levels-isobaric pressure levels (1000,925,850,700,500,400 and/or 300 mb)
13 % totalHrs - total number of hours of forecast data needed
14 % leftlon - Western edge of longitude window (0 - 360)
15 % rightlon - Eastern edge of longitude window (0 - 360)
16 % toplat - Northern edge of latitude window (-90 - 90)
17 % bottomlat - Southern edge of latitude window (-90 - 90)
18
19 W = [];          % NWP model step times
20 lat = [];        % Latitudes in degrees (-90 - 90)
21 lon  = [];       % Longitudes in degrees (0 - 360)
22 iso = [];        % Isobaric pressure levels (translates to altitude MSL)
23 % Approximate altitude calculations from pressure can be found here:
24 % http://ww2010.atmos.uiuc.edu/(Gh)/wwhlpr/constant_pressure_surface.rxml
```

```
25
26 % Dimensions in order are time (entries in W), ensemble member (1 to 20),
27 % pressure level (entries in iso in millibars), latitude, and longitude
28 T = [];      % Temperature values in degrees Celsius
29 U = [];      % U component of winds in meters per second
30 V = [];      % V component of winds in meters per second
31 % Explanation for how to use U and V wind components can be found here:
32 % http://colaweb.gmu.edu/dev/clim301/lectures/wind/wind-uv.html
33
34 % Make sure top and bottom lat are correctly configured
35 if bottomlat > toplat
36     temp=bottomlat;
37     bottomlat = toplat;
38     toplat=temp;
39 end
40
41 if toplat > 90 || bottomlat < -90
42     return;
43 end
44
45 % Make sure left and right longitudes are correctly configured
46 if leftlon > rightlon
47     temp = rightlon;
48     rightlon = leftlon;
49     leftlon = temp;
50 end
51
52 if leftlon < 0 || rightlon > 360
53     return;
54 end
55
56 % Set up the pressure levels to ensure only standard levels are entered
57 levels = intersect(levels,[1000,975,950,925,900,850,800,750,700,650,600,...
58   550,500,475,450,400,350,300,250,200,150,125,100,70,50,30,20,10,7,5,2,1]);
59
60
61 % Set up nctoolbox
62 setup_nctoolbox;
63
64 % Count back to the model time
65 while (mod(hour(t1),6) ~= 0)
66     t1 = t1 - hours(1);
67 end
68
69 % Adjust the total hours to a multiple of 6 (model is in 6 hour increments)
70 while (mod(totalHrs,6) ~= 0)
71     totalHrs = totalHrs + 1;
72 end
73
74 t2 = t1-hours(12);
75
76 DownLoadError = 1;
77
78 while (DownLoadError && t1 >= t2)
```

```
79
80      % Store model data times
81      W = t1 + hours(0:3:totalHrs);
82
83      for j = 0:3:totalHrs
84
85          % The weather data URL
86          URL = 'http://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_1p00.pl?file=gfs.t';
87          URL=strcat(URL,num2str(hour(t1),'%02i'),'z.pgrb2.1p00.f',num2str(j,'%03i'));
88          for k = 1:1:max(size(levels))
89              URL=strcat(URL, '&lev_',num2str(levels(k)),'_mb=on');
90          end
91          URL=strcat(URL,'&var_TMP=on&var_UGRD=on&var_VGRD=on&subregion=&leftlon=',num2str(leftlon));
92          URL=strcat(URL,'&rightlon=',num2str(rightlon),'&toplat=',num2str(toplat), ...
93              '&bottomlat=',num2str(bottomlat),'&dir=%2Fgfs.');
94          URL=strcat(URL,num2str(year(t1)));
95          URL=strcat(URL,num2str(month(t1),'%02i'));
96          URL=strcat(URL,num2str(day(t1),'%02i'));
97          URL=strcat(URL,num2str(hour(t1),'%02i'));
98          fileName = strcat(pwd(),'/deterministic_winds_',num2str(j),'.grib2');
99
100         % Download weather data
101         try
102             outfilename = websave(fileName,URL);
103         catch
104             if (j==0 )
105                 t1 = t1 - hours(6);
106                 totalHrs = totalHrs + 6;
107                 DownLoadError = 1;
108                 break;
109             else
110                 return;
111             end
112         end
113
114         % Import weather data into MATLAB
115         nc = ncgeodataset(outfilename);
116
117         % Fill values for T, U and V
118         V(floor(j/3)+1,1,:,:,:) = nc{'v-component_of_wind_isobaric'}(:); %#ok<AGROW>
119         U(floor(j/3)+1,1,:,:,:) =  nc{'u-component_of_wind_isobaric'}(:); %#ok<AGROW>
120         T(floor(j/3)+1,1,:,:,:) =  nc{'Temperature_isobaric'}(:); %#ok<AGROW>
121         DownLoadError = 0;
122     end
123
124 end
125
126 if (DownLoadError == 1)
127    return;
128 end
129
130 lat = nc{'lat'}(:);
131 lon = nc{'lon'}(:);
132 iso = nc{'isobaric'}(:)/100;
```

```
133
134 for j = 0:3:totalHrs
135     outfilename = strcat(pwd(),'/deterministic_winds_',num2str(j),'.grib2');
136     % Delete the weather data file
137     delete(outfilename);
138     delete(strcat(outfilename,'.gbx9'));
139     delete(strcat(outfilename,'.ncx'));
140 end
141
142 end
```

Appendix B. Matlab Code: Network Building

A Linear Interpolation in Time and Space

Code/InterpolateAllData.m

```
routes = {'KSUU-PHNL','KTCM-CYQX', 'KTCM-KCHS', 'KRIV-KWRI'};

% lat/longs from airnav.com

latlongs_routes = [38.2645367, -121.9241315, 21.3178275, -157.9202627; %Travis lat/long, Honolulu lat\long
                   47.1376778, -122.4764750, 48.936944, -54.567778 ; %McChord Lat/Long, Gander lat/long (from skyvector)
                   47.1376778, -122.4764750, 32.8986389, -80.0405278; %McChord Lat/Long, %Charleston
                   33.8819433 -117.2590169, 40.0155833, -74.5916991]; %March Air Reserve Air base, JBMDL lat/long

[num_routes,~] = size(latlongs_routes);
%mkdir Data
for i =1:num_routes  %create all folders based on route names before pulling data
    FolderName = sprintf('%s',routes{i}) %save in folder named after route
    matfile = fullfile('C:\Users\smboo\Desktop\Thesis (1)\Thesis\Data',FolderName);
    cd(matfile)
    addpath(genpath('C:\Users\smboo\Desktop\Thesis (1)\Thesis'))
files = dir('*.mat');
for file = files'
    load(file.name);
    [num_its,~,~,~] = size(U)
    tic
%% issues with interpolation code, can only interpolate between two times, this is a workaround
% NOTE: this works for the short routes we are investigating during this
% research, will need to adjust for longer routes (i.e. routes > 6 hours)

   if file.name(12) == 'D'  %determine if its ensemble or deterministic data
       model = 'D';
       num_mem = 1;
        for j = 3:num_its-2
        [U_interp1, V_interp1] = time_alt_interp(U(j:j+1,:,:,:,:),V(j:j+1,:,:,:,:),model,num_mem);
        [U_interp2, V_interp2] =
             time_alt_interp(U(j+1:j+2,:,:,:,:),V(j+1:j+2,:,:,:,:),model,num_mem);
        U_interp = [U_interp1 ; U_interp2(2:end,:,:,:,:)];
        V_interp = [V_interp1 ; V_interp2(2:end,:,:,:,:)];
    end
   else
       model = 'E';
       num_mem = 20;
       [U_interp, V_interp] = time_alt_interp(U(2:end,:,:,:,:),V(2:end,:,:,:,:),model,num_mem);
   end


```

```
42      time_elapsed = toc;
43      if time_elapsed > 60
44          time_elapsed = time_elapsed/60;
45          metric = 'min';
46      else
47          metric = 'sec';
48          end
49       matfile = fullfile('C:\Users\smboo\Desktop\Thesis (1)\Thesis\Data',FolderName,file.name);
50       save(matfile,'U_interp','V_interp','-append')
51       fprintf('%s: %.2f %s\n',file.name, time_elapsed,metric)
52
53 end
54 clear matfile
55 end
```

Code/time_alt_interp.m

```
1
2 function [U_interp, V_interp] = time_alt_interp(U,V,model,num_mem)
3 if model == 'E' %if ensemble data
4 %% time interpolation
5 x = [0;6]; %because time is in 6 hour timesteps, we have time 0 and time 6
6 xi = [0:6];  %interpolating between 0 and 6 (included so that they are in the output vector)
7 [num_times,~,num_alts,num_lats,num_longs] = size(U);
8 U_time = []; %initialize matrix for concating
9 V_time = [];
10 for m = 1:num_times-1
11 for l = 1:num_longs
12 for k = 1:num_lats
13 for j = 1:num_alts
14 for i = 1:num_mem
15     y_U = U(m:m+1,i,j,k,l);
16     U_time_temp(:,i,j,k,l) = interp1(x,y_U,xi); %interpolate across U
17     y_V = V(m:m+1,i,j,k,l);
18     V_time_temp(:,i,j,k,l) = interp1(x,y_V,xi);  %interpolate across V
19 end
20 if m > 1
21     [cur_row_U,~] = size(U_time);  %ensure no duplicate rows
22     U_time(cur_row_U:cur_row_U+6,:,j,k,l) = U_time_temp(:,:,j,k,l);
23     [cur_row_V,~] = size(V_time);  %ensure no duplicate rows
24     V_time(cur_row_V:cur_row_V+6,:,j,k,l) = V_time_temp(:,:,j,k,l);
25 else
26    U_time(:,:,j,k,l) = U_time_temp(:,:,j,k,l); %add temporary to U with interpolation
27    V_time(:,:,j,k,l) = V_time_temp(:,:,j,k,l); %add temporary to V with interpolation
28 end
29 end
30 end
31 end
32 end
33
34 %% altitude interpolation
35 [num_times,~,num_alts,num_lats,num_longs] = size(U_time);
36 U_interp = zeros(num_times, 20, 11, num_lats, num_longs);
37 V_interp = zeros(num_times, 20, 11, num_lats, num_longs);
```

```
for l = 1:num_longs
for k = 1:num_lats

step_low_U = (U_time(:,:,2,k,l) - U_time(:,:,1,k,l))/5;
step_high_U = (U_time(:,:,3,k,l) - U_time(:,:,2,k,l))/5;

step_low_V = (V_time(:,:,2,k,l) - V_time(:,:,1,k,l))/5;
step_high_V = (V_time(:,:,3,k,l) - V_time(:,:,2,k,l))/5;
for incr = 0:10
    if incr <= 5
    U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_low_U; %each new dim in num_alts
        will be 1000k increments from 25:35k
    V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_low_V; %each new dim in num_alts
        will be 1000k increments from 25:35k

    else
    U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_high_U; %each new dim in num_alts
        will be 1000k increments from 25:35k
    V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_high_V; %each new dim in num_alts
        will be 1000k increments from 25:35k
    end
end
end
end
elseif model == 'D' %if deterministic data
%% time interpolation
x = [0;3]; %because time is in 6 hour timesteps, we have time 0 and time 6
xi = [0:3];  %interpolating between 0 and 6 (included so that they are in the output vector)
[num_times,~,num_alts,num_lats,num_longs] = size(U);
U_time = []; %initialize matrix for concating
V_time = [];
for m = 1:num_times-1
for l = 1:num_longs
for k = 1:num_lats
for j = 1:num_alts
for i = 1:num_mem
    y_U = U(m:m+1,i,j,k,l);
    U_time_temp(:,i,j,k,l) = interp1(x,y_U,xi); %interpolate across U
    y_V = V(m:m+1,i,j,k,l);
    V_time_temp(:,i,j,k,l) = interp1(x,y_V,xi);  %interpolate across V
end
if m > 1
    [cur_row_U,~] = size(U_time);  %ensure no duplicate rows
    U_time(cur_row_U:cur_row_U+3,:,j,k,l) = U_time_temp(:,:,j,k,l);
    [cur_row_V,~] = size(V_time);  %ensure no duplicate rows
    V_time(cur_row_V:cur_row_V+3,:,j,k,l) = V_time_temp(:,:,j,k,l);
else
   U_time(:,:,j,k,l) = U_time_temp(:,:,j,k,l); %add temporary to U with interpolation
   V_time(:,:,j,k,l) = V_time_temp(:,:,j,k,l); %add temporary to V with interpolation
end
end
end
end
end
```

```
88
89 %% altitude interpolation
90 [num_times,~,num_alts,num_lats,num_longs] = size(U_time);
91 U_interp = zeros(num_times, 1, 11, num_lats, num_longs);
92 V_interp = zeros(num_times, 1, 11, num_lats, num_longs);
93 for l = 1:num_longs
94 for k = 1:num_lats
95
96 step_low_U = (U_time(:,:,2,k,l) - U_time(:,:,1,k,l))/2;
97 step_high_U = (U_time(:,:,3,k,l) - U_time(:,:,2,k,l))/2;
98
99 step_low_V = (V_time(:,:,2,k,l) - V_time(:,:,1,k,l))/2;
100 step_high_V = (V_time(:,:,3,k,l) - V_time(:,:,2,k,l))/2;
101 for incr = 0:10
102     if incr <= 2
103     U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_low_U; %each new dim in num_alts
            will be 1000k increments from 25:35k
104     V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_low_V; %each new dim in num_alts
            will be 1000k increments from 25:35k
105
106     else
107     U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_high_U; %each new dim in num_alts
            will be 1000k increments from 25:35k
108     V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_high_V; %each new dim in num_alts
            will be 1000k increments from 25:35k
109     end
110 end
111 end
112 end
113 else
114 %% Truth Source
115 %% time interpolation
116 x = [0;6]; %because time is in 6 hour timesteps, we have time 0 and time 6
117 xi = [0:6];  %interpolating between 0 and 6 (included so that they are in the output vector)
118 [num_times,~,num_alts,num_lats,num_longs] = size(U);
119 U_time = []; %initialize matrix for concating
120 V_time = [];
121 for m = 1:num_times-1
122 for l = 1:num_longs
123 for k = 1:num_lats
124 for j = 1:num_alts
125 for i = 1:num_mem
126     y_U = U(m:m+1,i,j,k,l);
127     U_time_temp(:,i,j,k,l) = interp1(x,y_U,xi); %interpolate across U
128     y_V = V(m:m+1,i,j,k,l);
129     V_time_temp(:,i,j,k,l) = interp1(x,y_V,xi);  %interpolate across V
130 end
131 if m > 1
132     [cur_row_U,~] = size(U_time);  %ensure no duplicate rows
133     U_time(cur_row_U:cur_row_U+6,:,j,k,l) = U_time_temp(:,:,j,k,l);
134     [cur_row_V,~] = size(V_time);  %ensure no duplicate rows
135     V_time(cur_row_V:cur_row_V+6,:,j,k,l) = V_time_temp(:,:,j,k,l);
136 else
137    U_time(:,:,j,k,l) = U_time_temp(:,:,j,k,l); %add temporary to U with interpolation
```

```
138     V_time(:,:,j,k,l) = V_time_temp(:,:,j,k,l); %add temporary to V with interpolation
139 end
140 end
141 end
142 end
143 end
144
145 %% altitude interpolation
146 [num_times,~,num_alts,num_lats,num_longs] = size(U_time);
147 U_interp = zeros(num_times, 1, 11, num_lats, num_longs);
148 V_interp = zeros(num_times, 1, 11, num_lats, num_longs);
149 for l = 1:num_longs
150 for k = 1:num_lats
151
152 step_low_U = (U_time(:,:,2,k,l) - U_time(:,:,1,k,l))/2;
153 step_high_U = (U_time(:,:,3,k,l) - U_time(:,:,2,k,l))/2;
154
155 step_low_V = (V_time(:,:,2,k,l) - V_time(:,:,1,k,l))/2;
156 step_high_V = (V_time(:,:,3,k,l) - V_time(:,:,2,k,l))/2;
157 for incr = 0:10
158     if incr <= 2
159     U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_low_U; %each new dim in num_alts
            will be 1000k increments from 25:35k
160     V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_low_V; %each new dim in num_alts
            will be 1000k increments from 25:35k
161
162     else
163     U_interp(:,:,incr+1,k,l) = U_time(:,:,1,k,l) + incr*step_high_U; %each new dim in num_alts
            will be 1000k increments from 25:35k
164     V_interp(:,:,incr+1,k,l) = V_time(:,:,1,k,l) + incr*step_high_V; %each new dim in num_alts
            will be 1000k increments from 25:35k
165     end
166 end
167 end
168 end
169
170 end
171 end
```

B Wind Calculations

Code/WindCalcs_26Oct.m

```
1 function [optimal_value, path, DG] = windcalcs(nlegs, mem_num, latlongmat, lat, U_interp,
      V_interp, lon, TAS,model,num_mem)
2
3 AC = 2; %use C-17 regression models
4 omega = 496.5; %AC gross weight estimate
5 PW = 5; %payload weight estimate
6
7 load RegressionCoeffs.mat %load Betas for Reiman regression models
8
```

```
%%
U_temp = U_interp(:,mem_num,:,:,:);  % look at one ensemble member at a time
U_mem = squeeze(U_temp);  % reduce size of U since ensemble mem dimension went from 20 to 1 (now
    singular)
% Positive: West

V_temp = V_interp(:,mem_num,:,:,:);  % look at one ensemble member at a time
V_mem = squeeze(V_temp);
% Positive: South

[nhrs,~,~,~,~] = size(U_interp);
%% Determine U and V components along the route
%1 degree model, so need to look at each degree. so we can either:
% Used a weighted average based on the location of the actual data pt wrt to the upper/lower
lat_rnd = round(latlongmat(:,1)); % just looking at the lower bnds for lat
long_lw = floor(latlongmat(:,2));
long_up = ceil(latlongmat(:,2));


lon = mod(lon,360);  %change lon coordinate from -180 to 180 to 0 to 360

for i = 1:nlegs+1  %find idx of our route among all data
    long_idx_lw(i) = find(lon == long_lw(i));
    long_idx_up(i) = find(lon == long_up(i));
    lat_idx(i) = find(lat == lat_rnd(i));
end


for i = 1:nlegs  %determine U and V components along route using a weighted average between
    degrees; dims: time, altitude, leg
U_route(:,:,i) = ((U_mem(:,:,lat_idx(i+1),long_idx_lw(i+1))*(latlongmat(i+1,2) - long_lw(i+1)) +
    U_mem(:,:,lat_idx(i+1),long_idx_up(i+1))*(1-(latlongmat(i+1,2) -
    long_lw(i+1))))+(U_mem(:,:,lat_idx(i),long_idx_lw(i))*(latlongmat(i,2) - long_lw(i)) +
    U_mem(:,:,lat_idx(i),long_idx_up(i))*(1-(latlongmat(i,2) - long_lw(i)))))/2;
V_route(:,:,i) = ((V_mem(:,:,lat_idx(i+1),long_idx_lw(i+1))*(latlongmat(i+1,2) - long_lw(i+1)) +
    V_mem(:,:,lat_idx(i+1),long_idx_up(i+1))*(1-(latlongmat(i+1,2) -
    long_lw(i+1))))+(V_mem(:,:,lat_idx(i),long_idx_lw(i))*(latlongmat(i,2) - long_lw(i)) +
    V_mem(:,:,lat_idx(i),long_idx_up(i))*(1-(latlongmat(i,2) - long_lw(i)))))/2;
end

%% calculate headwind/tailwinds
WS = sqrt(U_route.^2 + V_route.^2); %Calculate windspeed m/s.. converted to knots later
angle_W = atan2(-U_route, -V_route)*180/pi; %angle in degrees

for i = 2:nlegs+1  % calculate the aircraft heading between waypoints i: from, j: to
ac_heading_orig(i-1,1) =
    mod(atan2d(sin(latlongmat(i,2)-latlongmat(i-1,2))*cos(latlongmat(i,1)),cos(latlongmat(i-1,1))*sin(latlongmat(i,1))-
end

windfrom = atan2(-U_route, -V_route)*180/pi;  %Wind Direction (degrees)


for i = 1:nlegs % calculate new aircraft heading taking into account drift, the group speed, and
    the wind correction angle (pos to the right)
```

```
53 [ac_heading_corr(:,:,i), GS(:,:,i), windcorrangle(:,:,i)] = driftcorr(ac_heading_orig(i),TAS,
       windfrom(:,:,i),WS(:,:,i)); %corrected heading to stay on course, groundspeed (knots), the
       wind correcting angle in degrees
54 end
55
56 [~,GC] = legs(latlongmat(1:2,1),latlongmat(1:2,2),'gc'); %dist between pts around GC
57
58 alt_delta = [0:10; %hard coded from altitudes we are investigating
59              1,0:9;
60              2:-1:1,0:8;
61              3:-1:1,0:7;
62              4:-1:1,0:6;
63              5:-1:1,0:5;
64              6:-1:1,0:4;
65              7:-1:1,0:3;
66              8:-1:1,0:2;
67              9:-1:1,0:1;
68              10:-1:0]; % (i,j) i is starting alt, j is ending alt
69
70
71
72 dist = sqrt(alt_delta.^2 + GC^2); % assuming a straight line distance between points
73
74 avg_time_leg = mean(mean(dist/TAS)); %avg time across all altitudes to fly 1 leg
75 flt_time = avg_time_leg;
76 for i = 2:nlegs+1
77     flt_time(i,1) = flt_time(i-1) + avg_time_leg; %sum the time at each leg
78 end
79 timestep_use = round(flt_time); %round to nearest integer and use that hour of data
80
81 %% Create Network and Determine Optimal Path
82
83
84 for latlong = 1:nlegs %Calculate the time to traverse each arc based on ground speed and the
       distance between them
85     for row = 1:size(alt_delta,1)
86         for col = 1:size(alt_delta,1)
87             alpha_i = 24+row;
88             alpha_j = 24+col;
89             time_TAS=dist(row,col)/TAS; %time in hours to fly distance without winds
90             time_GS=dist(row,col)/GS(timestep_use(row)+1,col,latlong); %time in hours to fly
                   distance with winds
91             dist_leg = dist(row,col)*time_GS/time_TAS; %ratio to determine equivalent distance
                   of fuel used with winds in NM
92             if row == col
93                 fuel{latlong}(row,col) = FuelCalc('cruis',AC, alpha_i, omega, PW, dist_leg);
94             elseif row < col
95                 fuel_i = FuelCalc('climb',AC, alpha_i, omega);
96                 fuel_j = FuelCalc('climb',AC, alpha_j, omega);
97                 fuel{latlong}(row,col) = abs(fuel_j - fuel_i);
98                 dist_climb_i = Climb_reg_dist(1,AC) + Climb_reg_dist(2,AC)*alpha_i +
                       Climb_reg_dist(3,AC)*alpha_i^2 + Climb_reg_dist(4,AC)*alpha_i^3 +
                       Climb_reg_dist(5,AC)*omega + Climb_reg_dist(6,AC)*omega^2 +
                       Climb_reg_dist(7,AC)*omega^3 + 10^(-6)*Climb_reg_dist(8,AC)*alpha_i^2*omega^3
```

```
                    + 10^(-6)*Climb_reg_dist(9,AC)*alpha_i^2*omega^3; %determine distance to climb
99                  dist_climb_j = Climb_reg_dist(1,AC) + Climb_reg_dist(2,AC)*alpha_j +
                        Climb_reg_dist(3,AC)*alpha_j^2 + Climb_reg_dist(4,AC)*alpha_j^3 +
                        Climb_reg_dist(5,AC)*omega + Climb_reg_dist(6,AC)*omega^2 +
                        Climb_reg_dist(7,AC)*omega^3 + 10^(-6)*Climb_reg_dist(8,AC)*alpha_j^2*omega^3
                        + 10^(-6)*Climb_reg_dist(9,AC)*alpha_j^2*omega^3; %determine distance to climb
100                 dist_climb =  dist_climb_j - dist_climb_i;
101                 if dist_climb < dist_leg %if climb dist is < dist of leg, calculate fuel consumed
                        on remaining dist as cruise
102                     fuel{latlong}(row,col) = fuel{latlong}(row,col) + FuelCalc('cruis',AC,
                            alpha_i, omega, PW, dist_leg-dist_climb);
103                 elseif dist_climb > dist_leg %if climb dist is > dist of leg, only use a fraction
                        of the total fuel for that climb
104                     fuel{latlong}(row,col) = fuel{latlong}(row,col)*dist_leg/dist_climb;
105                 end
106             else
107                 fuel_i = FuelCalc('descd',AC, alpha_i, omega);
108                 fuel_j = FuelCalc('descd',AC, alpha_j, omega);
109                 fuel{latlong}(row,col) = abs(fuel_j - fuel_i);
110                 dist_descd_i = Descend_reg_fuel(1,AC) + Descend_reg_fuel(2,AC)*omega +
                        Descend_reg_fuel(3,AC)*omega^2 + Descend_reg_fuel(4,AC)*alpha_i +
                        Descend_reg_fuel(5,AC)*alpha_i*omega;
111                 dist_descd_j = Descend_reg_fuel(1,AC) + Descend_reg_fuel(2,AC)*omega +
                        Descend_reg_fuel(3,AC)*omega^2 + Descend_reg_fuel(4,AC)*alpha_j +
                        Descend_reg_fuel(5,AC)*alpha_j*omega;
112                 dist_descd =  dist_descd_j - dist_descd_i;
113                 if dist_descd < dist_leg %if descend dist is < dist of leg, calculate fuel
                        consumed on remaining dist as cruise
114                     fuel{latlong}(row,col) = fuel{latlong}(row,col) + FuelCalc('cruis',AC,
                            alpha_i, omega, PW, dist_leg-dist_descd);
115                 elseif dist_descd > dist_leg %if descend dist is > dist of leg, only use a
                        fraction of the total fuel for that descend
116                     fuel{latlong}(row,col) = fuel{latlong}(row,col)*dist_leg/dist_descd;
117                 end
118             end
119         end
120     end
121 end
122
123 fuels =[];  %initialize
124 strt_node =[]; %initialize
125 end_node =[]; %initialize
126 for i = 2:nlegs-1  %create a vector for time to traverse nodes across cruise waypoints (omit
        nodes 1 and nlegs because we are forcing them to happen at the lowest altitude)
127     fuels = [fuels fuel{i}(:)'];
128
129 end
130 for i = 2:nlegs-1 % create a vector of "from" nodes that correspond to the times vector above
131     strt_node = [strt_node repmat([1+11*(i-1) 2+11*(i-1) 3+11*(i-1) 4+11*(i-1) 5+11*(i-1)
            6+11*(i-1) 7+11*(i-1) 8+11*(i-1) 9+11*(i-1) 10+11*(i-1) 11+11*(i-1)],1,11)];
132 end
133
134 strt_node = [strt_node (11*nlegs)-10:11*nlegs];
135 for i = 23:11*nlegs+1  %create "to" nodes that correspond to the "from" nodes and the times above
```

```
136     end_node = [end_node repmat(i,1,11)];
137 end
138
139
140 W = [fuel{1}(1,:), fuels, fuel{nlegs}(:,1)']; %the arc weights for all nodes (correspond to the
        times to traverse, for now)
141 DG = sparse( [repmat(1,1,11), strt_node],[12:22, end_node], W); %create the network
142 DG =[DG; zeros(size(DG,2)-size(DG,1),size(DG,2))]; % spare arrays need the same number of rows
        and cols (in order to implement graphshortestpath), so add an empty row to make the array
        square
143
144 [optimal_value,path,pred] = graphshortestpath(DG,1,(11*nlegs) + 1);
145 %optimal_value %in kLbs
146
147 end
```

C Fuel Regression Equations

Code/FuelCalc.m

```
1  function fuel_consumed = FuelCalc(eq,AC, alpha, omega, PW, dist)
2  load RegressionCoeffs.mat %load Beta's from Reiman regression models
3
4  if eq == 'climb'
5  % fuel to climb in Klbs
6  fuel_consumed = Climb_reg_fuel(1,AC) + Climb_reg_fuel(2,AC)*alpha + Climb_reg_fuel(3,AC)*alpha^2
       + Climb_reg_fuel(4,AC)*alpha^3 + Climb_reg_fuel(5,AC)*omega + Climb_reg_fuel(6,AC)*omega^2 +
       Climb_reg_fuel(7,AC)*omega^3 + 10^(-6)*Climb_reg_fuel(8,AC)*alpha^2*omega^3 +
       10^(-6)*Climb_reg_fuel(9,AC)*alpha^2*omega^3;
7
8  elseif eq == 'descd'
9  % fuel to descend in Klbs
10 fuel_consumed = Descend_reg_fuel(1,AC) + Descend_reg_fuel(2,AC)*omega +
       Descend_reg_fuel(3,AC)*omega^2 + Descend_reg_fuel(4,AC)*alpha +
       Descend_reg_fuel(5,AC)*alpha*omega;
11
12 else
13     % fuel to cruise in Klbs
14     OW = PayloadAssumptions(1,AC); % operating weight
15     FRC = PayloadAssumptions(5,AC); % reserve/contingency fuel weight
16     FAH = PayloadAssumptions(6,AC) + PayloadAssumptions(7,AC); %alternate/holding fuel weight
17     A = SpecRange_reg(5,AC)/3;
18     B = (SpecRange_reg(4,AC)/2) + SpecRange_reg(5,AC)*(OW + FRC + FAH + PW) +
           (SpecRange_reg(6,AC)/2)*alpha;
19     C = SpecRange_reg(1,AC) + SpecRange_reg(2,AC)*alpha + SpecRange_reg(3,AC)*alpha^2 +
           SpecRange_reg(4,AC)*(OW+FRC+FAH+PW)+SpecRange_reg(5,AC)*((OW+FRC+FAH+PW)^2) +
           SpecRange_reg(6,AC)*alpha*(OW+FRC+FAH+PW);
20     D = -dist;
21     %fuel_consumed = -B/(3*A) -
           1/(3*A)*((1/2)*(2*B^3-9*A*B*C+27*A^2*D+sqrt((2*B^3-9*A*B*C+27*A^2*D)^2-4*(B^2-3*A*C)^3))^(1/3))
           -
           1/(3*A)*((1/2)*(2*B^3-9*A*B*C+27*A^2*D-sqrt((2*B^3-9*A*B*C+27*A^2*D)^2-4*(B^2-3*A*C)^3))^(1/3))
```

```
22      commonterm1 = 2*B^3 - 9*A*B*C + 27*A^2*D;
23      commonterm2 = 4*(B^2 - 3*A*C)^3;
24      cubterm1 = ((1/2)*(commonterm1 + sqrt(commonterm1^2 - commonterm2)))^(1/3);
25      cubterm2 = ((1/2)*abs(commonterm1 - sqrt(commonterm1^2 - commonterm2)))^(1/3) ;
26      cubterm2 = sign(commonterm1 - sqrt(commonterm1^2 - commonterm2))*cubterm2;  %b/c matlab
            yields a complex number
27      fuel_consumed = -B/(3*A) - (1/(3*A))*cubterm1 - (1/(3*A))*cubterm2;
28  end
29  end
```

Appendix C. Matlab Code: The Models

A Deterministic Model

Code/SolveRoutes_Det.m

```
function [optimal_value, path] = SolveRoutes_Det(latlongs_routes,
    route_num,strt_trial,stepsize,num_trial, TAS, U_interp, V_interp,lat,lon)
model = 'D'; %using deterministic data

sol_mat = zeros(num_trial, 3); %ensemble number | nlegs | optimal value
nlegs = strt_trial:stepsize:num_trial*strt_trial;
num_mem = 1;
sol_mat = [];


%based on which route analyzing (route_num= 1,2,3,4,5)
    LatA = latlongs_routes(route_num,1);
    LongA = latlongs_routes(route_num,2);
    LatB = latlongs_routes(route_num,3);
    LongB = latlongs_routes(route_num,4);

for i = 1:num_trial
    path_temp = zeros(20,nlegs(i) +1);
    sol_mat_temp = [];

latlongmat = gcwaypts(LatA,LongA,LatB,LongB,nlegs(i));
latlongmat(:,2) = mod(latlongmat(:,2),360);

    [optimal_value, path, DG] = WindCalcs_26Oct(nlegs(i), 1, latlongmat, lat, U_interp, V_interp,
        lon, TAS,model,num_mem);
    sol_mat_temp(1, 1) = 1;
    sol_mat_temp(1, 2) = nlegs(i);
    sol_mat_temp(1, 3) = optimal_value;
    path_temp(1,:) = path;
    path_mat(:,1:nlegs(i)+1,i) = path_temp;
    sol_mat = [sol_mat; sol_mat_temp];
end
end
```

B Homan Model

Code/SolveRoutes_Mean.m

```
function [optimal_value, path] = SolveRoutes_Mean(latlongs_routes,
    route_num,strt_trial,stepsize,num_trial, TAS, U_interp, V_interp,lat,lon)
model = 'E'; %using ensemble data
num_mem = 20;

%based on which route analyzing
```

```
6       LatA = latlongs_routes(route_num,1);
7       LongA = latlongs_routes(route_num,2);
8       LatB = latlongs_routes(route_num,3);
9       LongB = latlongs_routes(route_num,4);
10
11  sol_mat = zeros(num_trial, 3); %ensemble number | nlegs | optimal value
12  nlegs = strt_trial:stepsize:num_trial*strt_trial;
13  sol_mat = [];
14  for i = 1:num_trial
15      path_temp = zeros(20,nlegs(i) +1);
16      sol_mat_temp = [];
17
18      latlongmat = gcwaypts(LatA,LongA,LatB,LongB,nlegs(i));
19      latlongmat(:,2) = mod(latlongmat(:,2),360);
20      mean_U = mean(U_interp,2);
21      mean_V = mean(V_interp,2);
22      [optimal_value, path, DG] = WindCalcs_26Oct(nlegs(i), 1, latlongmat, lat, mean_U, mean_V,
            lon, TAS, model,num_mem);
23      sol_mat_temp(1, 1) = 1;
24      sol_mat_temp(1, 2) = nlegs(i);
25      sol_mat_temp(1, 3) = optimal_value;
26
27  end
28  end
```

C IID Model

Code/SolveRoutes_IID.m

```
1  function [path_mat,optimal_vals,unique_routes, num_routes, cost] =
       SolveRoutes_IID(latlongs_routes, route_num,strt_trial,stepsize,num_trial, TAS, U_interp,
       V_interp,lat,lon)
2  model = 'E'; %using ensemble data
3  num_mem = 20;
4
5   %based on which route analyzing
6       LatA = latlongs_routes(route_num,1);
7       LongA = latlongs_routes(route_num,2);
8       LatB = latlongs_routes(route_num,3);
9       LongB = latlongs_routes(route_num,4);
10
11  sol_mat = zeros(num_trial, 3); %ensemble number | nlegs | optimal value
12  nlegs = strt_trial:stepsize:num_trial*strt_trial;
13  mem = 1:num_mem; % # of ensemble member to analyze
14  sol_mat = [];
15  for i = 1:num_trial
16      path_temp = zeros(20,nlegs(i) +1);
17      sol_mat_temp = [];
18      for j = 1:num_mem
19  latlongmat = gcwaypts(LatA,LongA,LatB,LongB,nlegs(i));
20  latlongmat(:,2) = mod(latlongmat(:,2),360);
21
```

```
22 [optimal_value, path, DG] = WindCalcs_26Oct(nlegs(i), mem(j), latlongmat, lat, U_interp,
       V_interp, lon, TAS);
23 sol_mat_temp(j, 1) = j;
24 sol_mat_temp(j, 2) = nlegs(i);
25 sol_mat_temp(j, 3) = optimal_value;
26 path_temp(j,:) = path;
27     end
28     path_mat(:,1:nlegs(i)+1,i) = path_temp;
29     sol_mat = [sol_mat; sol_mat_temp];
30 end
31 optimal_vals = sol_mat(:,3);
32
33 %%
34 ncol = 0;
35 all_unique_routes = [];
36 for l = 1:num_trial
37         unique_routes = unique(path_mat(:,:,l),'rows'); %determine which paths are unique across
               ensemble members for each val of nlegs
38        if num_trial > 1
39         unique_routes(:,l*stepsize+2:end) = []; % remove 0's from nlegs being different
40        end
41         [num_routes,~] = size(unique_routes); % determine the number of unique paths
42         latlongmat = gcwaypts(LatA,LongA,LatB,LongB,nlegs(l));
43         latlongmat(:,2) = mod(latlongmat(:,2),360);
44     for k = 1:num_routes
45         for i = 1:20
46            cost_vect = [];
47            [~, ~, DG] = WindCalcs_26Oct(nlegs(l), mem(i), latlongmat, lat, U_interp, V_interp,
                   lon, TAS, num_mem);
48             for j = 2:nlegs(l)+1
49                 cost_vect = [cost_vect; full(DG(unique_routes(k,j-1),unique_routes(k,j)))];
50             end
51            cost((20*(k-1)+i),ncol+1:ncol+2) = [sum(cost_vect) k];
52         end
53     end
54     num_routes_vec(l) = num_routes;
55     [~,ncol] = size(cost);
56
57 end
58 end
```

Bibliography

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, 1 edition, 1993. ISBN 013617549X. URL http://www.citeulike.org/group/328/article/486019.

2. Steve Altus. Effective Flight Plans Can Help Airlines Economize. *AERO Magazine*, pages 27–30, 2009. URL http://www.boeing.com/commercial/aeromagazine/articles/qtr_03_09/article_08_1.html.

3. AMC/A3W. WX White Paper - Draft 1. amc.a3w@scott.af.mil, 2005.

4. Jeffrey L. Anderson. A Method for Producing and Evaluating Probabilistic Forecasts from Ensemble Model Integrations. *Journal of Climate*, 9:1518–1530, 1996. URL https://journals.ametsoc.org/doi/pdf/10.1175/1520-0442%281996%29009%3C1518%3AAMFPAE%3E2.0.CO%3B2.

5. Lee J. Bain and Max. Engelhardt. *Introduction to Probability and Mathematical Statistics*. PWS-KENT Pub, 1992. ISBN 0534380204. URL https://books.google.com/books/about/Introduction_to_Probability_and_Mathemat.html?id=MkFRIAAACAAJ.

6. Dimitri P. Bertsekas and John N. Tsitsiklis. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research*, 16(3):580–595, 1991. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3690040.

7. Garrison Boone. Personal Interview, 2018.

8. Neill E Bowler, Alberto Arribas, and Kenneth R Mylne. The Benefits of Multi-Analysis and Poor-Man's Ensembles. URL http://research.metoffice.gov.uk/research/nwp/publications/papers/technical_reports/reports/505.pdf.

9. Sarah Burke. Confidence Intervals for the Median and Other Percentiles Best Practice. Technical report, Air Force Institute of Technology, STAT T&E COE, 2016. URL https://www.afit.edu/stat/statcoe_files/CI_for_Population_Median.pdf.

10. A. Charnes and W. W. Cooper. Chance-Constrained Programming. *Management Science*, 6(1):73–79, oct 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.73. URL http://pubsonline.informs.org/doi/abs/10.1287/mnsc.6.1.73.

11. Robert Davis and Armand Prieditis. The Expected Length of a Shortest Path. *Information Processing Letters*, 46(3):135–141, jun 1993. ISSN 00200190. doi: 10.1016/0020-0190(93)90059-I. URL http://linkinghub.elsevier.com/retrieve/pii/002001909390059I.

12. DoD. C-17 Globemaster III, 2015. URL http://www.amc.af.mil/About-Us/Fact-Sheets/Display/Article/977489/c-17-globemaster-iii/.

13. F. Anthony Eckel, Jeffrey G. Cunningham, and Dale E. Hetke. Weather and the Calculated Risk: Exploiting Forecast Uncertainty for Operational Risk Management. *Air and Space Power Journal*, 22(1):71–83, mar 2008. URL http://go.galegroup.com/ps/anonymous?id=GALE%7CA179736758{&}sid=googleScholar{&}v=2.1{&}it=r{&}linkaccess=fulltext{&}issn=1555385X{&}p=AONE{&}sw=w{&}authCount=1{&}isAnonymousEntry=true.

14. ECMWF. ECMWF Ensemble Weather Forecasting, 2016. URL https://www.ecmwf.int/sites/default/files/medialibrary/2017-03/ecmwf-fact-sheet-ensemble-forecasting.pdf.

15. Y. Y. Fan, R. E. Kalaba, and J. E. Moore. Shortest Paths in Stochastic Networks with Correlated Link Costs. *Computers & Mathematics with Applications*, 49(9-10):1549–1564, may 2005. doi: 10.1016/J.CAMWA.2004.07.028. URL http://www.sciencedirect.com/science/article/pii/S0898122105001689.

16. Robert Gall, David McCarren, and Fred Toepfer. Deterministic vs. Ensemble Forecasts: The Case from Sandy. *FOCUS*, 3(2), 2013. URL http://www.apectyphoon.org/sdt175/img/img/3860/Newsletter_July_2013/Deterministic_vs_Ensemble_Forecasting_The_case_from_Sandy.pdf.

17. Tilmann Gneiting, Adrian E Raftery, Anton H Westveld III, and Tom Goldman. Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation. *Monthly Weather Review*, 133(5):1098–1118, 2005. doi: 10.1175/MWR2904.1. URL https://doi.org/10.1175/MWR2904.1.

18. Kelly J. Hayhurst and Douglas R. Shier. A Factoring Approach for the Stochastic Shortest Path Problem. *Operations Research Letters*, 10(6):329–334, aug 1991. ISSN 0167-6377. doi: 10.1016/0167-6377(91)90005-A. URL http://www.sciencedirect.com/science/article/pii/016763779190005A.

19. B. P. Heseltine. Contemporary Issues. *Air Force Journal of Logistics*, 31(4):29–37, 2008.

20. Haley A. Homan. Comparison of Ensemble Mean and Deterministic Forecasts For Long-Range Airlift Fuel Planning. Master's thesis, Air Force Institute of Technology, 2014.

21. Eric Hughes. Data Analysis using MATLAB and NCTOOLBOX. http://polar.ncep.noaa.gov/waves/examples/usingmatlab.shtml, 2017.

22. Patrick Jaillet, Jin Qi, and Melvyn Sim. Routing Optimization Under Uncertainty. *Operations Research*, 64(1):186–200, 2016. doi: 10.1287/opre.2015.1462. URL http://dx.doi.org/10.1287/opre.2015.1462.

23. L Jensen, JR Hansman, and J Venuti. Commercial Airline Altitude Optimization Strategies for Reduced Cruise Fuel Consumption. *14th AIAA Aviation*, 2014. URL https://arc.aiaa.org/doi/pdfplus/10.2514/6.2014-3006.

24. Xiaoyu Ji. Models and Algorithm for Stochastic Shortest Path Problem. *Applied Mathematics and Computation*, 170(1):503–514, 2005. doi: 10.1016/j.amc.2004.12.015. URL http://www.sciencedirect.com/science/article/pii/S0096300304009646.

25. Joint Working Group on Forecast Verification Research (JWGFVR). Report to 5th Joint Scientific Committee meeting of the World Weather. Technical report, 2012. URL https://www.wmo.int/pages/prog/arep/wwrp/new/documents/WWRP_JSC5_Doc3_6_JWGFVR.pdf.

26. Candy Knight. AMC Continues to Pursue Fuel Efficiency Initiatives, 2017. URL http://www.amc.af.mil/News/Article-Display/Article/1204903/amc-continues-to-pursue-fuel-efficiency-initiatives/.

27. T. N. Krishnamurti, C. M. Kishtawal, Zhan Zhang, Timothy LaRow, David Bachiochi, Eric Williford, Sulochana Gadgil, and Sajani Surendran. Multimodel Ensemble Forecasts for Weather and Seasonal Climate. *Journal of Climate*, 13(23):4196–4216, dec 2000. ISSN 0894-8755. doi: 10.1175/1520-0442(2000)013⟨4196:MEFFWA⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0442%282000%29013%3C4196%3AMEFFWA%3E2.0.CO%3B2.

28. V. G. Kulkarni. Shortest paths in Networks with Exponentially Distributed Arc Lengths. *Networks*, 16(3):255–274, 1986. ISSN 00283045. doi: 10.1002/net.3230160303. URL http://doi.wiley.com/10.1002/net.3230160303.

29. Baoding Liu. Dependent-Chance Programming: A Class of Stochastic Optimization. *Computers & Mathematics with Applications*, 34(12):89–104, 1997. URL https://ac.els-cdn.com/S089812219700237X/1-s2.0-S089812219700237X-main.pdf?_tid=3b60a564-d384-11e7-abf8-00000aab0f01{&}acdnat=1511795234_8d7a408870950e08843c34ad32ca0df1.

30. Baoding Liu. Uncertain programming. In *Studies in Fuzziness and Soft Computing*, volume 239, pages 111–128. Physica, Heidelberg, 2009. ISBN 9783540894834. doi: 10.1007/978-3-540-89484-1. URL http://link.springer.com/10.1007/978-3-7908-1781-2_24.

31. MathWorks. MATLAB version 8.5.0.197613 (R2016a), 2016.

32. S. F. Milton. Practical Extended-Range Forecasting Using Dynaminal Models. *The Meterological Magazine*, pages 221–233, nov 1990. URL https://digital.nmla.metoffice.gov.uk/download/file/sdb%3AdigitalFile%7C6481f0a6-5ad0-4317-8c63-0a727902208f/.

33. John M. Mirtich. Cost Index Flying. Master's thesis, Air Force Institute of Technology, 2011.

34. Ishwar Murthy and Sumit Sarkar. A Relaxation-Based Pruning Technique for a Class of Stochastic Shortest Path Problems. *Transportation Science*, 30(3), 1996. doi: 10.1287/trsc.30.3.220. URL https://doi.org/10.1287/trsc.30.3.220.

35. U.S. Department of Commerce National Centers for Environmental Prediction, National Weather Service, NOAA. NCEP GFS 0.25 Degree Global Forecast Grids Historical Archive, 2015. URL https://doi.org/10.5065/D65D8PWK.

36. National Oceanic and Atmospheric Administration (NOAA). NOAA's National Weather Service - Glossary, 2017. URL http://forecast.weather.gov/glossary.php?letter=e.

37. Hok K. Ng, Banavar Sridhar, and Shon Grabbe. Optimizing Aircraft Trajectories with Multiple Cruise Altitudes in the Presence of Winds. *Journal of Aerospace Information Systems*, 11(1):35–47, jan 2014. ISSN 2327-3097. doi: 10.2514/1.I010084. URL http://arc.aiaa.org/doi/10.2514/1.I010084.

38. NOAA. Numerical Weather Prediction, 2017. URL https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/numerical-weather-prediction.

39. NOAA. NOAA Operational Model Archive and Distribution System, 2017. URL http://nomads.ncep.noaa.gov/l.

40. Inc On Time Systems. Worldwide Aeronautical Route Planner (WARP), 2017. URL http://www.otsys.com/warp.php.

41. T. N. Palmer, Č. Branković, F. Molteni, and S. Tibaldi. Extended-Range Predictions with ECMWF Models: Interannual Variability in Operational Model Integrations. *Quarterly Journal of the Royal Meteorological Society*, 116(494): 799–834, jul 1990. doi: 10.1002/qj.49711649403. URL http://doi.wiley.com/10.1002/qj.49711649403.

42. Adam D Reiman. Enterprise Analysis of Strategic Airlift to Obtain Competitive Advantage Through Fuel Efficiency, 2014. URL http://www.dtic.mil/docs/citations/ADA609511.

43. Deniz Sarioz and Victor Dan. *Journal of Computing Sciences in Colleges.*, volume 16. Consortium for Computing Sciences in Colleges, 2001. URL http://dl.acm.org/citation.cfm?id=378757.

44. B. Schlining, R. Signell, and A. Crosby. nctoolbox, 2009. URL https://github.com/nctoolbox/nctoolbox.

45. Zoltan Toth and Eugenia Kalnay. Ensemble Forecasting at NMC: The Generation of Perturbations. *Bulletin of the American Meteorological Society*, 74 (12):2317–2330, dec 1993. doi: 10.1175/1520-0477(1993)074⟨2317:EFANTG⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0477%281993%29074%3C2317%3AEFANTG%3E2.0.CO%3B2.

46. M. Steven Tracton and Eugenia Kalnay. Operational Ensemble Prediction at the National Meteorological Center: Practical Aspects. *Weather and Forecasting*, 8(3):379–398, sep 1993. doi: 10.1175/1520-0434(1993)008⟨0379:OEPATN⟩2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0434%281993%29008%3C0379%3AOEPATN%3E2.0.CO%3B2.

47. Akshay Upadhyay. Formula to Find Bearing or Heading angle between two points: Latitude Longitude, 2015. URL http://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/.

48. Gang Yu and Jian Yang. On the Robust Shortest Path Problem. *Computers & Operations Research*, 25(6):457–468, 1998. ISSN 03050548. doi: 10.1016/S0305-0548(97)00085-3. URL http://www.sciencedirect.com/science/article/pii/S0305054897000853.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22–03–2018	Master's Thesis	Sept 2017 — Mar 2018

4. TITLE AND SUBTITLE

SHORTEST PATH ACROSS STOCHASTIC NETWORK WITH CORRELATED RANDOM ARCS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Stephanie M.Boone, 1st Lt, U.S. Air Force

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENC-MS-18-M-109

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

AMC Weather Operations and Plans Branch
402 Scott Drive, Unit 3A1
Scott AFB, IL 62225-5302
COMM 618-229-3636
Email:frederick.wirsing@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AMC/A3AW

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper introduces a new approach to identify the shortest path across a stochastic network with correlated random arcs utilizing nonparametric samples of arc lengths. This approach is applied to find optimal aircraft routes that minimize expected fuel consumption for a given airspeed utilizing predicted wind output from NWP ensemble models. Results from this new methodology are then compared to the current fuel minimization route planning method that utilizes deterministic NWP wind data for arc lengths. Comparisons are also made to other previously proposed alternative fuel minimization methodologies that utilize mean and median wind data calculated from NWP ensemble wind data.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col A. Geyer, AFIT/ENC
U	U	U	U	81	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565, x4584; Andrew.Geyer@afit.edu