3-21-2019

Time Series Analysis of Stochastic Networks with Correlated Random Arcs

Brendon T. Sands

Time Series Analysis of Stochastic Networks with Correlated Random Arcs

THESIS

Brendon T. Sands, 2nd Lt, USAF

AFIT-ENS-MS-19-M-147

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-19-M-147

TIME SERIES ANALYSIS OF STOCHASTIC NETWORKS WITH RANDOM CORRELATED 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

Brendon T. Sands, B.S.

2nd Lt, USAF

March 4, 2019

AFIT-ENS-MS-19-M-147

TIME SERIES ANALYSIS OF STOCHASTIC NETWORKS WITH RANDOM CORRELATED ARCS

THESIS

Brendon T. Sands, B.S.
2nd Lt, USAF

Committee Membership:

Lt Col A.J. Geyer
Chair

Dr. R. Hill
Reader

AFIT-ENS-MS-19-M-147

Abstract

While modern day weather forecasting is not perfect, there are many benefits given by the multitude and variety of predictive models. In the interest of routing airplanes, this paper uses time series analysis on successive weather forecasts to predict the optimal path and fuel burn of wind-based, fuel-burn networks with stochastic correlated arcs. Networks are populated with either deterministic or ensemble-based weather data, and the two data sources with and without time series analysis are compared. Methods were compared by fuel burn prediction accuracy and ability to predict a future optimal path. Of the four options, the ensemble-based methods were on average the least accurate. Using time series analysis on ensemble data gave a nominal change in correct future path prediction and an increase in fuel burn prediction accuracy. The deterministic method gave the most accurate results but the worst correct future path prediction rate. Time series analysis on deterministic data had a marginal decrease in accuracy but the highest correct future path prediction rate.

Acknowledgements

I would like to thank all of my friends and family for their continued support throughout this analysis. I would also like to thank my faculty research advisor, Lt Col Geyer, and committee member, Dr. Hill, for their guidance and support throughout this process.

Brendon T. Sands

Table of Contents

List of Figures

List of Tables

TIME SERIES ANALYSIS OF STOCHASTIC NETWORKS WITH RANDOM CORRELATED ARCS

I. Introduction

1.1 Background

Despite the Department of Defense's massive budget, there is always a need to allocate funds as efficiently as possible to complete the agency's mission. For the 2019 fiscal year, the United States Air Force alone requested a budget of $156 billion [1]. In the Air Force, this money is allocated to different major commands (MAJCOMs) to undertake and accomplish different subsections of the Air Force's total mission. For example, Air Mobility Command (AMC) specializes in "worldwide cargo and passenger delivery" among other things [2]. In 2017, AMC consumed more than half of the Air Force's $6.8 billion aviation fuel budget [3]. With AMC using the majority of the Air Force's annual reported 2 billion gallon supply of aviation fuel, any small increase in efficiency will have a large impact throughout this MAJCOM [4].

1.2 Overview

There are numerous ways to increase fuel efficiency that can be applied to AMC's mission and the Air Force as a whole. This work focuses on improving flight planning to better predict fuel burn by routing aircraft away from predicted headwinds. In particular, this research focuses on flight planning for the C-17, a cargo plane that is a staple for AMC's transportation mission. Better fuel burn predictions will result in lighter planes as pilots can take less contingency fuel. Further, avoiding

headwinds is effectively traveling a shorter distance for an airframe.

Current weather models can be broken into two distinct classes: Deterministic and Ensemble. Deterministic models are those that are composed by a single model, while multiple deterministic models comprise an ensemble model [5]. An Ensemble model initializes each of its members with slightly different parameter values generating a forecast with a spread of results. This spread can give better insight into the range of a forecast than its deterministic counterpart.

Although ensemble models are a weather industry mainstay, AMC still uses deterministic weather forecasts when route planning for future flights. Recent research related to predicting aircraft fuel burn has made a strong case for the departure from deterministic weather models in favor of ensemble weather models. In 2014, Homan [6] used an iterative algorithm to compare predicted fuel burn estimates between ensemble and deterministic weather models across a variety of aircraft and routes. Homan [6] found averaging ensemble forecasts to "generally provide more accurate fuel burn estimates than GFS (Global Forecast System) deterministic model wind forecasts". In recent research, Boone [7] showed ensemble weather forecasts could more frequently find the optimal route for future C-17 flights while also being better at predicting fuel burn.

The remainder of this thesis explores the value of ensemble weather models and time series analysis to the flight planning community. Chapter 2 discusses previous literature on this subject. Chapter 3 describes current practices then frame the problem and possible improvements. Next, Chapter 4 reviews the results. Lastly, Chapter 5 finishes with lessons learned and a conclusion.

II. Literature Review

2.1 Overview

This chapter discusses the stochastic shortest path problem (SPP) with correlated arcs and how it relates to the AMC flight planning problem. The deterministic SPP is a classic network problem that seeks to minimize cost of traversing between nodes of a network. In the stochastic SPP case, arc lengths are uncertain which violates various assumptions of the deterministic SPP. One popular approach to solving the stochastic SPP is the expected shortest path [8]. Further complications of the stochastic SPP arise when arcs in a network are correlated. Boone's [7] recent approach to stochastic SPP with correlated arcs was using independent and identically distributed (IID) ensembles of an entire network to find the optimal path through a future instance of a network. Lastly, the different benefits between ensemble and deterministic weather models are briefly discussed.

2.2 Stochastic SPP

The deterministic SPP is shown in Equation (1) where A is the set of all arcs (i, j), x_{ij} is the binary decision variable represneting the selection of the arc from node i to node j with $1 \leq i,j \leq n$, and c_{ij} is the cost of traveling a particular arc. In scenarios such as this AMC flight planning problem, a network can be created as an outline for a potential flight route. In this sense, the shortest path is the path that minimizes the fuel burn traversing from one end of the network to the other. However, since fuel burn is based on wind, which is stochastic in nature, this formulation is not appropriate.

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

The expected shortest path model attacks the stochastic nature of stochastic SPP by changing the objective function of Equation (1) to minimize an expected value of travel cost across each arc [8]. Each arc's expected value is calculated based off of its respective probability distribution. This formulation is given by Equation (2).

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

Further complications of solving the stochastic SPP arise when both arcs are correlated and the probability distributions of arc lengths are unknown [9]. In certain applications, multiple predictions of a stochastic correlated network exist and can be

used to increase the likelihood of finding the shortest path.

For instance, weather models can be comprised of ensemble members that independently predict future weather [5]. Boone [7] used individual ensemble members to create travel networks across several popular C-17 flight routes and predict the most fuel efficient flight plan. All ensemble networks were solved and a list of unique optimal paths were recorded. Next, each unique optimal path was run through each ensemble network giving an IID sample of each path's true fuel burn. This method, known as the *a posterori* Shortest Path (APSP) was able to identify the future optimal path through a stochastic network with correlated arcs.

2.3 Ensemble vs Deterministic

The comparison between ensemble and deterministic models has been well studied. Keith and Leyton [10] showed ensemble weather models to be better at predicting adverse weather conditions where commercial planes would require extra fuel. Krishnamurti *et al.* [11] found an ensemble model to be the superior forecaster when compared to the deterministic models that composed it in a multitude of applications. However, the benefits of ensemble forecasting are not absolute to all problems and times. While ensemble models showed increased forecasting accuracy for flooding in Venice due to storms 4+ days out, the short term ensemble and deterministic forecasts were comparable [12]. Leonardo and Colle [13] validated an assortment of popular weather models' accuracy in predicting North Atlantic tropical cyclones. They found a deterministic model gave the lowest total track error with several ensembles being statistically comparable. The World Meteorological Organization notes that ensemble models typically produce more reliable forecasts than their deterministic counterparts, but "this is particularly true for forecasts more than 1-3 days ahead" [14].

2.4 Conclusion

As already shown, correlation in arc lengths disrupts the integrity of traditional stochastic SPP solutions. In unique situations, such as weather modeling, different representations of a stochastic network can be used to add context that would not normally be available. Naturally, the benefit of this context can be applied to the AMC problem by using successive ensemble and deterministic weather predictions of a pertinent area.

III. Methodology

3.1 Introduction

This chapter will first focus on current methodology for flight routing in AMC. Next, it will be shown how weather networks can be created to resemble potential flight paths, and all necessary components of the network are described. The current methodology is transposed to the network interpretation of this problem alongside other possible solution methods. These new proposed methods are built using successive predictions and time series analysis. Lastly, the measures for differentiating between methods are introduced.

3.2 Past AMC

AMC currently uses a system known as the Advance Computer Flight Planner (ACFP) to plan future flight routes. The ACFP system plans flight routes based on weather and daily operational considerations. The data intake for the ACFP platform is currently the deterministic GFS model provided by the National Oceanic and Atmospheric Administration (NOAA) [15]. Apart from small adjustments, the networks to be described are of the same form as created by Boone [7].

3.3 AMC Application

Data

The ensemble weather data came from the Global Ensemble Forecasting System (GEFS) model curated by NOAA, composed of 21 ensemble models. Each of the 21 ensemble members had wind data at 14 different pressure levels at each integer intersection of latitudinal and longitudinal coordinates [16]. Of the 14 pressure levels,

only three were inside or near the conventional cruising altitude of a modern airplane with respective altitudes of 20800, 26600, and 44640 ft. The deterministic weather data came from the GFS model. Wind data was also available at the crossing of integer latitude and longitudes but was predicted at 31 pressure levels. Five pressure levels from 23500 ft to 38600 ft were applicable, but only the three pressure levels common to both models surrounding typical C-17 airspace were used to create networks. These three pressure levels were interpolated to 25000, 30000, and 35000 ft for both models, or the typical bounds for a C-17 in cruise. The GFS and GEFS models had weather predictions for each coordinate a week ahead. However, this study did not look at predictions further than five days into the future.

The dataset contained wind data corresponding to five popular C-17 routes: Travis Air Force Base, CA to Honolulu, HI, McChord Air Force Base, WA to Gander, Newfoundland, Shaw Air Force Bace, SC to Rota, Sinapalo, Shaw Air Force Base, SC, to Ramstein, Germany, and Joint Base McGuire Dix Lakehurst, NJ to Spangdahlem, Germany. Each route included all available pressure levels interpolated to a set of unique waypoints. For succinctness each route is referred to by each airport's respective four letter designation:KSUU-PHNL, KTCM-CYQX, KCHS-LERT, KCHS-ETAR, and KWRI-ETAD. The Matlab function *gcwaypts* created a list of evenly spaced waypoints arranged in a great circle track between the starting and ending airports. The number of waypoints on each route was calculated based on the length of the route, with one waypoint occuring every half degree. The interpolation was done using the function *griddata*. *Griddata* used the one degree gridded latitudinal and longitudinal wind data at each pressure level and interpolated the wind components of each pressure level to the route's waypoints using 4-D linear interpolation.

Building the Network

To find the most fuel efficient path between two airports, a network was created based on the waypoints associated with each of the five routes that resembled the flights possible path. To mirror typical flight planning eleven nodes were created at each waypoint, one for each altitude between 25k and 35k feet. Ng *et al.* [17] found fuel savings for initial climbs and final descents to be negligible. So, the first and last nodes were forced to be at 25k feet. Ultimately, a network with n waypoints, where n does not include the starting and ending nodes, would be comprised by $11n + 2$ nodes and $(n-1)11^2 + 22$ arcs. In the network a plane cannot fly to another altitude at the same waypoint, and each arc is directional flowing towards the next waypoint.

Wind data is predicted at various pressure levels across the globe. This wind data takes the form of U and V components in meters per second. A positive U component represents wind blowing to the East while a positive V component represents wind blowing to the North [18]. With wind data at all pressure levels at each of the waypoints along a route, the U and V components were separately interpolated to the eleven altitudes at each waypoint along any given route by iteratively using the *griddata* function. The total wind speed at each waypoint and altitude in meters per second was calculated using the Pythagorean Theorem. The wind direction related to the recently calculated wind speed was calculated using Matlabs *atan2d* function which returns the four-quadrant arctangent of the U and V components. The wind speed was then converted from meters per second to knots by the *convvel* function.

As the C-17 traverses the network, the winds will naturally be changing in flight. Assuming the C-17 maintains its typical cruising speed of 450 knots [19], three out of the five routes take over seven hours to travel from the starting point to the destination. Given weather predictions every six hours, the initial network is interpolated with up to two future predictions of the network to account for changing

winds in flight over the duration of a flight.

In the available dataset, there are five days worth of weather predictions for any given time slot. A true weather forecast is the +0 hour initialization of the GFS model at that hour. Therefore, there are 20 predictions of the true weather generated from 120 hours to 6 hours (+120 to +6) before the actual predicted time. Since each network is interpolated across the flight's duration, the +06 model for a selected time is paired with the +12 model for six hours ahead of the selected time and if necessary the +18 model for twelve hours ahead of the selected time. This interpolation mirrors the data that would be available real time for a flight scheduler. While realistic, this method shrinks the amount of forecasts available for the GFS and GEFS data from 20 to 18. This occurs because in practice at +120 from flight time there would not yet exist the forecasts to needed to interpolate into the future. This is shown in Table 1 where the red indicates unavailable forecasts while the blue indicates unused forecasts.

Next, the azimuth from each waypoint to each following waypoint was calculated using the *azimuth* function. Assuming a constant true air speed (TAS) of 450 knots at all times, the wind-affected ground speed was calculated using the *driftcorr* function in Matlab. The *driftcorr* function takes in the C-17's heading and speed and calculates the corrected ground speed based on the wind speed and direction at every

Table 1. Composition of Complete Forecasts When Interpolating Across Time

			Forecast 1	Forecast 2	...	Forecast 17	Forecast 18			
July 26th 6:00 PM	+120	+114	+108	+106	...	+12	+06	+00		
July 27th 12:00 AM		+120	+114	+108	...	+18	+12	+06	+00	
July 27th 6:00 AM			+120	+114	...	+24	+18	+12	+06	+00

node. Each arc in the network is symbolic of the C-17 flying from one of the eleven possible altitudes of a waypoint to another of the eleven possible altitudes in the successive waypoint. The fuel burn to travel each arc was calculated using Reiman's [20] C-17 regression models. These regression models depend on weight, altitude change, and distance flown. The weight of a C-17 was assumed to be constant at 496,500 lbs. The distance between waypoints was altered based on a ratio of TAS to ground speed (Equation (3)). This effective distance reflects how far a C-17 actually travels in cruise without being affected by wind.

$$Distance_{wind} = Distance \times \frac{TAS}{GroundSpeed} \tag{3}$$

Reiman's regression models have three parts: ascent (Equation (4)), descent (Equation (5)), and cruise (Equation (6)). The ascent and descent models give an estimate on either the fuel burned changing altitudes, or the distance required to change altitude based on which β coefficients are used. The β coefficients associated with each equation's descending and climbing regression equations are found in Tables 2 and 3, respectively.

$$\phi_C = \beta_0 + \beta_1\alpha + \beta_2\alpha^2 + \beta_3\alpha^3 + \beta_4\omega + \beta_5\omega^2 + \beta_6\omega^3 + 10^{-6}\beta_7\alpha^2\omega^3 + 10^{-6}\beta_8\alpha^2\omega^3 \tag{4}$$

$$\phi_D = \beta_0 + \beta_1\omega + \beta_2\omega^2 + \beta_3\alpha + \beta_4\alpha\omega \tag{5}$$

Where:

ϕ_C = Fuel to Climb in Klbs or Distance to Descend 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 Descent Start

Table 2. Descent Regression Terms (ϕ_D)

	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

Table 3. Climb Regression Terms (ϕ_C)

	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

Before calculating the fuel burn of an arc, the distance needed to change altitudes was calculated. If the distance needed to change altitudes for a particular arc was greater than the distance between waypoints, then the arc was discarded. Naturally all arcs that were not discarded had some distance where the plane was cruising between waypoints. The regression model for fuel burn in cruise is shown in Equation (6) with the β coefficients shown in Table 4.

$$\omega_{ff} = -\frac{B}{3A} - \frac{1}{3A}\sqrt[3]{\frac{1}{2}[2B^3 - 9ABC + 27A^2D + \sqrt{(2B^3 - 9ABC + 27A^2D)^2 - 4(B^2 - 3AC)^3}]} - \frac{1}{3A}\sqrt[3]{\frac{1}{2}[2B^3 - 9ABC + 27A^2D - \sqrt{(2B^3 - 9ABC + 27A^2D)^2 - 4(B^2 - 3AC)^3}]} \quad (6)$$

Where:

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

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

Table 4. Cruise (ω_{ff}) Regression Terms

	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

After establishing the fuel required to traverse each arc, the network is solved using the *shortestpath* function. This function solves the shortest path problem as shown in Equation (1).

Different Model Predictions

This research uses four different fuel burn predictions based on the weather data of the GEFS and GFS forecast models. Each of the four methods for predicting fuel burn are based on the data populated from one of the two forecast models. Two out of the four methods use a time series approach to predicting the future. These methods analyze how a path's predicted fuel burn changes over successive model predictions to better predict how the path's predicted fuel burn will behave in the future. A method that mimics current practices is used, along with Boone's [7] proposed method.

While both the GEFS and GFS model have their own version of the world's current true weather, the GFS model initialization is used as the truth model when comparing predictions to real world conditions.

Deterministic

The deterministic method is simply the best last model and imitates current AMC routing practices. This method populates the network described above with the GFS models pth forecast before planning a route. From this data, the network is solved and the shortest path and predicted fuel burn recorded.

Deterministic Time Series

The deterministic time series method is an extension of the deterministic method. For a given set of true weather data, a network is created for every p available successive weather forecasts. As already described there can be a maximum of 18 weather networks created. Each network is solved, and the optimal paths and predicted fuel burns are recorded. There can be not be more unique optimal routes than networks. Next, the optimal paths are reduced to only unique optimal paths. Each of these candidate paths are run through all networks where the fuel burn estimate in each

network is recorded. Time series analysis is used on the fuel burn predictions to both determine the best path and predict the fuel burn of the best path. The path with the lowest time series fuel burn prediction is chosen as the prediction. A prediction interval on the fuel burn is then created using principles from linear regression.

Boone's Method

Unlike the previous two methods, Boone's [7] method utilizes the last available GEFS wind data. For a given time, the 21 ensemble models that comprise the pth GEFS model are used to make 21 different networks. The optimal path is then solved for each network. Each unique optimal path is then run through each of the 21 networks where the fuel burn prediction is recorded. The best path is chosen based on the lowest average fuel burn across each path's ensemble predictions. Lastly, a kernel density estimate discussed later in the paper is created as a prediction interval around the fuel burn.

Time Series Ensemble

The last of the four methods expands Boone's [7] method over an interval of successive forecasts. A total of 21 weather networks, one for each ensemble, are created for each p available GEFS forecasts. Similarly to the deterministic time series, there can be at most 18 sets of 21 ensemble networks. Each network is solved and the unique routes are found. Then, each path is run through every ensemble's network of every times prediction. Time series analysis is used to create a fuel burn prediction of each unique optimal path in all ensembles at the truth time. After all 21 ensemble's time series predictions are calculated, the median time series prediction is used as the fuel burn prediction for a given path. The path with the lowest median time series prediction is used as the best path. Kernel density estimation is then used

to created a prediction interval around the ensemble time series predictions.

Time Series Analysis

Both the ensemble time series and deterministic time series use the same time series model, a second order autoregressive model as described by Bowerman *et al.* [21]. This model is a linear regression model where the next time series value is predicted as a function of the previous two.

$$\hat{y}_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} \tag{7}$$

A second order model was chosen solely based on performance. Autoregressive models of order one through six were compared in a random sample of 30% of the data set with the second order model giving the lowest error on average. Each unique path identified in the deterministic time series prediction has its own model based on its own time series data. Each unique path identified in the ensemble time series prediction has 21 different models, one for each ensemble. After constructing each model, path forecasts are created by setting the future time to $\hat{y}_t$ in Equation (7) and solving.

Prediction Intervals

Prediction intervals are created for both time series methods and Boone's method to give more context about each method's prediction. The deterministic time series prediction interval is a byproduct of being created through linear regression. The prediction interval is slightly larger than a confidence interval since it "depends on both the error from the fitted model and the error associated with future observations" [22]. In Equation (8), μ is the true mean, $\bar{x}$ is the sample mean, t is a t-distribution with $n - p$ degrees of freedom, α is the confidence level, s^2 is the sample variance,

X is the data matrix and x_0 is the row vector containing the numbers multiplied by the β coefficients in Equation (7). In the t-distribution's degrees of freedom, p is the same as the order of the autoregressive model.

$$\mu \in \bar{x} \pm t_{\frac{\alpha}{2},n-p}\sqrt{s^2(1+\mathbf{x_0'}(\mathbf{X'X})^{-1}\mathbf{x_0})} \tag{8}$$

Prediction intervals for the ensemble time series and Boone's method are created using kernel density estimation (KDE). KDE was used as a non-parametric way to compute the probability density functions (PDFs) given multiple predictions for a path in the Boone and time series ensemble methods. KDE gives insight from the predictions without making assumptions about their distribution. For both methods, PDFs were created using the Matlab function *ksdensity*. KDE is shown in Equation (9) where n is the sample size, $(x_1, x_2, ..., x_n)$ are the n samples, $K(x, \cdot)$ is the Kernel function, and h is the bandwidth parameter [23]. The Gaussian kernel, used to calculate the PDFs in *ksdensity* is shown in Equation (10) [24].

$$\hat{f} = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \tag{9}$$

$$K(t) = \frac{1}{\sqrt{2\pi}}\exp^{-(1/2)t^2} \tag{10}$$

Using *quantile* in Matlab, the 2.5% and 97.5% quantiles of the kernel's cumulative density function are calculated. This interval, containing 95% of distribution, is taken as the 95% prediction interval.

Multiple Paths

For the last three methods there is potentially multiple candidate paths to choose from. When this occurs, the path with the lowest prediction is always chosen

as the path to compare against fuel burn predictions from the other methods. Given multiple paths, this is ideal based on how the path is chosen. The lowest average path will be, at worst, statistically comparable to another path.

Methods Over a Planning Window

To further evaluate real world applicability, the methods are evaluated at 96, 72, 48, 24, and 6 hours before takeoff. The number of forecasts available at each period is shown in Table 5. There isn't enough information to create a reliable time series model four days from take off, so fuel burn error and optimal path prediction rate is only calculated for the Boone and deterministic methods at that time period.

Table 5. Composition of Complete Forecasts When Interpolating Across Time

Time Before Takeoff	Forecasts (p)
+96	3
+72	7
+48	11
+24	15
+6	18

Root Mean Square Error

To compare the accuracy of each method across routes, the Root Mean Squared Error (RMSE) was calculated based on each method's prediction. The RMSE is a popular indicator of average model performance [25]. The formula for the RMSE of a prediction is shown in Equation (11).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{11}$$

Where n is the sample size and e_i is the error term of the ith prediction. In this problem, n is the number of instances of wind data and e_i is calculated as the difference between a method's ith predicted fuel burn and the fuel burn generated by the optimal path in the ith truth network.

IV. Analysis

4.1 Introduction

The four methods selected were tested on 139 instances of wind data from July 26th to Dec 5th. Each route is analyzed separately to determine performance in fuel burn as well as path prediction.

4.2 KSUU-PHNL

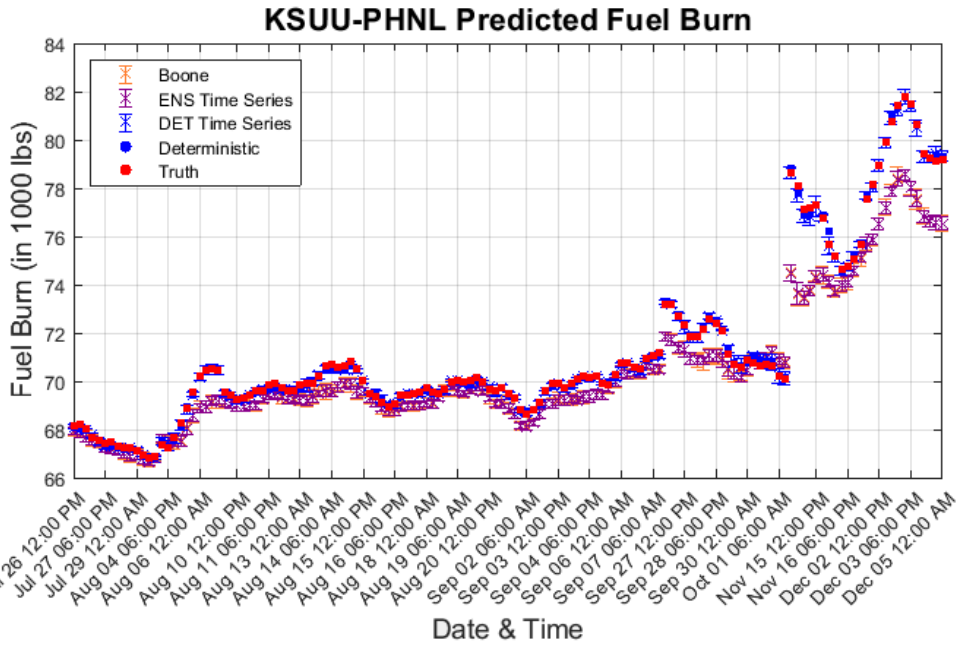

Figure 1. KSUU-PHNL Predicted Fuel Burn

The first route is KSUU-PHNL, a long journey over the Pacific Ocean from California to Hawaii. Each method's predicted fuel burn and prediction interval if applicable for this route is shown in Figure 1. The prediction intervals are shown with an error-bar, where the actual prediction is shown as an X inside the bounds of the error-bar. Each time series method predicts closely to their respective non-time

series counterpart. The deterministic methods outperform the ensemble methods throughout the data set. The associated RMSE's for each method are shown in Figure 2. In regards to fuel burn prediction, six hours before takeoff the deterministic based-models are both almost 800 lbs more accurate than the ensemble-based models. The RMSE based on available prediction data is shown in Figure 3. While the RMSE drops for all four methods with increasing predictions, the deterministic gives the lowest RMSE at all points.

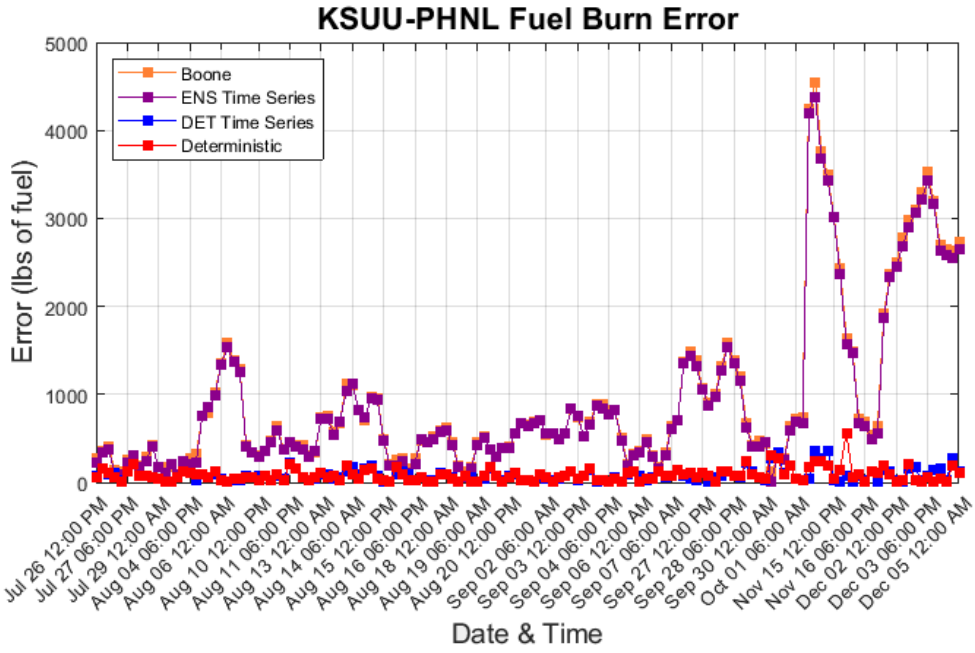

Figure 2. KSUU-PHNL Fuel Burn Prediction Error

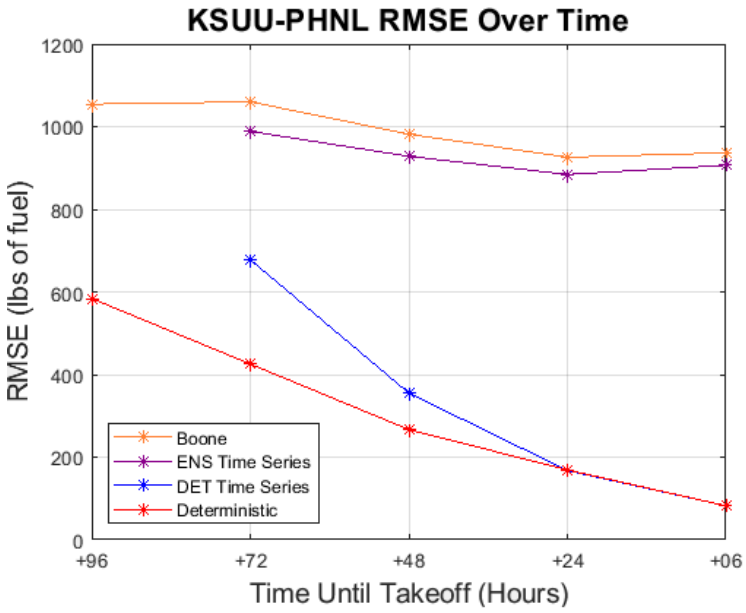

Figure 3. KSUU-PHNL Change In RMSE

Out of the 139 dates, the Boone method predicted the correct future optimal path 71 times while the ensemble time series method predicted correctly 72 times. The deterministic method correctly predicted the optimal path 80 times and the time series deterministic method correctly predicted the optimal path 121 times. Both time series methods improve with successive data. Boone's method improve slightly and the deterministic method decreases slightly over the course of four days.

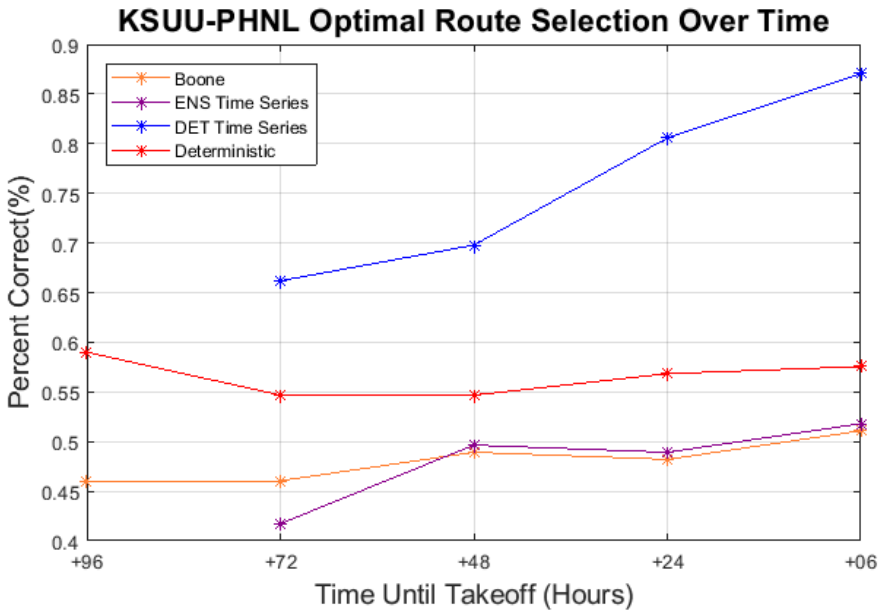

Figure 4. KSUU-PHNL Change In Optimal Path Selection

4.3 KTCM-CYQX

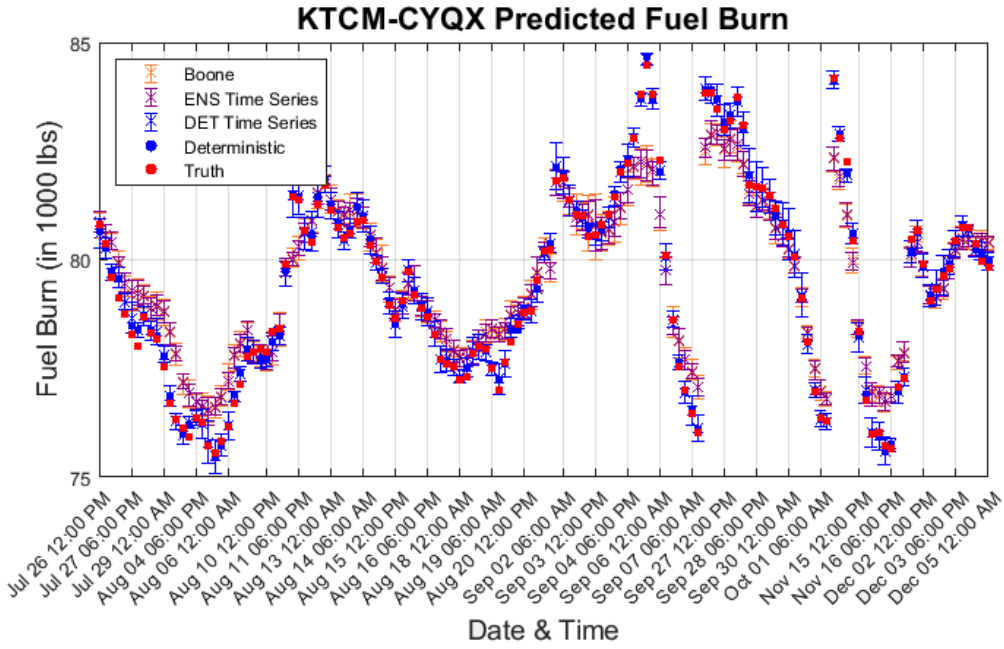

Figure 5. KCTM-CYQX Predicted Fuel Burn

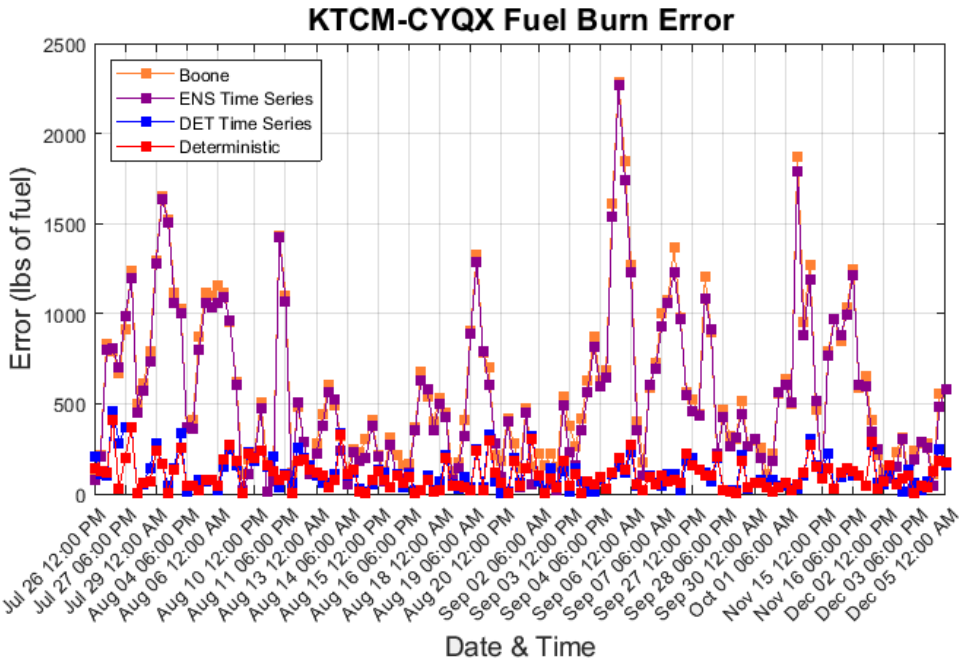

Figure 6. KCTM-CYQX Fuel Burn Prediction Error

The second route analyzed is an international route from the Pacific North West to Canada's East coast. All four methods fluctuate in similar manners as seen in Figure 5, however, the deterministic-based methods consistently predict closer to the true fuel burn. The RMSEs of each time instances are shown in Figure 6. Figure 7 shows that three days prior to takeoff the deterministic time series method has the highest RMSE. Six hours prior to takeoff the the ensemble-based methods generate fuel burn predictions on average 400 lbs worse than the deterministic-based methods. Figure 8 shows that the deterministic time series method gets much better at predicting future optimal paths with newer weather data, while the deterministic, ensemble time series, and Boone's method show little change.

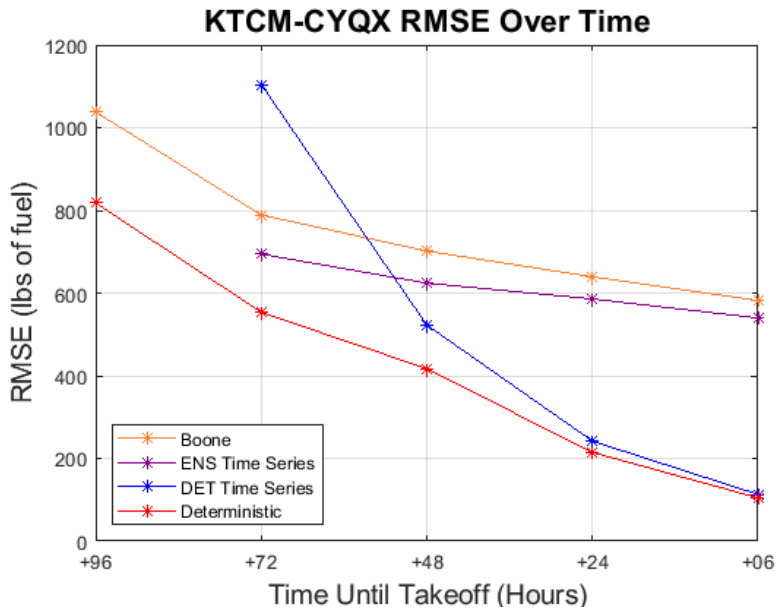

Figure 7. KCTM-CYQX Change In RMSE

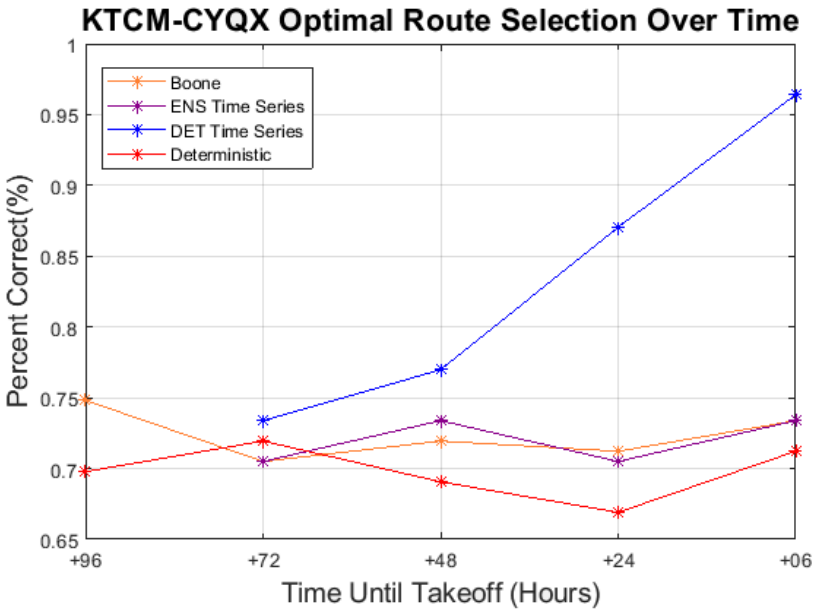

Figure 8. KCTM-CYQX Change In Optimal Path Selection

4.4 KCHS-LERT

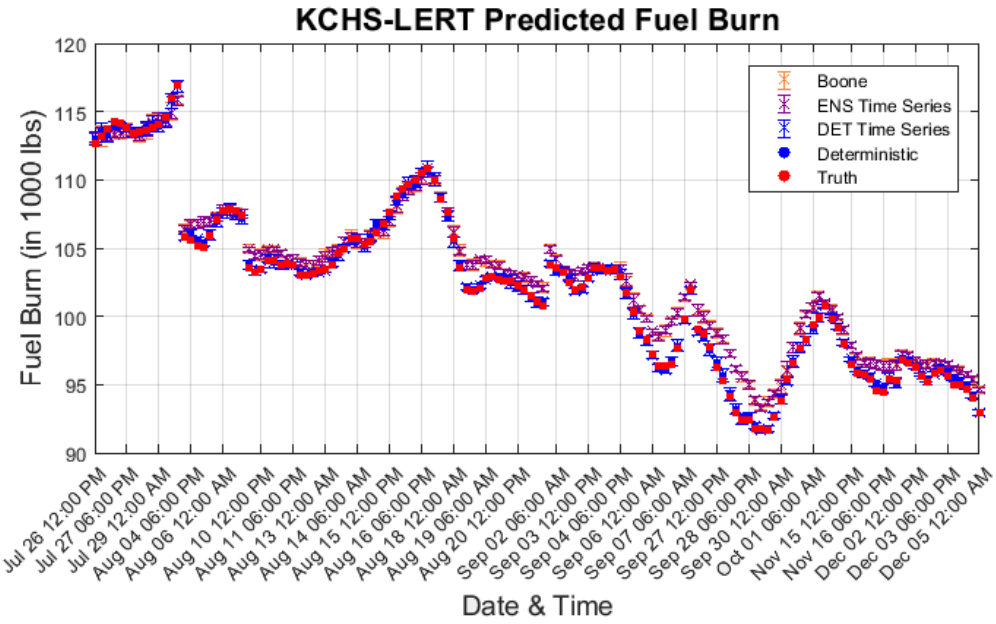

Figure 9. KCHS-LERT Predicted Fuel Burn

The third route is the second longest of the five routes and spans from South Carolina across the Pacific Ocean to the Northern Mariana Islands. The predicted fuel burn is shown in Figure 9 and the RMSE in Figure 10. Figure 11 shows that all four methods show significant improvement with newer weather data and more predictions. Over the course of the planning period, Boone's method and the deterministic method's RMSE is respectively lowered by 674 and 740 lbs. The deterministic method continuously produces the lowest average RMSE. The deterministic, deterministic time series, ensemble time series, and Boone's method respectively predicted 83, 120, 98, and 97 future paths correctly with +6 forecast data.

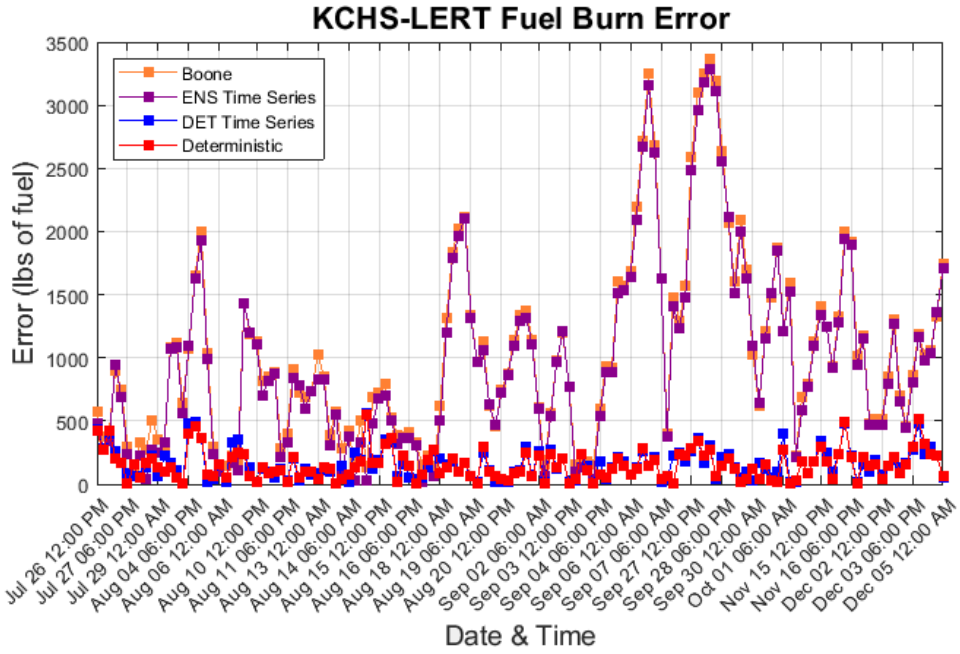

Figure 10. KCHS-LERT Fuel Burn Prediction Error

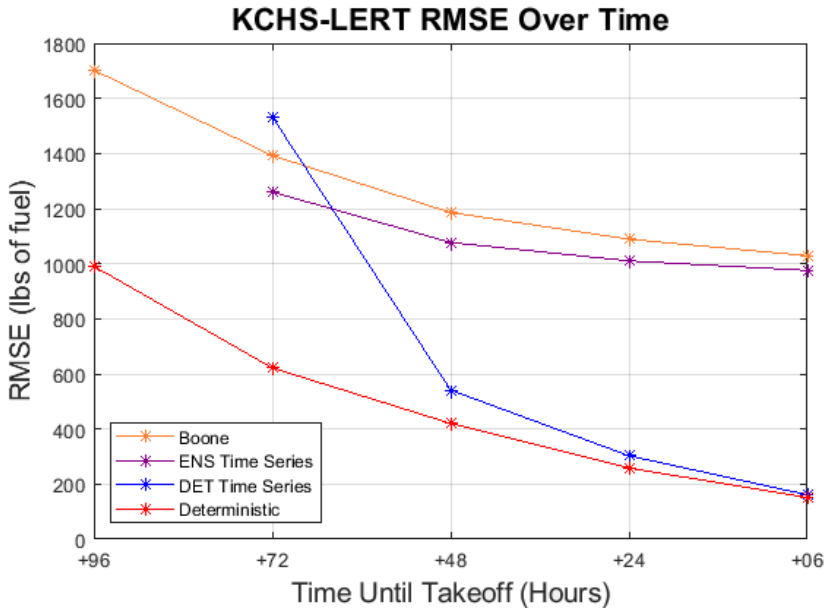

Figure 11. KCHS-LERT Change In RMSE

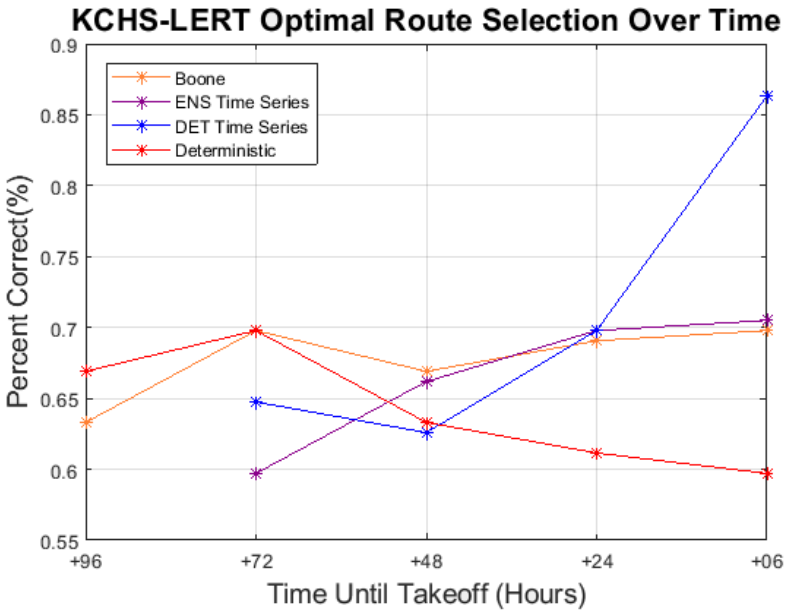

Figure 12. KCHS-LERT Change In Optimal Path Selection

4.5 KCHS-ETAR

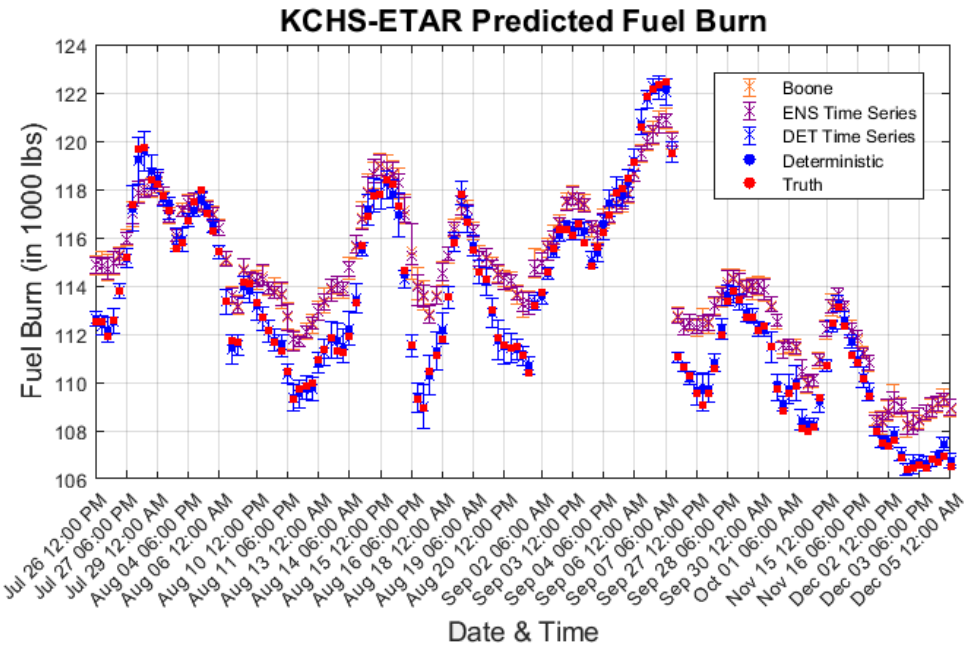

Figure 13. KCHS-ETAR Predicted Fuel Burn

The longest of the four routes analyzed was from South Carolina across the Atlantic Ocean to Germany. The predicted fuel burn is shown in Figure 13 and the RMSE in Figure 14. On average, the pre-takeoff RMSE of the deterministic and time series deterministic method was respectively 161 and 172. Comparatively, the RMSE of Boone's and the time series ensemble method was 1611 and 1549 lbs. Increasingly updated weather predictions all lower the RMSE of the deterministic, ensemble time series, and Boone's method (Figure 15). The deterministic time series saw the large increase in accuracy shedding 2151 lbs off of it's RMSE from +72 hour data to +6 hour data. Similar to past routes, the deterministic time series saw that greatest increase in optimal route selection with an increase in data (Figure 16), and had the highest prediction accuracy using data up to the +6 hour point.

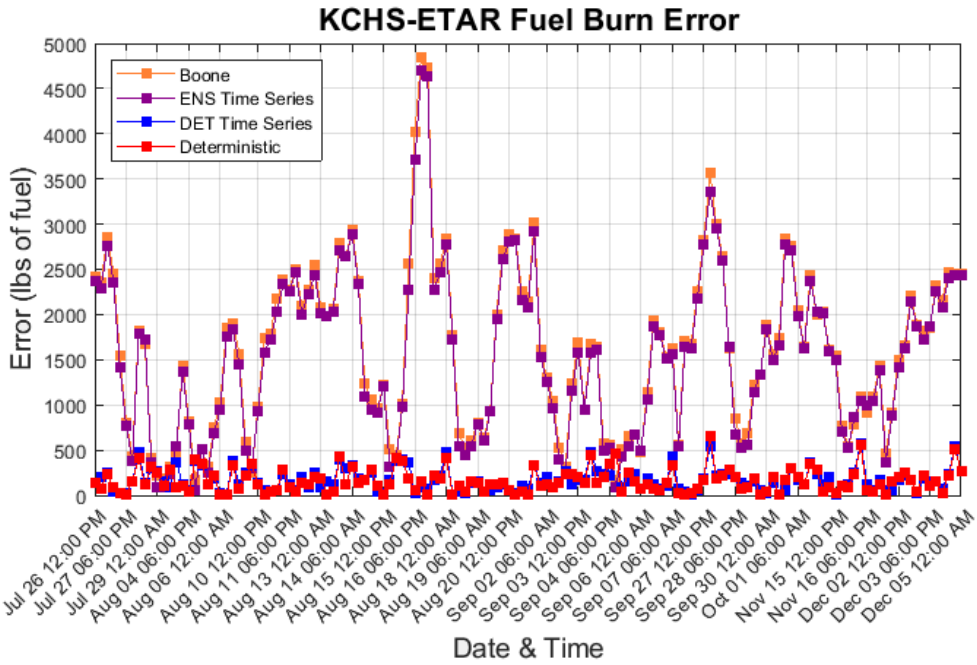

Figure 14. KCHS-ETAR Fuel Burn Prediction Error

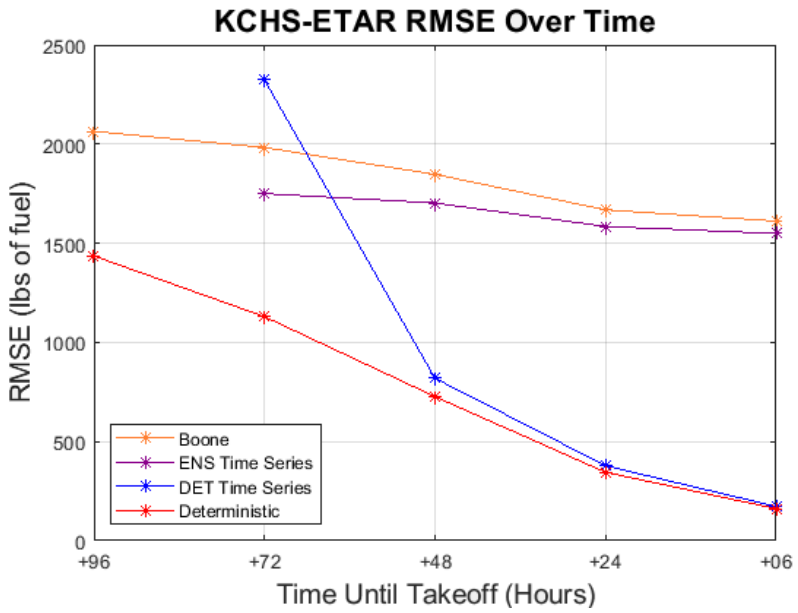

Figure 15. KCHS-ETAR Change In RMSE

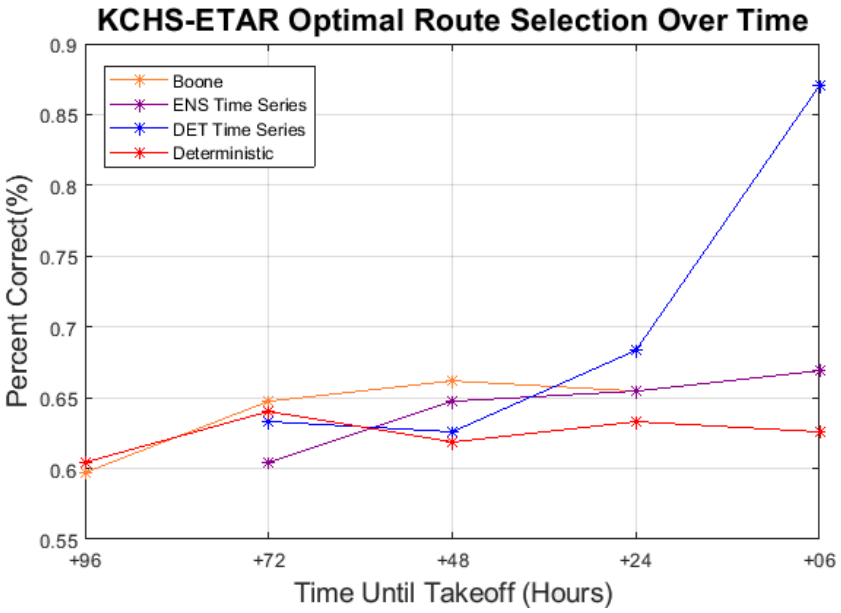

Figure 16. KCHS-ETAR Change In Optimal Path Selection

4.6 KWRI-ETAD

The last route analyzed is another trans-Atlantic route to Germany that originates in New Jersey. The predicted fuel burn is shown in Figure 17 and the associated

RMSE in Figure 18. Once again, the deterministic and time series deterministic methods outperformed the ensemble-based methods having respective pre-takeoff RMSEs of 139 and 138 lbs. This is compared to Boone's method having an RMSE of 1237 lbs and the ensemble time series method having an RMSE of 1176 lbs in the same period. Once again, the ensemble time series method consistently gives a lower average RMSE than Boone's method no matter the available information as seen in Figure 19. Boone's method is the best predictor of optimal paths past two days out while time series deterministic is the best otherwise (Figure 20).

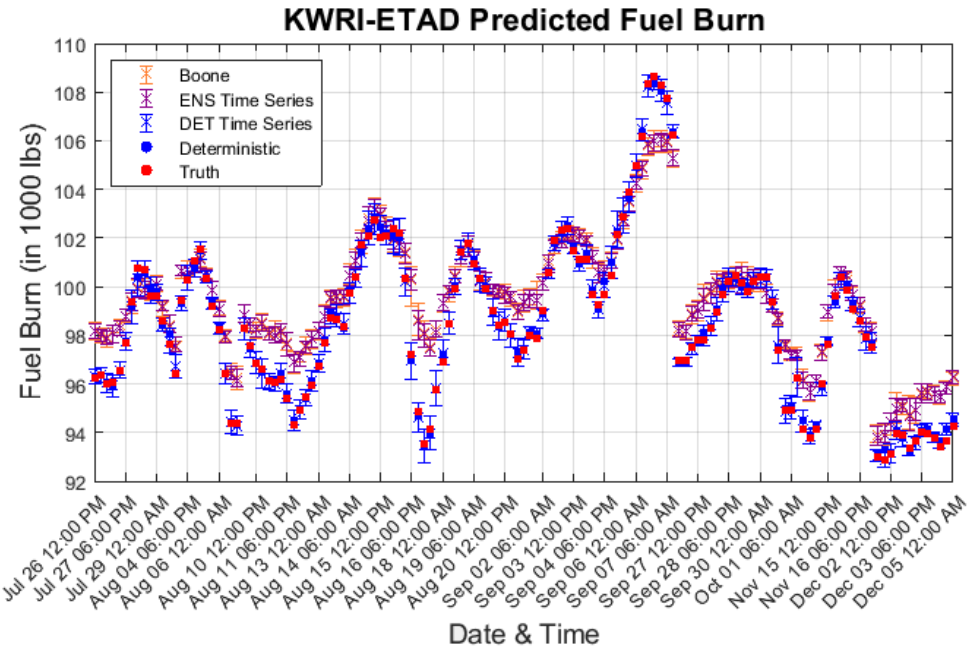

Figure 17. KWRI-ETAD Predicted Fuel Burn

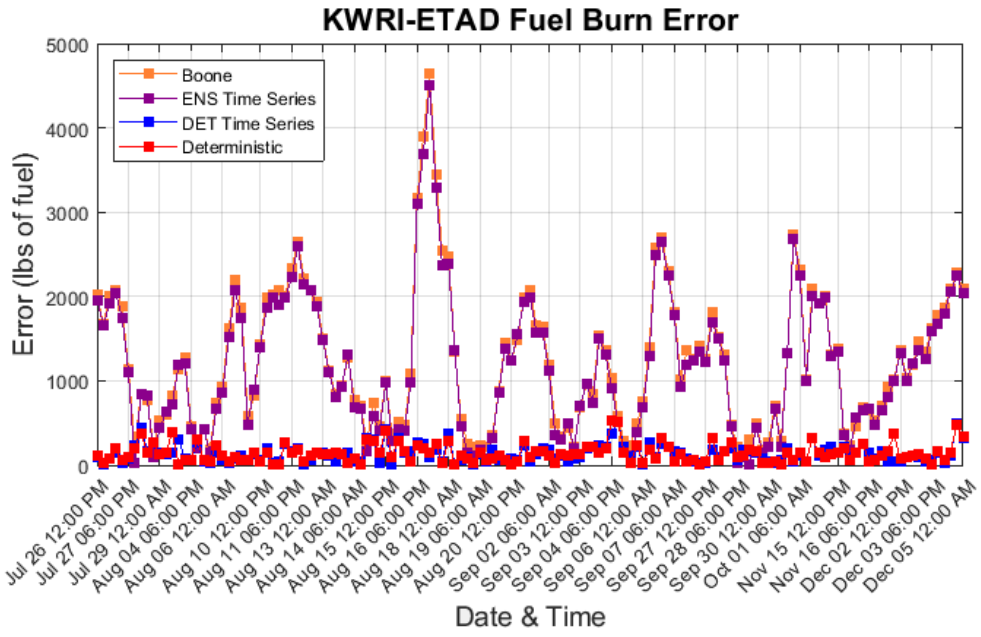

Figure 18. KWRI-ETAD Fuel Burn Prediction Error

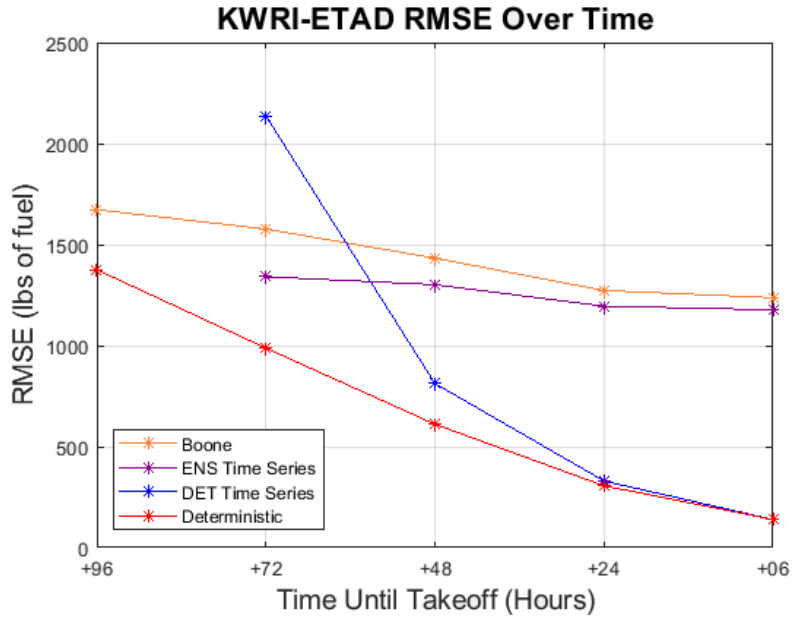

Figure 19. KWRI-ETAD Change In RMSE

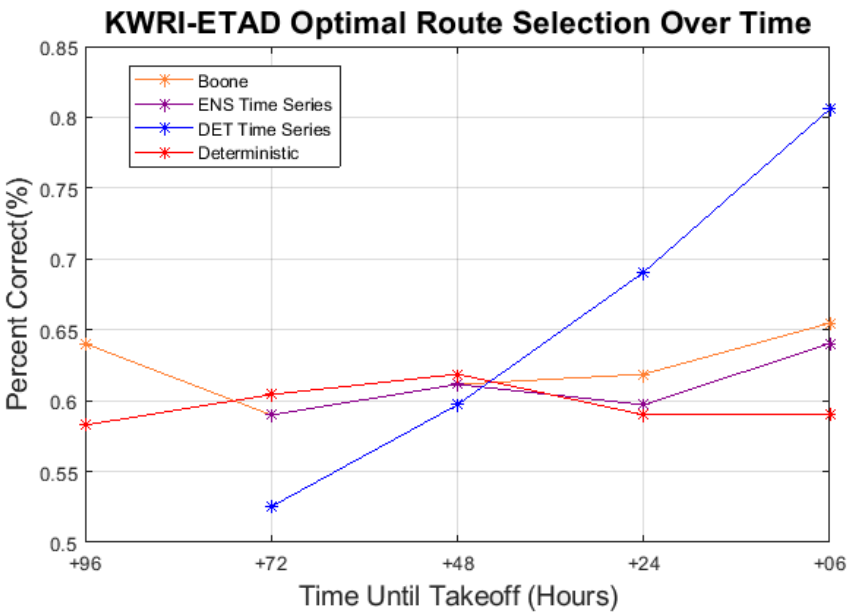

Figure 20. KWRI-ETAD Change In Optimal Path Selection

V. Conclusions and Future Research

5.1 Conclusion

When dealing with large networks, such as the ones created when flight planning for C-17s, the number of possible of solutions is very large. Adding in correlated stochastic arcs to a large network makes finding the optimal solution extremely difficult. Using continuously updating weather data gives an opportunity to solve slightly different versions of a large network with stochastic correlated arcs multiple times, adding more insight into the network and potential optimal solutions.

Out of the four methods introduced, the deterministic-based methods consistently produce lower RMSEs than the ensemble-based methods for all routes. Performing time series analysis to Boone's method gave a slight increase in accuracy to the predicted fuel burn of every route. Using time series analysis on the deterministic method had a nominal affect on increasing accuracy when using sufficient data and predicting reasonably close. When not meeting this criteria, such as predicting three days using time series analysis on two days of data, the deterministic time series had the lowest fuel burn prediction accuracy.

The percentage of each method's correct future optimal path prediction rate across all routes is shown in Table 6. In total, Boone's method and the ensemble

Table 6. Percent of Future Optimal Paths Correctly Predicted Over 4-Day Window

Future Optimal Paths Correctly Predicted (%)					
	Latest Forecast Available				
	+96	**+72**	**+48**	**+24**	**+6**
Boone	61.6	62.0	63.0	63.2	65.3
ENS Time Series	N/A	58.3	63.0	62.8	65.3
DET Time Series	N/A	64.0	66.3	74.9	87.4
Deterministic	62.8	64.2	62.2	61.4	62.0

time series method both correctly predicted 65% of optimal future routes with data up to six hours before takeoff. Comparatively, the deterministic method predicted 62% and deterministic time series 87% in this same period. At three days prior to takeoff all four methods predict the correct optimal route between 58% and 64%. Over the course of a planning period, the deterministic time series saw the highest improvement rate.

Of the metrics concerning AMC the current way of flight planning using the GFS remains the best way. For improving fuel burn predictions the deterministic method is constantly the best method throughout the planning period. Further, time series analysis on deterministic data offers no improvement at the cost of creating more networks. The added networks created by the ensemble-based methods also show no improvement on the accuracy of fuel burn predictions. For better predicting future optimal paths the current deterministic method gives the best results early in the planning window. In the last two days of the planning window the Boone, ensemble time series, and deterministic time series all give more accurate future path predictions. However, the deterministic time series gives the largest increase in path selection accuracy despite needing the least amount of networks. Therefor, if looking to increase optimal path prediction rate, the deterministic time series is the best method to use under 48 hours of takeoff.

This conclusion hinges on the idea that the GFS +00 model is actually the true weather. This is a necessary assumption that highly skews performance in favor of the deterministic-based methods. Comparing these predictions to real-world C-17 fuel burn data would have been preferred, but was unavailable in this project. Also of concern, Reiman's [20] C-17 fuel burn regression models are only based on aircraft weight and changes in altitude. They were used in this work by converting the wind an aircraft faced into an effective travel distance, but there are physical properties

wind could affect, such as engine performance, that were considered negligible.

5.2 Future Research

Boone [7] introduced a heuristic that developed a reasonable candidate list of potential optimal paths to increase the AMC's likelihood of predicting the future optimal path. This research used time series analysis to improve upon Boone's heuristic. Further, a prediction interval of predicted fuel burn in deterministic based weather networks was generated as well as an increased ability to find the future minimal fuel burn path. Future research should concentrate on comparing more weather models, using higher resolution data, and validating C-17 fuel burn predictions.

Appendix A. Solving the Network

SolveRoutes.m

```
clear;clc;

% Number of hours in the forecast we have
numHrs = 5*24;

% Flight routes
 routes = {'KSUU-PHNL','KTCM-CYQX','KCHS-LERT','KCHS-ETAR','KWRI-ETAD'};

% 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)
                    32.8986389, -80.0405278, 36.645, -6.349444; %Charleston Lat/Long, Rota Lat/
                        Long (skyvector)
                   32.8986389, -80.0405278, 49.436167, 7.6065; %Charleston, Ramstein (skyvector)
                   40.0155833, -74.5916991, 49.976389, 6.698333];  %JBMDL lat/long, spangdahlem
                       lat/long
[num_routes,~] = size(latlongs_routes);

% Determine waypoints for each route in 1/2 degree increments
waypts = ceil(distance(latlongs_routes(:, 1  ), latlongs_routes(:, 2), latlongs_routes(:, 3),
    latlongs_routes(:, 4))) * 2;
%hours to travel (assume each degree is 60nm and plane travels at 450 TAS)
hourdists = distance(latlongs_routes(:, 1  ), latlongs_routes(:, 2), latlongs_routes(:, 3),
    latlongs_routes(:, 4))*60/450;

H = -numHrs:6:0; %Number of forecast models for a given day
FcstSize = size(H,2)-2; %Number of forecasts to run

%Altitudes the plane can fly at
AltLevels=[25,26,27,28,29,30,31,32,33,34,35]*1000;

%Pressure levels in feet of data
KnownPLevels = 100*[250,300,350];

Dates = [(datetime(2018,8,19,'TimeZone','utc')+hours(12)):hours(6):(datetime(2018,8,21,'TimeZone',
    'utc')+hours(12))...
    (datetime(2018,9,2,'TimeZone','utc')+hours(0)):hours(6):(datetime(2018,9,7,'TimeZone','utc')+
        hours(12))...
    (datetime(2018,9,26,'TimeZone','utc')+hours(18)):hours(6):(datetime(2018,10,1,'TimeZone','utc'
        )+hours(12))...
    (datetime(2018,11,14,'TimeZone','utc')+hours(12)):hours(6):(datetime(2018,11,17,'TimeZone','
        utc')+hours(6)) ...
    (datetime(2018,12,2,'TimeZone','utc')+hours(0)):hours(6):(datetime(2018,12,5,'TimeZone','utc')
        +hours(0))];

%Dates to interpolate to help predict
```

```
Dates2 = Dates + hours(6);
Dates3 = Dates + hours(12);
length_t = length(Dates);

for i = 5:1:5   %num_routes
    %generate lat/lon vectors
    [lat,lon] = gcwaypts(latlongs_routes(i,1), latlongs_routes(i,2), latlongs_routes(i,3),
        latlongs_routes(i,4), waypts(i));
    lon = mod(lon,360);

    %Pressures to be used when interpolating data
    RepeatingPressures = [repmat(KnownPLevels(1),[size(lat),1]);repmat(KnownPLevels(2),[size(lat)
        ,1]);repmat(KnownPLevels(3),[size(lat),1])];

%   Matrix of data to record
    preds = zeros(89,length_t);

    %how much time moves when traveling a waypoint
    timestep_mvmt = (hourdists(i)/6)/waypts(i);

    for pred = 1:1:length_t
        dists = zeros(21,size(H,2));%Cell that will hold optimal dists
        paths = cell(size(dists)); %Cell that will hold paths of optimal
        paths_DET = cell(1,FcstSize);%Hold paths of det optimal

        graphs = cell(size(dists)); %Cell that will hold each network
        graphs_DET = cell(size(paths_DET));%Cell that will hold each Det network
        dists_DET = cell(size(paths_DET));

        OptimalRoutes = zeros(21*FcstSize,waypts(i)+1); %Matrix form of optimal routes
        OptimalRoutes_DET = zeros(FcstSize,waypts(i)+1);%Matrix form of det optimal routes
        BooneOptimalRoutes = zeros(21,waypts(i)+1);%Matrix form of boone optimal routes


        FileName_ENS = strcat('E_', num2str(year(Dates(pred)),'%04i'), num2str(month(Dates(pred)),
            '%02i'), num2str(day(Dates(pred)),'%02i'),num2str(hour(Dates(pred)),'%02i'),'.mat');
        ENS = load(strcat(routes{i},'/',FileName_ENS)); %Pull out the mat file

        FileName_ENS2 = strcat('E_', num2str(year(Dates2(pred)),'%04i'), num2str(month(Dates2(pred
            )),'%02i'), num2str(day(Dates2(pred)),'%02i'),num2str(hour(Dates2(pred)),'%02i'),'.mat
            ');
        ENS2 = load(strcat(routes{i},'/',FileName_ENS2)); %Pull out the mat file (interpolation)

        FileName_ENS3 = strcat('E_', num2str(year(Dates3(pred)),'%04i'), num2str(month(Dates3(pred
            )),'%02i'), num2str(day(Dates3(pred)),'%02i'),num2str(hour(Dates3(pred)),'%02i'),'.mat
            ');
        ENS3 = load(strcat(routes{i},'/',FileName_ENS3)); %Pull out the mat file (interpolation)


        FileName_DET = strcat('D_', num2str(year(Dates(pred)),'%04i'), num2str(month(Dates(pred)),
            '%02i'), num2str(day(Dates(pred)),'%02i'),num2str(hour(Dates(pred)),'%02i'),'.mat');
        DET = load(strcat(routes{i},'/',FileName_DET));
```

```
FileName_DET2 = strcat('D_', num2str(year(Dates2(pred)),'%04i'), num2str(month(Dates2(pred
    )),'%02i'), num2str(day(Dates2(pred)),'%02i'),num2str(hour(Dates2(pred)),'%02i'),'.mat
    ');
DET2 = load(strcat(routes{i},'/',FileName_DET2));

FileName_DET3 = strcat('D_', num2str(year(Dates3(pred)),'%04i'), num2str(month(Dates3(pred
    )),'%02i'), num2str(day(Dates3(pred)),'%02i'),num2str(hour(Dates3(pred)),'%02i'),'.mat
    ');
DET3 = load(strcat(routes{i},'/',FileName_DET3));

for steps = 1:FcstSize



  %Pull all truth data if checking last route
  if steps == FcstSize
        %Matrix is UWindValues(time,lat,lon)
        DET_U = squeeze(DET.U(steps+2,:,:))';
        DET_V = squeeze(DET.V(steps+2,:,:))';
        DET_U2 = squeeze(DET2.U(steps+2,:,:))';
        DET_V2 = squeeze(DET2.V(steps+2,:,:))';
        DET_U3 = squeeze(DET3.U(steps+2,:,:))';
        DET_V3 = squeeze(DET3.V(steps+2,:,:))';

  else %Taper truth data based on what would be available
        DET_U = squeeze(DET.U(steps+2,:,:))';
        DET_V = squeeze(DET.V(steps+2,:,:))';
        DET_U2 = squeeze(DET2.U(steps+1,:,:))';
        DET_V2 = squeeze(DET2.V(steps+1,:,:))';
        DET_U3 = squeeze(DET3.U(steps,:,:))';
        DET_V3 = squeeze(DET3.V(steps,:,:))';
  end
    %Prepare Us/Vs for interpolation
    RepeatingUs_DET = [DET_U(:,1);DET_U(:,2);DET_U(:,3)];
    RepeatingVs_DET = [DET_V(:,1);DET_V(:,2);DET_V(:,3)];
    RepeatingUs_DET2 = [DET_U2(:,1);DET_U2(:,2);DET_U2(:,3)];
    RepeatingVs_DET2 = [DET_V2(:,1);DET_V2(:,2);DET_V2(:,3)];
    RepeatingUs_DET3 = [DET_U3(:,1);DET_U3(:,2);DET_U3(:,3)];
    RepeatingVs_DET3 = [DET_V3(:,1);DET_V3(:,2);DET_V3(:,3)];

    PressureInt_U_DET = zeros(11,waypts(i)+1);
    PressureInt_V_DET = PressureInt_U_DET;
    PressureInt_U_DET2 = PressureInt_U_DET;
    PressureInt_V_DET2 = PressureInt_U_DET;
    PressureInt_U_DET3 = PressureInt_U_DET;
    PressureInt_V_DET3 = PressureInt_U_DET;

    Us_Time_DET = zeros(size(PressureInt_U_DET));
    Vs_Time_DET = Us_Time_DET;

    for kk = 1:1:(waypts(i)+1) %Lat/Lon interpolation for Det, done at each way point

        if kk*timestep_mvmt <= 1 %Interpolate between 0-6 hours from takeoff
```

```
            PressureInt_U_DET(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingUs_DET,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');
            PressureInt_V_DET(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingVs_DET,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');

            PressureInt_U_DET2(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingUs_DET2,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');
            PressureInt_V_DET2(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingVs_DET2,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');

            Us_Time_DET(:,kk) = PressureInt_U_DET(:,kk)*(1-kk*timestep_mvmt) +
                PressureInt_U_DET2(:,kk)*(timestep_mvmt*kk);
            Vs_Time_DET(:,kk) = PressureInt_V_DET(:,kk)*(1-kk*timestep_mvmt) +
                PressureInt_V_DET2(:,kk)*(timestep_mvmt*kk);
        else %Interpolate from 6-12 hours from takeoff
            PressureInt_U_DET2(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingUs_DET2,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');
            PressureInt_V_DET2(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingVs_DET2,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');

            PressureInt_U_DET3(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingUs_DET3,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');
            PressureInt_V_DET3(:,kk) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
                RepeatingPressures,RepeatingVs_DET3,repmat(lat(kk),[11,1]),repmat(lon(
                kk),[11,1]),AltLevels');

            Us_Time_DET(:,kk) = PressureInt_U_DET2(:,kk)*(2-kk*timestep_mvmt) +
                PressureInt_U_DET3(:,kk)* (timestep_mvmt*kk-1);
            Vs_Time_DET(:,kk) = PressureInt_V_DET2(:,kk)*(2-kk*timestep_mvmt) +
                PressureInt_V_DET3(:,kk)* (timestep_mvmt*kk-1);
        end
    end
     %Solve and record
    [paths_DET{steps} ,dists_DET{steps}, graphs_DET{steps}] = Networks(Us_Time_DET,
        Vs_Time_DET,waypts(i),lat,lon);
    OptimalRoutes_DET(steps,:) = cell2mat(paths_DET(steps));

        parfor j = 1:1:21 %21 ensembles
            %Matrix is of form UWinds(Ensemble,Time,lat,lon)
            xx_U = squeeze(ENS.U(j,steps+2,:,:))';
            xx_V = squeeze(ENS.V(j,steps+2,:,:))';
            xx_U2 = squeeze(ENS2.U(j,steps+1,:,:))';
            xx_V2 = squeeze(ENS2.V(j,steps+1,:,:))';
            xx_U3 = squeeze(ENS3.U(j,steps,:,:))';
            xx_V3 = squeeze(ENS3.V(j,steps,:,:))';
             %Prepare for interpolation
            RepeatingUs = [xx_U(:,1);xx_U(:,2);xx_U(:,3)];
```

```
RepeatingVs = [xx_V(:,1);xx_V(:,2);xx_V(:,3)];
RepeatingUs2 = [xx_U2(:,1);xx_U2(:,2);xx_U2(:,3)];
RepeatingVs2 = [xx_V2(:,1);xx_V2(:,2);xx_V2(:,3)];
RepeatingUs3 = [xx_U3(:,1);xx_U3(:,2);xx_U3(:,3)];
RepeatingVs3 = [xx_V3(:,1);xx_V3(:,2);xx_V3(:,3)];

PressureInt_U = zeros(11,waypts(i)+1);
PressureInt_V = PressureInt_U;
PressureInt_U2 = PressureInt_U;
PressureInt_V2 = PressureInt_U;
PressureInt_U3 = PressureInt_U;
PressureInt_V3 = PressureInt_U;

Us_Time = PressureInt_U;
Vs_Time = PressureInt_U;

for k = 1:1:(waypts(i)+1) %Number of lat/lons spots on journey for Ensemble
    if k*timestep_mvmt <= 1 %0-6 hour interpolation
        PressureInt_U(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingUs,repmat(lat(k),[11,1]),repmat(lon(k)
            ,[11,1]),AltLevels');
        PressureInt_V(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingVs,repmat(lat(k),[11,1]),repmat(lon(k)
            ,[11,1]),AltLevels');

        PressureInt_U2(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingUs2,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');
        PressureInt_V2(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingVs2,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');
        %Record based on where the plane is
        Us_Time(:,k) = PressureInt_U(:,k)*(1-k*timestep_mvmt) + PressureInt_U2
            (:,k)*(timestep_mvmt*k);
        Vs_Time(:,k) = PressureInt_V(:,k)*(1-k*timestep_mvmt) + PressureInt_V2
            (:,k)*(timestep_mvmt*k);
    else %6-12 hour interpolation
        PressureInt_U2(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingUs2,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');
        PressureInt_V2(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingVs2,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');

        PressureInt_U3(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingUs3,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');
        PressureInt_V3(:,k) = griddata(repmat(lat,[3,1]),repmat(lon,[3,1]),
            RepeatingPressures,RepeatingVs3,repmat(lat(k),[11,1]),repmat(lon(k
            ),[11,1]),AltLevels');
        %Record based on where the plane is
        Us_Time(:,k) = PressureInt_U2(:,k)*(2-k*timestep_mvmt) +
            PressureInt_U3(:,k)* (timestep_mvmt*k-1);
```

```
                    Vs_Time(:,k) = PressureInt_V2(:,k)*(2-k*timestep_mvmt) +
                        PressureInt_V3(:,k)* (timestep_mvmt*k-1);
                end
            end
            %Solve optimal for each ensemble
                [paths{j,steps} ,dists(j,steps), graphs{j,steps}] = Networks(Us_Time,
                    Vs_Time,waypts(i),lat,lon);

        end


        %Record optimal
        OptimalRoutes((((steps-1)*21+1):((steps-1)*21+21)),:) = cell2mat(paths(:,steps));


end

    %Times worth checking
    check_idx =[3 7 11 15 18];
    %Results to record
    OptDetPath_Pred = zeros(5,1);

    Min_TsDet_Pred = OptDetPath_Pred;
    TSDetPredInt = OptDetPath_Pred;

    Boone_Pred = OptDetPath_Pred;
    Boone_lower_pred = OptDetPath_Pred;
    Boone_upper_pred = OptDetPath_Pred;

    EnsTs_FuelPred  = OptDetPath_Pred;
    TS_lower_pred  = OptDetPath_Pred;
    TS_upper_pred = OptDetPath_Pred;

    DET_RMSE = OptDetPath_Pred;
    BooneRMSE = OptDetPath_Pred;
    DET_TS_RMSE = OptDetPath_Pred;
    TS_RMSE = OptDetPath_Pred;

    DetCorrect = OptDetPath_Pred;
    TS_DET_Correct = OptDetPath_Pred;
    Boone_Correct = OptDetPath_Pred;
    TS_Correct = OptDetPath_Pred;

    OptPath = paths_DET{FcstSize};
    OptPathFuel = dists_DET{FcstSize};

for checks = 1:5

FcstDist = 19-check_idx(checks);%Used for timeseries prediction

%Find Unique Routes out of all routes that were taken
%Second matrix will contain actual network distances (fuel)
UniqueRoutes = unique(OptimalRoutes(1:(check_idx(checks)*21),:),'rows'); %Not taking any
    routes from last run (truth data)
```

```
TimeDistances = zeros(size(UniqueRoutes,1),21,check_idx(checks));


%Find Unique Boone Routes
BooneUnique = unique(OptimalRoutes(((check_idx(checks)*21)-21):(check_idx(checks)*21),:),'
    rows');
BooneDistances = zeros(size(BooneUnique,1),21);

%Unique Det Routes
UniqueRoutes_DET = unique(OptimalRoutes_DET(1:check_idx(checks),:),'rows'); %Not taking
    any routes from last run (truth data)
TimeDistances_DET = zeros(size(UniqueRoutes_DET,1),check_idx(checks));

%Take each optimal route (Det/Ensemble), and calculate path through
%each respective network across all forecasts
for times = 1:check_idx(checks)

    graph_DET = graphs_DET{1,times};

    for detpaths = 1:1:size(UniqueRoutes_DET,1)
        dist_DET = 0;

        for points_DET = 1:1:waypts(i) %Sum of distances in/out each node on path
            dist_DET = dist_DET + graph_DET(and(graph_DET.InNodes == UniqueRoutes_DET(
                detpaths,points_DET),graph_DET.OutNodes == UniqueRoutes_DET(detpaths,
                points_DET+1)),:).Weight;
        end

        TimeDistances_DET(detpaths,times) = dist_DET;
    end


    %Ensemble adding up of distances across each optimal route and
    %network already calculated

    for ensmb = 1:1:21
        graph = graphs{ensmb,times};

        for optpaths = 1:1:size(UniqueRoutes,1)
            dist = 0;

            for points = 1:1:waypts(i)
                dist = dist + graph(and(graph.InNodes == UniqueRoutes(optpaths,points),
                    graph.OutNodes == UniqueRoutes(optpaths,points+1)),:).Weight;
            end
            TimeDistances(optpaths,ensmb,times) = dist;
        end
    end
end


%           Calculate Boone Distances
    for boonepaths = 1:1:size(BooneUnique,1)
```

```
    for ensmb_B = 1:1:21
        graph_B = graphs{ensmb_B,check_idx(checks)};
        dist_B = 0;
        for points_B = 1:1:waypts(i)
            dist_B = dist_B + graph_B(and(graph_B.InNodes == BooneUnique(boonepaths,
                points_B),graph_B.OutNodes == BooneUnique(boonepaths,points_B+1)),:).
                Weight;
        end
        BooneDistances(boonepaths,ensmb_B) = dist_B;
    end
end

    %Deterministic (Best Path From Last Forecast)
    OptDetPath = paths_DET{checks};
    %Search Optimal Det Routes for OptDetPath, pluf index into
    [~,OptDetPath_Loc] = ismember(OptDetPath,UniqueRoutes_DET,'rows');

    %Find Predicted Fuel Burn
    OptDetPath_Pred(checks) = TimeDistances_DET(OptDetPath_Loc,check_idx(checks));
    DET_RMSE(checks) = sqrt((OptDetPath_Pred(checks) - OptPathFuel)^2)*1000;
    DetCorrect(checks) = isequal(OptDetPath,paths_DET{FcstSize});

    %Ensemble (Boones Method)
     BooneAverages = mean(BooneDistances,2);
    [Boone_Pred(checks),BooneLoc] = min(BooneAverages);
    BoonePath = BooneUnique(BooneLoc,:);
    %Creater kernel CDF for 95% PI
    [f_b,xi_b] = ksdensity(BooneDistances(BooneLoc,:),'Function','cdf');
    [~,uniques_det_idx] = unique(f_b); %Interp1 wont work if density values are
        repeated
    %need unique (x,y) pairs
    Boone_lower_pred(checks) = interp1(f_b(uniques_det_idx),xi_b(uniques_det_idx)
        ,.025,'pchip');
    Boone_upper_pred(checks) = interp1(f_b(uniques_det_idx),xi_b(uniques_det_idx)
        ,.975,'pchip');
    %See if the right path was chosen
    Boone_Correct(checks) = isequal(BoonePath,paths_DET{FcstSize});
    %RMSE
    BooneRMSE(checks) = sqrt(sum((BooneDistances(BooneLoc,:) - OptPathFuel).^2)/20)
        *1000;
    if checks > 1 %Dont do at 1, not enough data for time series
    %TS Det matrixes to fill
    TS_DET = cell(size(UniqueRoutes_DET,1),1);
    DetPredictions = zeros(size(UniqueRoutes_DET,1),1);
    DetStandardDevs = DetPredictions;
    DetPredInt = DetPredictions;

        for uniques_DET = 1:1:size(UniqueRoutes_DET,1)
            TS_DET{uniques_DET,1} = timeseries(TimeDistances_DET(uniques_DET,:));
            %2nd Order auto regression
            sys = ar(squeeze(TS_DET{uniques_DET,1}.Data),2);
            [temp_pred,~,~,temp_sd,~,~] = forecast(sys,squeeze(TS_DET{uniques_DET,1}.
                Data),FcstDist);
            %Only care about last predicition
```

```
        DetStandardDevs(uniques_DET,1) = temp_sd(FcstDist);
        DetPredictions(uniques_DET,1) = temp_pred(FcstDist);
        tempdata = squeeze(TS_DET{uniques_DET,1}.Data);

        X_det = [ones(check_idx(checks)-2,1) tempdata(1:(check_idx(checks)-2))
            tempdata(2:(check_idx(checks)-1))];
        x0_det = [1 tempdata((check_idx(checks)-2):(check_idx(checks)-1))']';
        dist = x0_det'*inv(X_det'*X_det)*x0_det;
        %Pred Interval
        DetPredInt(uniques_DET,1) = tinv([0.975],check_idx(checks) -3)*
            DetStandardDevs(uniques_DET,1)*sqrt(dist);  %n = Fcst Size -1, p = 2

    end

    [Min_TsDet_Pred(checks),DetTSLoc] = min(DetPredictions);

    TSDetPredInt(checks) = DetPredInt(DetTSLoc,1);
    TsDetPath = UniqueRoutes_DET(DetTSLoc,:);
    TS_DET_Correct(checks) = isequal(TsDetPath,paths_DET{FcstSize});
    DET_TS_RMSE(checks) = sqrt((Min_TsDet_Pred(checks) - OptPathFuel)^2)*1000;

    %Ensemble (Time Series)
    TS_ENS = cell(size(UniqueRoutes,1),21);

    ENSPredictions = zeros(size(UniqueRoutes,1),21);
    ENSStandardDevs = ENSPredictions;

    for ENS_Uniques = 1:size(UniqueRoutes,1)

        for ensemb = 1:21
            TS_ENS{ENS_Uniques,ensemb} = timeseries(TimeDistances(ENS_Uniques,
                ensemb,:));
            sys_ENS = ar(squeeze(TS_ENS{ENS_Uniques,ensemb}.Data),2);
            [temp_preds,~,temp_sds,~,~] = forecast(sys_ENS,squeeze(TS_ENS{
                ENS_Uniques,ensemb}.Data),FcstDist);
             ENSPredictions(ENS_Uniques,ensemb)=temp_preds(FcstDist);
             ENSStandardDevs(ENS_Uniques,ensemb)=temp_sds(FcstDist);
        end
    end

    TSAverages = mean(ENSPredictions,2);
    [TS_Pred,TSLoc] = min(TSAverages);

    TSBestDist = 0;
    EnsTs_FuelPred(checks) = median(ENSPredictions(TSLoc,:));
    TSPath = UniqueRoutes(TSLoc,:);

    TS_Correct(checks) = isequal(TSPath,paths_DET{FcstSize});
    TS_RMSE(checks) = sqrt((EnsTs_FuelPred(checks) - OptPathFuel)^2)*1000;

    %KDE for Ens
    [f,xi] = ksdensity(ENSPredictions(TSLoc,:),'Function','cdf');
    [~,uniques_idx] = unique(f);
    TS_lower_pred(checks) = interp1(f(uniques_idx),xi(uniques_idx),.025,'pchip');
```

```
                    TS_upper_pred(checks) = interp1(f(uniques_idx),xi(uniques_idx),.975,'pchip');
                    end

        end

            %Compile the different predictions
            preds(1,pred) = OptPathFuel;
            preds(2:6,pred) = OptDetPath_Pred(:);

            preds(7:11,pred) = Min_TsDet_Pred(:);
            preds(12:16,pred) = TSDetPredInt(:);

            preds(17:21,pred) = EnsTs_FuelPred(:);
            preds(22:26,pred) = TS_lower_pred(:);
            preds(27:31,pred) = TS_upper_pred(:);

            preds(32:36,pred) = Boone_Pred(:);
            preds(37:41,pred) = Boone_lower_pred(:);
            preds(42:46,pred) = Boone_upper_pred(:);

            preds(47:51,pred) = DET_RMSE(:);
            preds(52:56,pred) = DET_TS_RMSE(:);
            preds(57:61,pred) =  TS_RMSE(:);
            preds(62:66,pred) = BooneRMSE(:);

            preds(67:71,pred) = DetCorrect(:);
            preds(72:76,pred) = TS_DET_Correct(:);
            preds(77:81,pred) = Boone_Correct(:);
            preds(82:86,pred) = TS_Correct(:);

            preds(87,pred) = size(UniqueRoutes_DET,1);
            preds(88,pred) = size(BooneUnique,1);
            preds(89,pred) = size(UniqueRoutes,1);

    end

    filename = 'Results.xlsx';
    T = array2table(preds) ;
    writetable(T,filename,'Sheet',i,'Range','B1');

end
```

Networks.m

```
function [path,dist,graph] = Networks(PressureInt_U,PressureInt_V,waypts,lat,lon)
%Creates network based on wind levels at waypoint

%Calculate the a magnitude of win speed based on U and V
WS = sqrt(PressureInt_U.^2 + PressureInt_V.^2); %Currently in m/s
WS = WS * convvel(1, 'm/s', 'kts'); %convert from m/s to knots

%Calculate which way the wind is blowing based on U and V
angle_W = mod(atan2(-PressureInt_U,-PressureInt_V) * 180/ pi,360);
```

```
%Calculate the heading from waypoint to way point
Az= zeros(waypts+1,1);
Az = azimuth('gc',lat(1:(size(lat)-1)),lon(1:(size(lon)-1)),lat(2:(size(lat))),lon(2:(size(lon))))
    ;

%Give the landing node the same heading as the second to last node (Needed
%for GS)
Az(waypts+1) = Az(waypts);


%Intialize Ground Speed Matrix
GS = zeros(size(WS));

%Calculate Ground Speed Matrix
for i = 1:(waypts+1)
    %True Airspeed is assumed to be 450NM
 [~,GS(:,i),~] = driftcorr(Az(i),450,angle_W(:,i),WS(:,i));
end

%Generate Network
%First Node connects to all 11 nodes at first waypoint
%Start the network at 25k ft
W1 = ArcFuelValues(6,GS(1,6),GS(:,2),[lat(1) lat(2)],[lon(1) lon(2)]);
G = digraph(repmat(1,[1,11]),[2:12],W1);

%Each Waypoint (First and Last Seperate)
    for k = 1:1:(waypts-2)
        %Connect to every Pressure Level of next waypoint
        %m=1 is 25k, m=2 is 26k and so on
        for m = 1:1:11

            %Where W is the Fuel value for each 11 arcs being added
            W = zeros(11,1);
            W = ArcFuelValues(m,GS(m,k),GS(:,k+1),[lat(k) lat(k+1)],[lon(k) lon(k+1)]);
            %Add edges with fuel costs W
            G = addedge(G,repmat(m+1+(k-1)*11,[1,11]) , [(2+k*11):(12+k*11)],W);

        end
    end

    %Get Fuel values for last waypoint going to last node, which is fixed
    %at 25k
     WLast = zeros(11,1);
     for p = 1:11
     WTemp = ArcFuelValues(p,GS(p,waypts),GS(:,waypts+1),[lat(waypts) lat(waypts+1)],[lon(waypts)
          lon(waypts+1)]);
     %WTemp(6) gives the fuel to travel from the altitude p to
     %25k (6) for each p value
     WLast(p) = WTemp(6);
     end
     %Add the fuel arc values from the last waypt to the end node
     G = addedge(G,[(2+(waypts-2)*11):(12+(waypts-2)*11)],repmat(13+(waypts-2)*11,[1,11]),WLast);
     %Shortest Path from node 1 to last node
```

```
    %[dist,path,pred] =
    [path,dist] = shortestpath(G,1,13+(waypts-2)*11);


    T = G.Edges(:,2);%Save to compute distance for other routes
    T.InNodes = G.Edges.EndNodes(:,1);
    T.OutNodes = G.Edges.EndNodes(:,2);
    graph = T;
    G = {};

end
```

FuelCalc.m

```
function fuel_consumed = FuelCalc(ftype ,AC, alpha1,alpha2, omega, PW, dist)
%Climb/Descend,
load RegressionCoeffs.mat
fuel1 = 0;
fuel2 = 0;

dist1=0;
dist2=0;

fuel_consumed = 0;
distance_traveled = 0;
%If Climbing
if strcmpi(ftype,'climb')
    %Distance to climb this amount
    dist1 = Climb_reg_dist(1,AC) + Climb_reg_dist(2,AC)*alpha1 + Climb_reg_dist(3,AC)*alpha1^2 +
        Climb_reg_dist(4,AC)*alpha1^3 + 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)*alpha1^2*omega^3 + 10^(-6)*
        Climb_reg_dist(9,AC)*alpha1^2*omega^3;
    dist2 = Climb_reg_dist(1,AC) + Climb_reg_dist(2,AC)*alpha2 + Climb_reg_dist(3,AC)*alpha2^2 +
        Climb_reg_dist(4,AC)*alpha2^3 + 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)*alpha2^2*omega^3 + 10^(-6)*
        Climb_reg_dist(9,AC)*alpha2^2*omega^3;
    % fuel to climb in Klbs

    fuel1 = Climb_reg_fuel(1,AC) + Climb_reg_fuel(2,AC)*alpha1 + Climb_reg_fuel(3,AC)*alpha1^2 +
        Climb_reg_fuel(4,AC)*alpha1^3 + 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)*alpha1^2*omega^3 + 10^(-6)*
        Climb_reg_fuel(9,AC)*alpha1^1*omega^3;
    fuel2 = Climb_reg_fuel(1,AC) + Climb_reg_fuel(2,AC)*alpha2 + Climb_reg_fuel(3,AC)*alpha2^2 +
        Climb_reg_fuel(4,AC)*alpha2^3 + 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)*alpha2^2*omega^3 + 10^(-6)*
        Climb_reg_fuel(9,AC)*alpha2^2*omega^3;

    distance_traveled = abs(dist1-dist2);
    fuel_consumed = abs(fuel1-fuel2);

elseif strcmpi(ftype,'descd') %If Descending

    dist1 = Descend_reg_dist(1,AC) + Descend_reg_dist(2,AC)*omega + Descend_reg_dist(3,AC)*omega^2
```

```
        + Descend_reg_dist(4,AC)*alpha1 + Descend_reg_dist(5,AC)*alpha1*omega;
    dist2 = Descend_reg_dist(1,AC) + Descend_reg_dist(2,AC)*omega + Descend_reg_dist(3,AC)*omega^2
        + Descend_reg_dist(4,AC)*alpha2 + Descend_reg_dist(5,AC)*alpha2*omega;
    % fuel to descend in Klbs
    fuel1 = Descend_reg_fuel(1,AC) + Descend_reg_fuel(2,AC)*omega + Descend_reg_fuel(3,AC)*omega^2
        + Descend_reg_fuel(4,AC)*alpha1 + Descend_reg_fuel(5,AC)*alpha1*omega;
    fuel2 = Descend_reg_fuel(1,AC) + Descend_reg_fuel(2,AC)*omega + Descend_reg_fuel(3,AC)*omega^2
        + Descend_reg_fuel(4,AC)*alpha2 + Descend_reg_fuel(5,AC)*alpha2*omega;

    distance_traveled = abs(dist1-dist2);
    fuel_consumed = abs(fuel1-fuel2);
end

%Climb or Descent completes within leg which gives time for aircraft cruising
if distance_traveled < dist

    %The distance left to be traveled
    distleft = dist - distance_traveled;

    % fuel to cruise in Klbs
    OW = PayloadAssumptions(1,AC); % operating weight
    FRC = PayloadAssumptions(5,AC); % reserve/contingency fuel weight
    FAH = PayloadAssumptions(6,AC) + PayloadAssumptions(7,AC); %alternate/holding fuel weight
    A = SpecRange_reg(5,AC)/3;
    B = (SpecRange_reg(4,AC)/2) + SpecRange_reg(5,AC)*(OW + FRC + FAH + PW) + (SpecRange_reg(6,AC)
        /2)*alpha2;
    C = SpecRange_reg(1,AC) + SpecRange_reg(2,AC)*alpha2 + SpecRange_reg(3,AC)*alpha2^2 +
        SpecRange_reg(4,AC)*(OW+FRC+FAH+PW)+SpecRange_reg(5,AC)*((OW+FRC+FAH+PW)^2) +
        SpecRange_reg(6,AC)*alpha2*(OW+FRC+FAH+PW);
    D = -distleft;
    %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))
    commonterm1 = 2*B^3 - 9*A*B*C + 27*A^2*D;
    commonterm2 = 4*(B^2 - 3*A*C)^3;
    cubterm1 = ((1/2)*(commonterm1 + sqrt(commonterm1^2 - commonterm2)))^(1/3);%sqrt
    cubterm2 = ((1/2)*abs(commonterm1 - sqrt(commonterm1^2 - commonterm2)))^(1/3) ;
    cubterm2 = sign(commonterm1 - sqrt(commonterm1^2 - commonterm2))*cubterm2;  %b/c matlab yields
        a complex number
    %Add the fuel already used to climb/descend if applicable
    fuel_consumed = -B/(3*A) - (1/(3*A))*cubterm1 - (1/(3*A))*cubterm2 + fuel_consumed;
elseif distance_traveled > dist
    fuel_consumed = 1000; %dont use this route, flight is too steep;
end

end
```

ArcFuelValues.m

```
function [W] = ArcFuelValues(NodeNum,StartNodeGS,EndNodesGS,lat,lon)
%Input is start node and windspeeds, output is arc value (fuel) to travel
%to all possible next 11 nodes
```

```
AC = 2; %Used in the RegressionCoeffs Table
omega = 496.5; %Weight
PW =5; %Payload
TAS = 450; %True airspeed
dist = distdim(distance(lat(1),lon(1),lat(2),lon(2)),'deg','nm');
truedist= zeros(11,1);

    for i = 1:11
        %find distance with altitude
        dist_with_alt = sqrt(dist^2 + distdim(abs(NodeNum-i)*1000,'ft','nm')^2);
        AverageGS = (StartNodeGS + EndNodesGS(i))/2;
        truedist(i) = dist_with_alt * (TAS/AverageGS);%Effective distance
        if NodeNum > i  %Decreasing Altitude
        W(i) = FuelCalc('descd',AC, 24 + NodeNum, 24 + i, omega, PW, truedist(i));
        elseif NodeNum < i %Increasing Altitude
        W(i) = FuelCalc('climb',AC, 24 + NodeNum, 24 + i, omega, PW, truedist(i));
        elseif NodeNum == i %Maintain Altitude
        %Change in altitude is 0
        W(i) = FuelCalc('else',AC,24 + NodeNum, 24 + i, omega, PW, truedist(i));
        end

    end
end
```

Bibliography

1. "USAF Fiscal Year 2019 Budget Overview," tech. rep., 2018.
2. "Air Mobility Command Mission," accessed Jan 15, 2019. https://www.amc.af.mil/About-Us/AMC-Mission/.
3. C. Knight, "AMC Continues to Pursue Fuel Efficiency Initiatives," 2017, accessed June 3, 2018. http://www.amc.af.mil/News/Article-Display/Article/1204903/amc-continues-to-pursue-fuel-efficiency-initiatives/.
4. C. Poland, "How the Air Force Got Smarter About its Aviation Fuel Use in 2018," 2018, accessed January 21, 2019. https://www.af.mil/News/Article-Display/Article/1711969/how-the-air-force-got-smarter-about-its-aviation-fuel-use-in-2018/.
5. National Oceanic and Atmospheric Administration, "Ensemble Prediction Systems," accessed January 21, 2019. https://www.wpc.ncep.noaa.gov/ensembletraining/.
6. H. A. Homan, "Comparison of Ensemble Mean and Deterministic Forecasts for Long-Range Airlift Fuel Planning," *Masters Thesis, Air Force Institute of Technology*, 2014.
7. S. Boone, "Shortest Path Across Stochastic Network with Correlated Random Arcs," *Masters Thesis, Air Force Institute of Technology*, 2017.
8. X. Ji, "Models and Algorithm for Stochastic Shortest Path Problem," *Applied Mathematics and Computation*, vol. 170, no. 1, pp. 503–514, 2005.
9. B. Y. Chen, W. H. Lam, A. Sumalee, and Z. l. Li, "Reliable Shortest Path Finding in Stochastic Networks with Spatial Correlated Link Travel Times," *International Journal of Geographical Information Science*, vol. 26, no. 2, pp. 365–386, 2012.
10. R. Keith and S. Leyton, "An Experiment to Measure the Value of Statistical Probability Forecasts for Airports," *Weather and Forecasting*, vol. 22, pp. 928–935, 2006.
11. T. N. Krishnamurti, C. M. Kishtawal, Z. Zhang, T. Larow, D. Bachiochi, and E. Williford, "Multimodel Ensemble Forecasts for Weather and Seasonal Climate," *Journal of Climate*, vol. 13, no. 23, 2000.
12. L. Bertotti, J. R. Bidlot, R. Buizza, L. Cavaleri, and M. Janousek, "Deterministic and ensemble-based prediction of Adriatic Sea sirocco storms leading to 'acqua alta' in Venice," *Quarterly Journal of the Royal Meteorological Society*, vol. 137, no. 659, pp. 1446–1466, 2011.

13. N. M. Leonardo, B. A. Colle, N. M. Leonardo, and B. A. Colle, "Verification of Multi-Model Ensemble Forecasts of North Atlantic Tropical Cyclones.," *Weather and Forecasting*, no. 2010, pp. 17–0058, 2017.

14. WMO, *Guidelines on Ensemble Prediction Systems and Forecasting*, vol. 1091. 2012.

15. National Oceanic and Atmospheric Administration, "Global Forecast System," accessed February 14, 2019. `https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs`.

16. National Oceanic and Atmospheric Administration, "Global Ensembe Forecast System," accessed February 14, 2019. `https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-ensemble-forecast-system-gefs`.

17. H. K. Ng, B. Sridhar, and S. Grabbe, "Optimizing Aircraft Trajectories with Multiple Cruise Altitudes in the Presence of Winds," *Journal of Aerospace Information Systems*, 2014.

18. Earth Observing Laboratory, "Wind Direction Quick Reference," accessed January 22, 2019. `https://www.eol.ucar.edu/content/wind-direction-quick-reference`.

19. DoD, "C-17 Globemaster III," accessed January 22, 2019. `https://www.af.mil/About-Us/Fact-Sheets/Display/Article/1529726/c-17-globemaster-iii/`.

20. A. D. Reiman, "Enterprise Analysis of Strategic Airlift to Obtain Competitive Advantage Through Fuel Efficency," *PhD Thesis, Air Force Institute of Technology*, no. March, 2014.

21. B. Bowerman, R. O'Connell, and A. Koehler, *Forecasting, Time Series, and Regression: An Applied Approach*. Belmont, CA: Thomson Brooks/Cole, 4th ed., 2005.

22. D. Montgomery, E. Peck, and G. Vining, *Introduction To Linear Regression Analysis*. Hoboken, New Jersey: John Wiley & Sons, Inc, 5th ed., 2012.

23. B. W. Silverman, *Density and Statistics and Data Analysis*, vol. 26. London: Chapman and Hall, 1986.

24. W. Zucchini, A. Berzel, and O. Nenadic, "Applied Smoothing Techniques: Kernel Density Estimation," 2003.

25. T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21-03-2019	Master's Thesis	SEP 2017 - MAR 2019

4. TITLE AND SUBTITLE

TIME SERIES ANALYSIS OF STOCHASTIC NETWORKS WITH CORRELATED RANDOM ARCS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Sands, Brendon, T. 2nd 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-ENS-MS-19-M-147

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

Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

AMC

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

AMC

12. DISTRIBUTION / AVAILABILITY STATEMENT

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT

While modern day weather forecasting isn't perfect, there are many benefits given by the multitude and variety of prediction models. In the interest of routing airplanes, this paper uses time series analysis on successive weather forecast predictions to create wind based fuel-burn networks with stochastic correlated arcs. Networks are populated with either deterministic or ensemble based weather data, and the two data sources with and without time series analysis are compared. Methods were compared by fuel burn prediction accuracy and ability to predict a future optimal path. Of the four options, the ensemble-based methods were on average the least accurate. Using time series analysis with ensemble data gave a nominal change in correct future path prediction and an increase in fuel burn prediction accuracy. The deterministic method gave the most accurate results but the worst correct future path prediction rate. Time series analysis with deterministic data had a marginal decrease in accuracy but the highest correct future path prediction rate.

15. SUBJECT TERMS

Time Series, Stochastic Shortest Path, Stochastic Correlated Arcs, Flight Planning

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	62	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565, x4584; Andrew.Geyer@afit.edu