The Effect of Icing on the Dispatch Reliability of Small Aircraft

Melinda M. Gates

Thesis Submitted to the Faculty of:
Virginia Polytechnic Institute and State University
and Ecole des Mines de Nantes
in partial fulfillment of the requirements for the degree of:

Masters of Science
in
Industrial and Systems Engineering

Dr. C. Patrick Koelling, Ph.D., Chair
Dr. Bruno Castanier, Ph.D., Chair
Mr. Howard Swingle
Dr. Michael R. Taaffe, Ph.D.

October 16, 2004
Blacksburg, Virginia

Keywords: Dispatch Reliability, SATS, Matlab Data Parsing

The Effect of Icing on the Dispatch Reliability of Small Aircraft

Melinda Gates

(ABSTRACT)

In 2000, the National Aeronautics and Space Administration (NASA) initiated a program to promote the use of small aircraft as an additional option for national public transportation. The Small Aircraft Transportation System (SATS) asserted the idea of everyday individuals piloting themselves on trips, within a specified distance range, using a small (4 person), piston powered, un-pressurized aircraft and small airports in close proximity to their origin and destination.

This thesis investigates how one weather phenomenon, in-flight icing, affects the dispatch reliability of this transportation system. Specifically, this research presumes that a route is considered a "no-go" for low time pilots in a small, piston powered aircraft if any icing conditions are forecast along the route at the altitude of the flight during the time the traveler desires to make the trip.

This thesis evaluates direct flights between Cleveland and Boston; Boston and Washington, D.C.; and Washington, D.C. and Cleveland during the months of November through May for the years 2001 to 2003 at maximum cruising altitudes of 6,000 feet, 8,000 feet, 10,000 feet, and 12,000 feet above mean sea level (MSL). It was found that the overall probability of a "no-go" for all three flight paths at the normal cruising altitude of 12,000 feet is 56.8%. When the cruising altitude is reduced to 10,000 feet, 8,000 feet, and 6,000 feet the probability of a "no-go" for all three flight paths reduces to 54.6%, 48.5%, and 43.7% respectively.

Acknowledgements

This research endeavor was made possible with the help of a remarkable group of individuals. Foremost, I would like to thank my committee for their assistance through this process. My dual committee chairmen, Dr. Patrick Koelling and Dr. Bruno Castanier, provided me with the guidance and support to allow me to accomplish this goal. I also thank my committee members, Dr. Michael Taaffe and Mr. Howard Swingle, for their tutelage and time commitment. The success of this research is largely due to the four of you.

I thank Dr. Antonio Trani, Dr. Hojong Baik, and Nicholas Hinze from the Department of Civil and Environmental Engineering. Their vast knowledge on the subject of SATS and Matlab programming enabled me to accomplish this venture.

I would like to thank my family for their continued support and encouragement through the many years of school it took for me to reach this point. You are truly the reason I am where I am today, and I could not have made it through this journey without you. Finally I would like to thank Brian, my best friend, for being there for me and ensuring I never lost sight of my dream.

Table of Contents

List of Figures

List of Tables

Chapter 1- Introduction

1.1 Background Information

Due to the ever growing population and their need for efficient means of travel, alternative modes of transportation are being considered for moving people from one location to another in a timely and safe manner. Roadways have become congested and airline travel is not always in accordance with passenger scheduling requirements. The National Aeronautics and Space Administration (NASA) has proposed an innovative means of transportation that will combine the flexibility of automobile travel with the speed of air travel. This travel method utilizes small aircraft to create an "air taxi" system that is available to fly a small number of passengers from one location to another utilizing an improved flight technology.

1.2 SATS Information

Approximately 600 hub-and-spoke airports exist that service a vast majority of airline traffic. These major airports are becoming congested and are not able to withstand the increased demand for their services. The national airspace system is expected to reach gridlock in approximately 2011 (Long et al, 2001). A 20-year outlook defined by Boeing states that on average, per year, the worldwide economic growth will be 3.2%, the passenger traffic growth will be 5.1%, and the cargo traffic growth will be 6.4% (Boeing, 2003). Small aircraft carry between one and twenty passengers. These aircraft can adequately cover trips that are less than 500 miles without requiring refueling, and currently the overall average speed of travel for these flights on commercial aircraft is only between 35 and 80 miles per hour from doorstep to destination (Holmes, 2004). Even if it presently remains adequate for certain passengers, in the future an increase in demand and air traffic will further decrease this speed making it important to incorporate

new technologies into these necessary components of the transportation system. Although changes could be implemented within the hub-and-spoke system itself, it is still likely that the supply of these airports will not be able to fulfill the escalating demand for their service. An alternative that is not being utilized to capacity is approximately 5,400 public use airports and 18,000 landing facilities within the United States (Holmes, 2004). Incorporating these additional resources would prove beneficial because these facilities are within a 30 minute commute for 98% of the population. The use of these airports could increase the average door-to-door speed through the decrease in the commuting distance and the more readily available flight times that correspond to a decrease in idle time.

The Small Aircraft Transportation System (SATS) is a new vision for transportation that was developed by NASA due to the congestion of both commercial aircraft and automobile travel. This alternative mostly corresponds to diverting automobile travel, due to the limited distances of small aircraft flights. The implementation of a SATS program could: provide additional economic resources to small communities, correlate to a decrease in idle time, incorporate a use for the numerous small airports, and could be implemented globally. The numerous public use airports in existence could support a system aimed at a substantial increase in the amount of small plane traffic, but the limiting factor for increasing their role in aviation transportation has been that less then 10% of these public airports have the required radar equipment for high traffic activities (Holmes, 2004). SATS technology would enable small aircraft to utilize already present landing facilities without the need for control towers or radar surveillance.

The possibility of implementing this type of transportation method initiates much enthusiasm, as well as many unknowns. NASA, the Federal Aviation Administration (FAA), and the National Consortium for Aviation Mobility (NCAM) have been researching the possibility of this inventive transportation system. The research consortium is composed of 130 public and private partners from industry and government. The small planes that will be incorporated with the SATS program will be equipped with on-board computing, advanced flight controls, and “highway in the skies”

displays (Figure 1.1). The "highway in the skies" display is an instrument that can be utilized to determine safe routes with respect to traffic, weather, and other external conditions.

Figure 1.1: Projected SATS Control Panel
Source: http://sats.erau.edu/media/images.html

To visually portray the potential advantages of a SATS system three feasible flight paths are outlined on a map of Virginia (Figure 1.2).

Figure 1.2: Map of Possible Day Routes
Source: http://sats.erau.edu/nationalsats/index.html#travel

The southeast SATS lab website (http://sats.erau.edu/nationalsats/index.html#travel) describes the following possible flight paths that could be completed in a business day. The first course in orange is a depiction of how SATS could be utilized to improve business travel. A businessperson could schedule meetings in three different cities in one business day, and still make it home to eat dinner with their family. The second course in magenta indicates how SATS could be utilized for medical attention. A patient could utilize SATS to travel over 300 miles round trip for an outpatient treatment, while still being able to return home that evening to recover in the comfort of their own home. This will enable people to get the best treatment, rather than settling for what is convenient. Finally, SATS could dramatically increase the reliability of mail delivery. A company in Roanoke could easily ensure the same-day delivery of a package from Richmond. The possibilities of different uses for a SATS system are endless, and could improve the way we conduct business and personal affairs.

1.3 Weather in Aviation

Aircraft can reduce traveling times under the right circumstances; however if an aircraft encounters adverse weather conditions it could provide the contrary effect due to cancelled or delayed flights. Weather conditions that could affect an aircraft and its flight include icing, turbulence, and low visibility. The two main components of aviation consist of commercial aviation and general aviation (GA). GA is defined as aircraft that carry between one and twenty passengers, which correlates to the aircraft that would be used within the SATS system. The GA category accounts for 96% of all aircraft while commercial aircraft account for only 4%, and general aviation accounts for 92% of all aviation accidents (Spirkovska, 2002). Weather is a major factor in these accidents. These small aircraft fly at lower altitudes and slower speeds; therefore they are required to fly in problematic weather more often and for longer periods. These aircraft are engaged in much shorter flights (average of 400 miles and 4 hours), but due to the above factors are still at risk for in-flight icing.

1.3.1 Icing

Icing is a hazard that is unsafe for prolonged aircraft operation and can prove to be detrimental to an aircraft. It affects an aircraft through decreasing the efficiency by roughening the wing surfaces, which disrupts airflow over the wings. This correlates to a reduction in lift and an increase in the drag. Additional weight is also added to the aircraft through the accretion of ice requiring the aircraft to produce additional lift and thrust. Other problems encountered due to icing include invalid flight instrument readings, communication equipment failure, and the inability to operate the brakes and landing gear (Flight Standards Service, 1975).

Structural icing occurs when two conditions are met: liquid precipitation is present and the temperatures are at or below freezing. Although severe icing is often correlated with very low temperatures, the most dangerous icing conditions are prevalent at approximately zero degrees Celsius. These temperatures are more dangerous because if the temperature falls much below this threshold the precipitation will be in the frozen state when it makes contact with the aircraft; therefore it will not adhere to the surface. The texture of the surface icing correlates to the freezing rate of the precipitation. The amount of ice collected depends on the time the aircraft is subjected to the conditions, the density of water present, and the collection efficiency (Civil Aviation Authority, 2000). The collection efficiency increases when any of the following criteria increase: the speed of the aircraft, the magnitude of the water droplets, and the number of small components to adhere (Civil Aviation Authority, 2000). The shape and roughness of the surface icing are what affects the airflow, and the amount of ice accumulated is what increases the weight of the aircraft. Another form of icing that can affect aircraft is induction system icing. This icing occurs when ice forms within the air intake of the engine. However, the scope of this research will limit the focus to only structural icing.

If it is known that surface icing will occur in-flight anti-icing techniques are employed, if the aircraft is so equipped, to clear the aircraft. This will control the accumulation of ice for a certain time period, but the aircraft should still exit the conditions as soon as

possible. Most often these procedures are implemented before the icing conditions are entered, and consist of a chemical fluid or a heat source that will prevent the precipitation from freezing on the surface.

1.4 Research Overview

The SATS concept has generated a vast amount of attention, but many more issues need to be resolved to ensure the viability of such a mode of transportation. One of the remaining issues to ensure that small aircraft will prove to be a feasible travel alternative was to determine the probability that they will be able to complete flights at the time travelers require, or desire, to travel based on adverse weather conditions.

This research addresses the dispatch reliability, the percentage of flights that depart without delay or cancellation, of small aircraft due to icing using a reduced set of origin and destination pairs. The origin and destination pairs consist of three cities: Boston (BOS), Cleveland (CLE), and Washington, D.C. (IAD). These cities were chosen to represent various weather patterns to ensure that a range of conditions were encountered. These cities incorporated six possible one-way flight paths ranging from 287 miles to 560 miles as shown in Table 1.1 and Figure 1.3 (http://www.webflyer.com/travel/milemarker/).

Table 1.1- Origin-Destination Mileage Distances

	Washington, D.C.	**Boston, MA**	**Cleveland, OH**
Washington, D.C.	----------	411 Miles	287 Miles
Boston, MA	411 Miles	----------	560 Miles
Cleveland, OH	287 Miles	560 Miles	----------

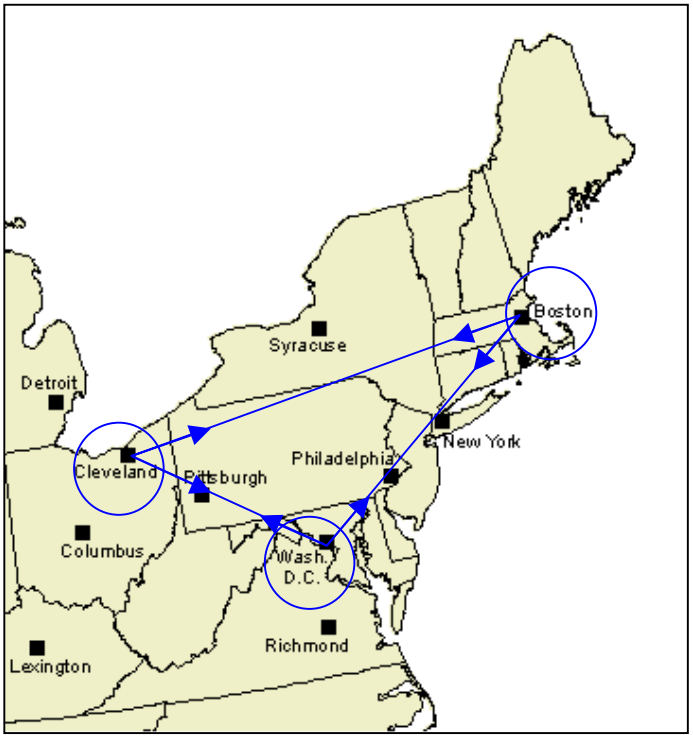

Figure 1.3: Selected Routes for O-D Pairs

Figure 1.4 depicts the six zones for Airman's Meteorological Information (AIRMETs), weather forecasts that were used for this study. All three of the relevant cities are included within the BOS zone; therefore an aircraft completing any of the possible flight paths for this research remained within the BOS zone at all times.

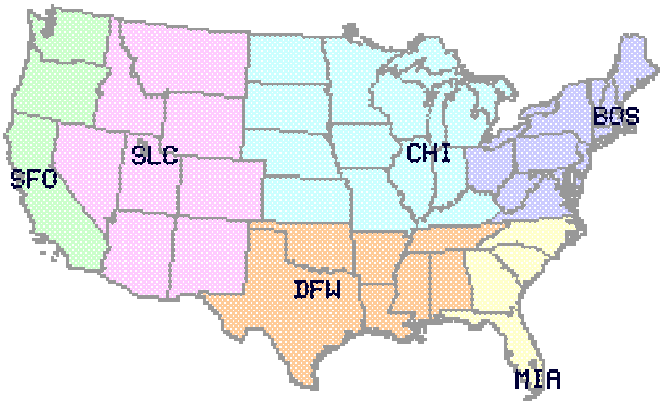

Figure 1.4: AIRMET Zones

Source: http://adds.aviationweather.gov/airmets/

The northeastern portion of the United States was the focus due to the inclement weather that is experienced in this region during the months November to May. This was verified through an experiment that studied the geographic locations in which freezing precipitation and aircraft icing most often occurred (Bernstein et al, 1998). The winter months of November through March were studied from 1993 to 1995. Within this time period 35 hazardous weather cases were evaluated. Throughout this study, the east coast had a significantly greater number of threats than the rest of the continental United States. This is because the airmasses along the east coast contain more moisture. The most prevalent producers of freezing precipitation were east coast airmasses and areas around warm fronts. It was concluded that the environment in the east provides adequate conditions for all types of icing since it also ranked the highest for WORST and SEVERE icing. Therefore, it was determined that by utilizing the northeast in this research a slightly higher "no-go" probability of dispatch reliability for small aircraft would be determined when compared to other areas of the United States.

Icing AIRMETs are issued when the following three criteria are met: the temperature is less than or equal to freezing, liquid precipitation is prevalent, and adequate humidity levels exist. AIRMETs were concluded to be the best representation for this research based on their focus on weather that affects small aircraft. AIRMETs are for widespread weather conditions up to 3,000 square miles, and they are normally issued for six-hour periods. There are three types of AIRMETs: Sierra, Tango, and Zulu. Sierra AIRMETs are issued for instrument flight rules (visibility and low ceilings), Tango AIRMETs are issued for turbulence, and Zulu AIRMETs are issued for icing. Zulu AIRMETs are the AIRMETs that are considered in this research.

Significant Meteorological Information (SIGMET) is another source of aviation weather forecasts. However, they include reports that are relevant to all aircraft (including commercial), and incorporate more severe weather than included in AIRMETs. Brief verification was completed that indicated that AIRMETs and SIGMETs often correlate to approximately the same area. It was assumed that a majority of times when there is going to be severe weather issued through a SIGMET, less severe weather will surround

these conditions and an AIRMET will be issued correlating to approximately the same area. While it is understood that a more complete estimation of dispatch reliability could be determined utilizing both AIRMETs and SIGMETs, it was not feasible based on the cost associated with obtaining these data.

The data were collected for a seven-month period: November to May. This is based on the assumption that icing conditions will be minimal for the geographic area studied in this research for the remaining months, June-October. It is also assumed that when the cabins of an aircraft are un-pressurized the maximum flying altitude is 12,000 feet; therefore the data will only be evaluated from ground level to 12,000 feet to determine if a flight can be safely completed. The aircraft incorporated into this research was a small, piston powered aircraft without pressurization and deicing instruments. These aircraft are assumed to be piloted by "every day" people, and these pilots are not qualified to fly into icing conditions. Therefore, for this research if there were any forecasted icing conditions present along the predetermined flight path the proposed flight was categorized as a "no-go."

The research was carried out by collecting data, formulating probability measures, and forming concluding evidence about the dispatch reliability of small planes with "every day" people as pilots. Chapter two is a literature review of concepts and information that have already been researched. It focuses on providing detailed information on SATS and icing. Chapter three presents the methodology used in this research. It further defines the problem statement, and provides specific details on the data and data parsing, the graphical methods, and the reliability analysis. Chapter four is a compilation of the results from the analysis. Chapter five provides the conclusions and further recommendations.

Chapter 2 - Literature Review

The literature review provides a synopsis of previous research completed to aid in the determination of the methodology required for this research. The initial focus is the SATS concept that NASA is developing. This chapter presents NASA's objectives for the program, and studies they have completed to best ensure the success of the project. Then, the weather condition of icing is outlined in regards to its overall characteristics and the effects it has on aircraft.

2.1 SATS: an economically feasible alternative

The SATS project is a five year $69 million NASA project scheduled to conclude in mid 2005. The scope of this project consists of the following four operating capabilities (http://www.asc.nasa.gov/factsheet/SATS_Fact_Sheet.htm):

1. High-volume operations at airports without control towers or terminal radar facilities;
2. Technologies enabling safe landings at more airports in almost all weather conditions;
3. Integration of SATS aircraft into a higher capacity air traffic control system, with complex flows and slower aircraft, for en route; and
4. Improved single-pilot ability to function competently in evolving, complex national airspace.

In order to succeed with the implementation of SATS, a great deal of research must be completed to ensure that the idea is feasible. Therefore, NASA has completed a range of assessment. This includes determining the type of aircraft that will be the most beneficial for specific trips within the system. The type of aircraft utilized will depend on the length of the trip because different aircraft have different optimal flight distances. In a

study completed by NASA, that reviewed the SATS concept, four different aircraft were being evaluated. These aircraft were (McGrath, 2002):

1. An eight-seater, twin engine, turbofan (TF) engine, fixed wing aircraft;
2. An eight-seater, twin engine, turboprop (TP) engine, fixed wing aircraft;
3. An eight-seater, twin engine, internal combustion engine (2ICE), fixed wing aircraft; and
4. A four-seater, single engine, internal combustion engine (1ICE), fixed wing aircraft.

The unit-cost optimal trip distance varies for each of these aircraft; therefore to uniformly compare them each was simulated flying close to their optimal distance and ideal velocity. Table 2.1 indicates the optimal trip distances for each of the above aircraft (McGrath, 2002).

Table 2.1- Unit-Cost Optimal Trip Distances

Aircraft type	Unit-Cost Optimal Trip Distance
1.	1500 statute miles
2.	900 statute miles
3.	750 statute miles
4.	600 statute miles

McGrath (2002) also completed research, within this study, on the economic benefits of implementing a SATS program. The life cycle cost (LCC) per operating hour, direct operating cost (DOC) per operating hour, and variable cost (VC) per operating hour were incorporated. Initially SATS costs were comparable to airline costs, but once the economic value of an individual's time was accounted for SATS proved to be the most economical transportation method within the airline segment (McGrath, 2002). To maintain equality over the aircraft alternatives the cost of employee travel was set as the time from when the employee left for the arrival airport to the time they arrived at the destination airport. Also, for both forms of transportation a positive return was applied for productivity throughout the course of this time frame. This is a meaningful study for a portion of the SATS goal, but it is not touching on the true aspirations of the SATS system. The SATS concept is not aimed to become an alternative to airline

transportation, but instead a substitute to the automobile (McGrath, 2002). This is due to the limitations placed on the distances of trips that small aircraft can fly based on their reduced fuel capacity, and the increased time required for a commercial aircraft to make a short distance flight.

Another study (McGrath and Young, 2002), that correlates directly with the main goal of SATS, compares the costs of small aircraft with that of automobile travel. For this research, productivity was set to 50% for both travel alternatives to maintain consistency between options (McGrath and Young, 2002). Multiple costs were calculated to better assess the situation including cost of air services, cost of employee travel time, and the offsetting value of productivity in route. The study included four passengers aboard the aircraft, with the pilot included, because this is comparable to an automobile. Trips that covered a distance of less than 100 miles were not included within this experiment because it is unlikely that an aircraft would be cost effective when used in these situations. Setting SATS costs at $260 per hour for DOC and $1158 per hour for LCC, and the automobile reimbursement costs at $0.32 per mile correlations between the two methods of travel could be determined. The study concluded that when the two alternatives were compared based on the automobile mileage and the SATS DOC the SATS system held an advantage of $1900 per trip ($475 per passenger), but when comparing based on the SATS LCC the automobile held an advantage of $677 per trip ($169 per passenger) (McGrath and Young, 2002). Combining all components into a single comparison yielded a large SATS advantage of $2462 per trip ($616 per passenger) (McGrath and Young, 2002). If productivity was not accounted for within the comparison SATS was advantageous for both DOC and LCC, as shown in Table 2.2 (McGrath and Young, 2002).

Table 2.2- Average Total Costs per Trip without Productivity Measures

	Automobile Cost	SATS DOC	SATS LCC
Average Total Cost per Trip	$10,110	$5,689	$8,266
Average Total Cost per Passenger	$2,527	$1,422	$2,066

This study resulted in the same conclusion as the airline comparison study. SATS was not proven to be a feasible travel alternative to the automobile until the economic value of the traveler's time was taken into account. The estimation of the value of a person's time consisted of utilizing the idea that while an employee is commuting to or from work the value of their time is approximately 60-75% of their wage rate, and when time constraints are prevalent these values can as much as double (McGrath and Young, 2002). Therefore, it is sufficient to forecast that as the congestion on the roadways and within the major hub-and-spoke airport system increases, the cost advantage of an alternative system will continue to correspondingly increase.

2.2 Characteristics and In-flight Effects of Icing

2.2.1 Characteristics of Icing in Atmosphere

The single weather condition of icing will be the focus of this research to keep the problem size reasonable. Airborne ice can be present in three forms: clear ice, rime ice, and mixed ice. Clear ice is formed when the conditions encountered by the aircraft are cumuliform clouds and/or large precipitation drops (Flight Standards Service, 1975). This type of icing is prevalent when the freezing rate of the liquid is gradual. The precipitation strikes the aircraft and rolls back across the surface freezing with a smooth texture that has minimal air between the ice and the aircraft. Rime ice occurs in clouds producing a light drizzle (Flight Standards Service, 1975). This type of icing freezes quickly on the aircraft allowing a bumpy surface to accumulate due to trapped air. The most common type of icing is mixed ice, which is a combination of clear and rime ice. This icing occurs when drop size is varied, or when liquid and frozen precipitation fall simultaneously (Flight Standards Service, 1975).

Clouds are a key element in icing type and potential, and should be avoided whenever possible. The type of clouds that are encountered can affect the drop size, drop distribution, and the consequences (Flight Standards Service, 1975). Newly formed parts

of clouds carry more precipitation than older segments and are therefore more hazardous. The majority of liquid is present near the upper surface of the cloud, and this area will yield the most ice accretion. Icing potential is the greatest right at freezing because water is most plentiful in clouds at this temperature (Flight Standards Service, 1975).

Any clouds that are present in freezing conditions could potentially cause icing, but specific variations are known to frequently produce certain conditions. Stratiform clouds carry icing potential down to approximately negative 15 degrees Celsius (Civil Aviation Authority, 2000). Lower level stratus clouds most commonly produce ice in the form of rime ice. The icing cloud parameter for stratiform clouds is "maximum continuous" due to spanning a wide horizontal distance but being limited in vertical size (Civil Aviation Authority, 2000). Thick, extensive stratified clouds are the most frequent producer of clear ice due to the large drop size produced from the abundance of water held in these clouds. Cumulus clouds can cause icing hazards to approximately negative 20 degrees Celsius, and below this temperature the precipitation will be frozen (Civil Aviation Authority, 2000). The icing cloud parameters for cumulus clouds is "maximum intermittent" since their spread is more vertical than horizontal (Civil Aviation Authority, 2000). Cumuliform clouds should be avoided whenever possible due to their ability to produce strong updrafts that carry water upwards (Civil Aviation Authority, 2000). These updrafts produce icing at higher altitudes than normally possible, and these conditions can accumulate a thick layer of ice very quickly if the temperature is below zero. Layer (stratus) type clouds seldom produce icing at greater altitudes than 5,000 feet above the freezing level (Flight Standards Service, 1975). Cirrus clouds are normally positioned with their bases above 20,000 feet; therefore the icing potential is slight due to the exceptionally cold conditions at that altitude (Civil Aviation Authority, 2000).

Many other factors affect the possibility of icing. Warm and cold fronts encourage icing conditions if the freezing rain is below the surface of the front (Flight Standards Service, 1975). Warm fronts can produce rime ice, freezing rain, and freezing drizzle. Cold fronts result in clear icing and large droplets, which are both most detrimental around zero degrees Celsius. Aircraft will also more frequently incur icing when flying over

mountainous terrain. Aircraft should avoid areas above mountain crests (up to about 5,000 feet above mountains) and the flight space on the windward side of mountains because these can posses the most ideal icing conditions (Flight Standards Service, 1975). Seasons can also affect the icing potential. The most prevalent time of year for the occurrence of in-flight icing correlates to the geographic location of the flight path. For the flights incorporated within this research winter will enable the most widespread icing because the freezing level is lower, therefore it will be encountered more frequently.

Although surface icing is the most discussed icing phenomenon, ground icing can greatly increase the risks involved with flying. Ground icing and frost occur at ground level, and must be identified before the aircraft attempts takeoff. The runway and aircraft should be clear of any signs of ice or large amounts of liquid precipitation before attempting flight. If water is blown by propellers or splashed by the wheels it may result in serious icing once the altitude increase causes a temperature decrease that is below the freezing level.

There are general limitations set to guide a pilot as to whether it is safe to attempt a take off in the present conditions. If the amount of dry snow exceeds 60 millimeters, or the amount of water or wet snow exceeds 15 millimeters take off should be postponed until these conditions are not present (Civil Aviation Authority, 2000). An aircraft attempting a landing in depths of water or wet snow as minimal as three millimeters could encounter adverse affects and hydroplaning (Civil Aviation Authority, 2000). It is also necessary to account for the condition of the overrun area and ensure safety if an overrun occurs.

When ground ice or frost is present deicing procedures are utilized to clear the precipitation. These procedures include utilizing a heat system to melt and evaporate precipitation or placing a chemical fluid on the infected area to clear the surface. Although "90 percent of the icing encounters are less than 50 miles in horizontal extent and none measured longer than 180 miles," (Civil Aviation Authority, 2000, p. 13) the brief encounters can still prove to be destructive and require much caution.

2.2.2 Weather Effects in Flight

Icing has unfavorable effects on flights and the ability to successfully complete predetermined flight paths. Avoiding icing conditions altogether is the primary goal for all aircraft. Therefore, if these conditions are encountered the new principal goal becomes escaping the conditions without severe damage. This is especially important for single and twin piston engines because these aircraft have limitations in regards to climbing above the conditions due to their loss of performance at high altitudes. The consequences are dependent on the icing conditions that are present. Even minimal icing conditions can have a serious effect on an aircraft in-flight. "Wind tunnel and flight tests have shown that frost, snow, and ice accumulations (on the leading edge or upper surface of the wing) no thicker or rougher than a coarse piece of sandpaper can reduce lift by 30 percent and increase drag up to 40 percent" (Civil Aviation Authority, 2000, p. 1).

When ice accumulates on the aircraft's surface it affects how the air moves over the wings, and the aerodynamic effects of the design are greatly decreased. This change in air flow affects the ability of the wing to lift the plane, and in turn reduces the handling capabilities of the aircraft. Sharp objects on the aircraft will accrue ice more rapidly because when air passes over blunt objects it tends to slow down which correlates to an increase in the temperature on those parts. The faster an aircraft is traveling the faster ice will adhere to its surface (Civil Aviation Authority, 2000).

Each of the three types of ice causes varying affects on the aircraft. Clear ice, which accumulates in freezing rain, alters the aerodynamic shape of the aircraft. Since the precipitation is in the form of freezing rain, the drops hit the surface and continue to flow rearward, which is known as flow back. The rate at which this precipitation freezes to the surface is dependent on the outside temperature; therefore the drops extend the furthest when the temperature is right at freezing. At this time the ice is also in the smoothest form because it has frozen in a solid sheet. This also correlates to the most difficult removal procedure because it is frozen directly to the aircraft with little or no air trapped in between.

Rime ice accrues when a rapid freezing occurs. The droplets that normally produce rime ice are small and do not possess enough liquid to carry out the flow back property. This type of icing creates a rugged surface due to the quick freezing rate; this in turn is more efficient in reducing the aerodynamics of the plane. It is not as heavy as clear ice because much of its volume is made up of the air between the surface and ice. This type of icing is most frequently found on leading edges, which make it more difficult to produce the airflow into the engine intake (Long et al, 2001). Rime ice is the most probable between negative 15 and negative 20 degrees Celsius.

Mixed ice is a combination of clear ice and rime ice because it accumulates when a variation in drop size is encountered. Therefore, the effects on the aircraft are a mixture of the previous two icing forms. Mixed ice is the most common form of icing because specific conditions must be present for pure forms of clear and rime ice to exist. The common outside air temperature ranges for each of the three types of icing can be seen in Table 2.3.

Table 2.3- Temperature Ranges for Ice Formation

Outside Air Temperature Range	Icing Type
0 ºC to -10 ºC	Clear
-10 ºC to -15 ºC	Mixed Clear and Rime
-15 ºC to -20 ºC	Rime

Source: U.S. Department of Transportation, 1996

Structural icing could result in a roll upset, wing tip stall, or tail stall. These occur when there is severe icing. Severe icing is prevalent when, "the rate of accumulation is such that the de-icing/anti-icing equipment fails to control the hazard" (Civil Aviation Authority, 2000, p. 6). Despite the increase in technology roll excursion incidents have been on an increase, and it is thought to be due to the fact that more severe weather conditions are occurring (Long et al, 2001). Some of the most typical indicators before a roll upset are that icing conditions are present, ice is accruing on surfaces, and the speed of the aircraft is decreasing even when all other control elements are stationary.

Roll upset is caused when external forces cause the ailerons, two movable control surfaces on the wings that aid in steering the aircraft, to lose all or part of their function. It is caused by supercooled large droplets (SLD). SLD can be identified in freezing conditions if visible ice accumulates on the wing surfaces, windows, spinner, or airframe (U.S. Department of Transportation, 1996).

A wing tip stall is a consequence of ice accumulation on the aircraft and occurs on a leading edge that is prone to ice. This phenomenon often starts at the tip of the wing because of the sharp edge and the lower temperature. The outer wing also tends to stall quickly because the propeller protects the inner wing (Civil Aviation Authority, 2000).

A tail stall is often another characteristic of severe icing. It is caused by ice accumulation on the horizontal portion of the tail, and causes control problems. This can go undetected until it is irreversible due to the position and poor visibility of the tail. The result consists of a drastic nose down pitch (Civil Aviation Authority, 2000). The tail surface collects ice at a faster rate then the wings; therefore it can be deceiving to gauge the entire surface based on the visible portions of the aircraft. The symptoms for a tail stall include alterations in the pitch of the aircraft.

If the pilot is unaware of the hazards present in-flight, it will increase the danger. Research has been completed on aiding pilots with the detection of hazards (Spirkovska and Lodha, 2002). The research is on providing a simplistic user interface that will give pilots the necessary information regarding their flight path. This system would incorporate making textual documents easier to read, and obtaining information that is essential to the specific aircraft and flight path. This will be accomplished by allowing the pilot to enter their airspeed, altitude, departure time, and preferred display output, and then the system will provide essential information about their specific flight path. This will allow the pilot to make necessary corrections before entering unfavorable weather conditions.

Chapter 3- Methodology

This chapter will describe the procedure implemented to complete the research. The purpose of this research is to determine the dispatch reliability of small aircraft in regards to the weather condition of icing through assessing the probability the flight will be able to take place within the travel hours of 0600 to 2000 local time. It will provide detailed information in regards to the activities that are required to complete the assessment.

The FAA allows certain aircraft to fly into icing conditions if the conditions are not severe and the aircraft is equipped with anti-icing instruments. However, for the purpose of this research a flight will not be attempted if any icing conditions are forecast. Instances do occur when the conditions change rapidly and aircraft will enter icing circumstances even though the flight path was clear at take off. If this occurs the pilot should exit the conditions immediately while controlling the plane manually instead of engaging the autopilot.

3.1 Problem Overview

The determination of the effect of icing on the dispatch reliability of small aircraft was broken into three steps. First, once the AIRMETs data are collected they need to be parsed to only include the necessary information from within the icing AIRMETs. Second, once these data are stored in an appropriate form they will be utilized to create a plot of the AIRMET polygon. This plot will be interwoven with a plot of the flight trajectories for the origin and destination pairs to determine when intersections are encountered. Third, once these intersections are determined the number of times the flight path intersects forecasted icing conditions can be determined. These intersections define the instances when a flight cannot be completed.

3.1.1 Data Collection and Data Parsing

In order to initiate the assessment, many modifications need to be made to the raw data. The data were obtained from the National Oceanic and Atmospheric Administration (NOAA). It consisted of AIRMETs for the collection time period, November through May for the years 2001 to 2003. A translation of how to read an AIRMET can be found in Appendix A.

The Zulu AIRMETs were intermixed within the entire data set. Therefore the initial task was to parse out the Zulu AIRMETs. After the new file was created that consisted of the Zulu AIRMETs, each individual icing AIRMET could be parsed to extract the information required for analysis. Three main pieces of information were relevant: the starting time and ending time of the weather condition, the shape and location of the AIRMET (which was incorporated as a set of data points that were connected to create a polygon), and the minimum and maximum altitude of the icing conditions. The maximum altitude of the AIRMET was listed as a single value, but the minimum altitude was frequently listed as an assortment of points connected by jagged lines. For purposes of this research the program read through all of the points and determined the minimum value. This value was stored as the minimum altitude to create a flat bottom for the polygon.

AIRMETs are issued at a point in time, but they may be corrected (COR) or amended (AMD) at any subsequent point in time until the end time of the condition. This needed to be accounted for in order to eliminate the possibility of double counting a single AIRMET.

The raw data were restructured in a structure array to allow for simple referencing. At this point the data were extracted and stored, but the boundary points were still were not in the format necessary to graphically represent the AIRMETs. Therefore, the last step in the data collection and data parsing phase was to transform the polygon boundary points into (longitude, latitude) data points for plotting purposes.

3.1.2 Plotting Trajectories of Flight Paths and AIRMETs

To determine the proportion of time that small aircraft will be grounded due to unsuitable icing conditions a graphical representation was created to determine the intersection of a weather condition and an aircraft flight path. The AIRMET polygons and flight trajectories were plotted on a single graph in three dimensions to determine instances when a fight path encountered an icing AIRMET. The initial assessment was completed based on a cruising altitude of 12,000 feet. However, aircraft have the advantage of being able to fly under adverse weather conditions if the terrain allows for them to reduce their cruising altitude. In order to account for this, analysis was repeated for those flights that were categorized as a "no-go" at 12,000 feet. If a flight was a "no-go" at 12,000 feet it was reanalyzed at a maximum cruising altitude of 10,000 feet. If a flight still remained a "no-go" at 10,000 feet it was reanalyzed at a maximum cruising altitude of 8,000 feet. Finally, if it was still a "no-go" at 8,000 feet it was reanalyzed at a maximum cruising altitude of 6,000 feet. The maximum cruising altitude could not be reduced further than 6,000 feet due to the terrain. The plot rotates through the set of 3,280 icing AIRMETs in the data set.

3.1.3 Analysis of Reliability based on Icing AIRMETs

The reliability of small aircraft in regards to the weather condition of icing was evaluated once the intersection points of the AIRMET polygon and flight path were established. Each day was divided into three periods: morning, afternoon, and evening. The morning period consisted of flights from 0600 to 1000, the afternoon consisted of flights from 1000 to 1500, and the evening consisted of flights from 1500 to 2000. Each of these periods was broken down into hour increments for assessment purposes, and then the total number of intersecting AIRMETs was summed over the entire period to determine the percentage of time a flight was considered a "no-go". The assessment was a straightforward probabilistic measure of how often in a semimonthly, three-week, and monthly period flights were not able to be flown based on the icing conditions forecasted

within the pre stated morning, afternoon, and evening periods. Frequency histograms were created to enable visual representation of how often and when these instances do hinder small aircraft flights.

3.2 Detailed Description of Completed Work

As described in the subsequent sections this work has been broken into three main components. They consist of data collection and data parsing, graphical representation of the flight trajectories and icing AIRMETs, and reliability analysis. Section 3.2.1 provides a detailed summary of the programming used to extract the necessary information for the data collection and data parsing phase. Section 3.2.2 describes the procedures that were implemented to graph the flight trajectories and AIRMETs and to determine the intersection points. Section 3.2.3 indicates the methods used to determine the reliability measures of small aircraft with respect to icing conditions in the winter months. Matlab 6.5.1 was used to complete the data parsing and plotting aspects of this research.

3.2.1 Data Collection and Data Parsing

The initial program was written to extract the icing AIRMETs from the entire data set. This was accomplished by setting an indicator for the start of a Zulu AIRMET and reading until a subsequent indicator was found that verified the conclusion of the weather condition. This program code can be viewed in Appendix B, and Figure 3.1 is a diagram of the logic.

Next, a program was created to parse out the necessary information from the icing AIRMETs. This information consisted of the start and end time of the condition, the polygon that represents the boundaries of the condition, and the minimum and maximum altitude. The extraction of this information was detailed because the AIRMETs were not uniformly structured; therefore numerous indicators were required to extract this data from the entire data set. This information was compiled in a structure array because this allowed all the information to be stored in a single reference that was beneficial for

plotting purposes. This program can be viewed in Appendix C, and Figure 3.2 is a diagram of the logic for this process.

Subsequently, a program was written to detect when a COR or AMD was present within an AIRMET. If a COR or AMD indicator was found, the program rewrote the end time of the initial AIRMET to be equal to the start time of the correction or amendment AIRMET. This eliminated the time lapse between the two AIRMETs; therefore eliminating the time period in which they would both be considered an intersection. This program for the flight path of BOS to CLE at 12,000 feet can be viewed in Appendix D, and Figure 3.3 is a diagram of the logic for this process.

The final data parsing program was written to transform the polygon boundary points into (longitude, latitude) data points that could be plotted. The data points for the polygons are navigational aid (NAVAID) points, but each point could also incorporate a variation from the indicator. The variation was structured as a distance and direction from the specified point (eg: 70SW PQI). The possible directions and their correlating degree values are indicated in Table 3.1.

Table 3.1: Degree Values for Directional Deviation from NAVAID Points

Direction:	Degree Values:
N	0/360
NNE	22.5
NE	45.0
ENE	67.5
E	90.0
ESE	112.5
SE	135.0
SSE	157.5
S	180.0
SSW	202.5
SW	225.0
WSW	247.5
W	270.0
WNW	292.5
NW	315.0
NNW	337.5

If a variation existed for a given point the direction was matched to the corresponding degree value, and then the new latitude and longitude was determined through the use of Equation 3.1.

$$[\text{NEWLAT, NEWLONG}] = \text{reckon}\ (\alpha, \beta, \delta, \gamma) \tag{3.1}$$

where,
reckon = Matlab command for determining a new distance
α = Latitude of the NAV point for the given point
β = Longitude of the NAV point for the given point
δ = Number to degree for the numerical distance from NAVAID
γ = The corresponding degree value for the direction of variation from NAVAID

To extract the longitude and latitude values of the NAVAID points each indicator was matched to the same indicator on a complete list of the NAVAID points, and the corresponding longitude and latitude values for that point were extracted. This was accomplished by setting the indicator as a variable, and comparing it to each entry in the file until a match was determined. To save time, once a match was found the loop in the program terminated so that it would not continue looking through the entire list for every point. If a variation existed the above equation was utilized to obtain a (longitude, latitude) data point. The longitude and latitude points were now uniform for indicators with and without variations. The longitude and latitude values were incorporated within the structure array as separate entities for recall. This was repeated for each point in the polygon.

This entire process was completed for each polygon that was present for each AIRMET. There were 1,696 time periods due to the four 6 hour increments per day and 212 days of data for two years. The number of polygons in each time period ranged from one Zulu AIRMET to five Zulu AIRMETs for the data used in this research. This program can be viewed in Appendix E, and Figure 3.4 is a diagram of the logic for this process.

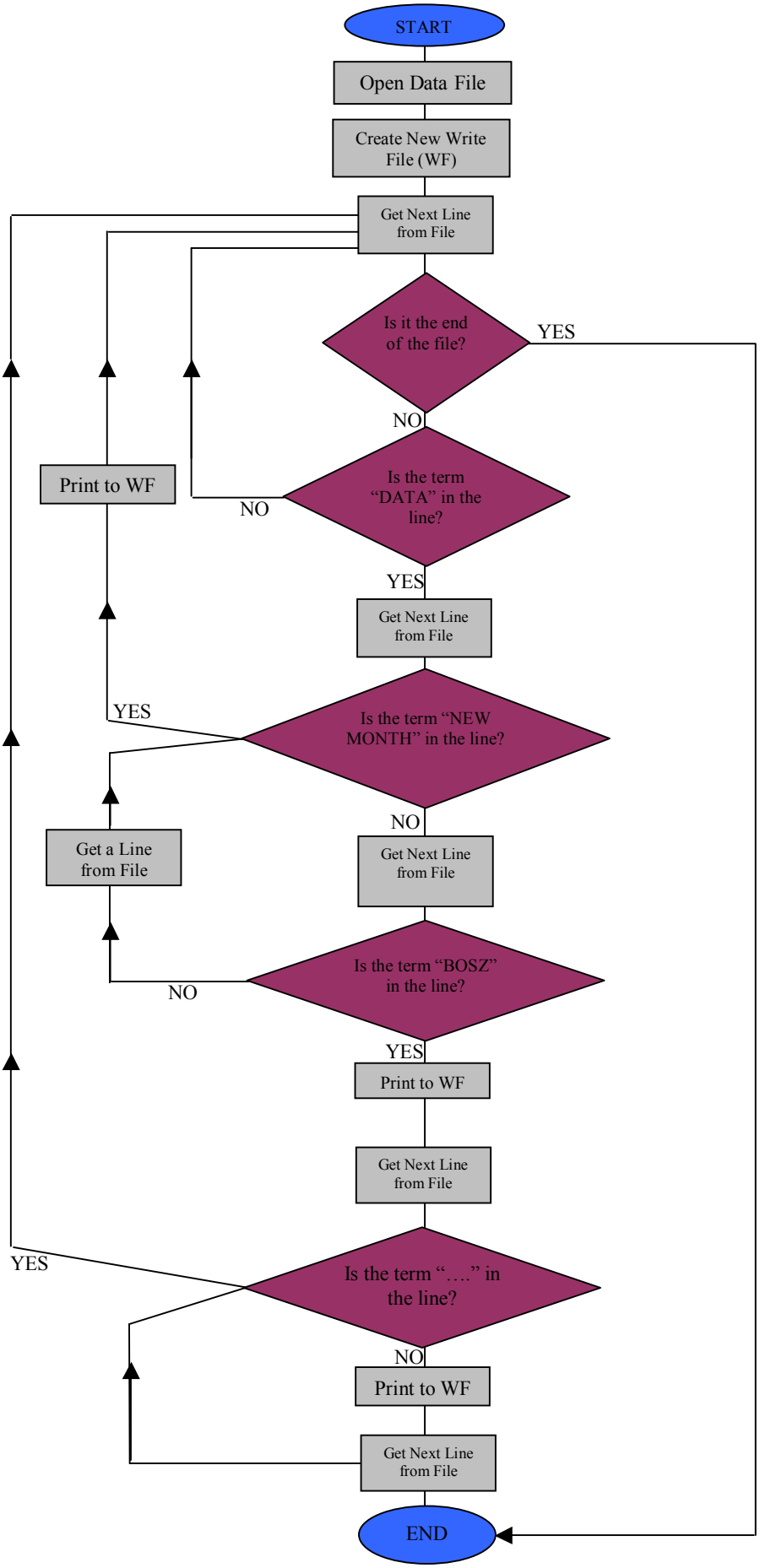

Figure 3.1: Diagram of Logic for Initial Zulu AIRMET Parsing Program for BOS Region

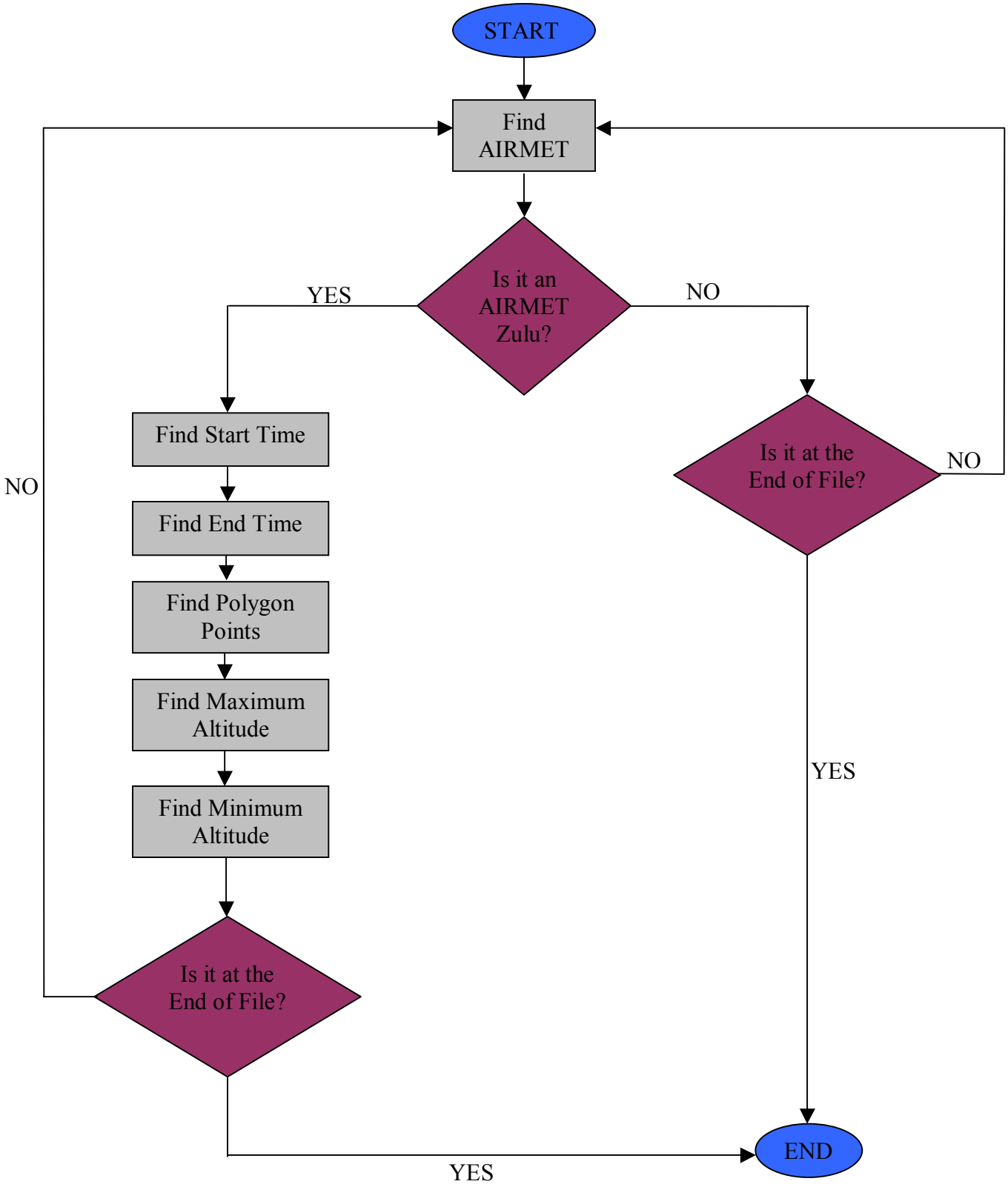

Figure 3.2: Diagram of Logic for Detailed Extraction of Data from AIRMET

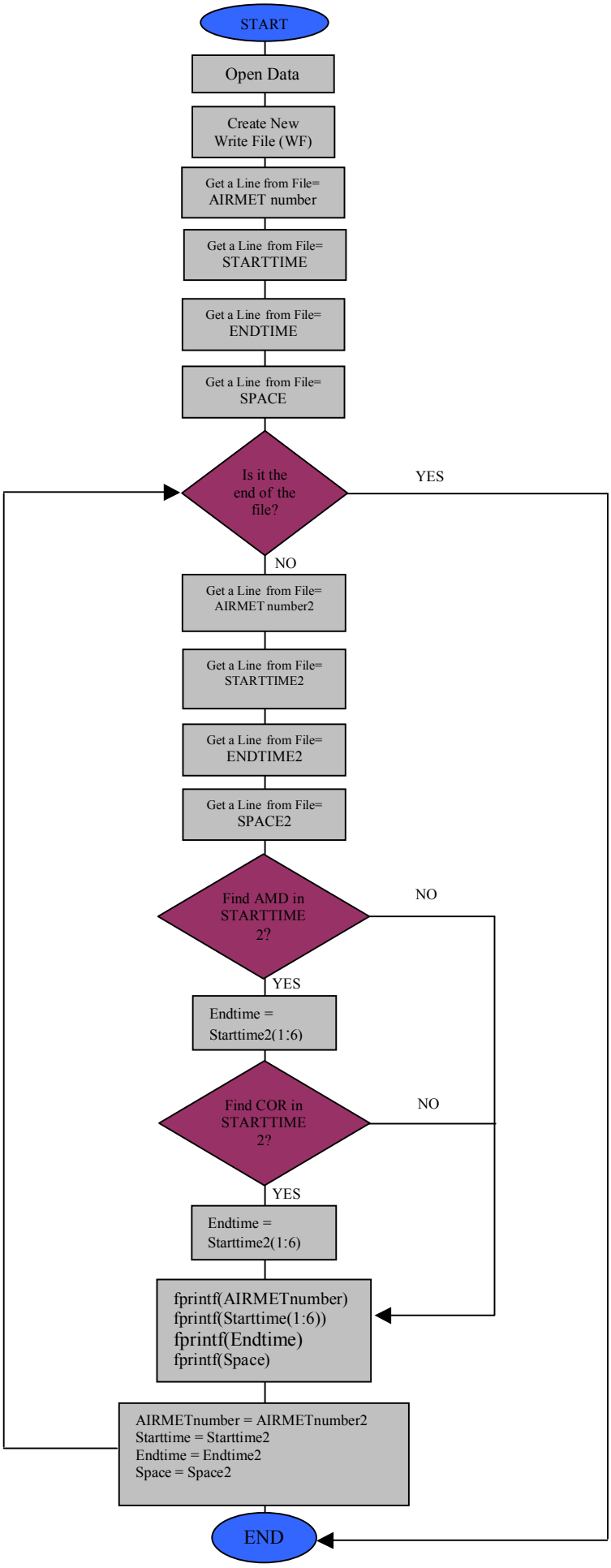

Figure 3.3: Diagram of Logic for Amendment and Correction Inclusion

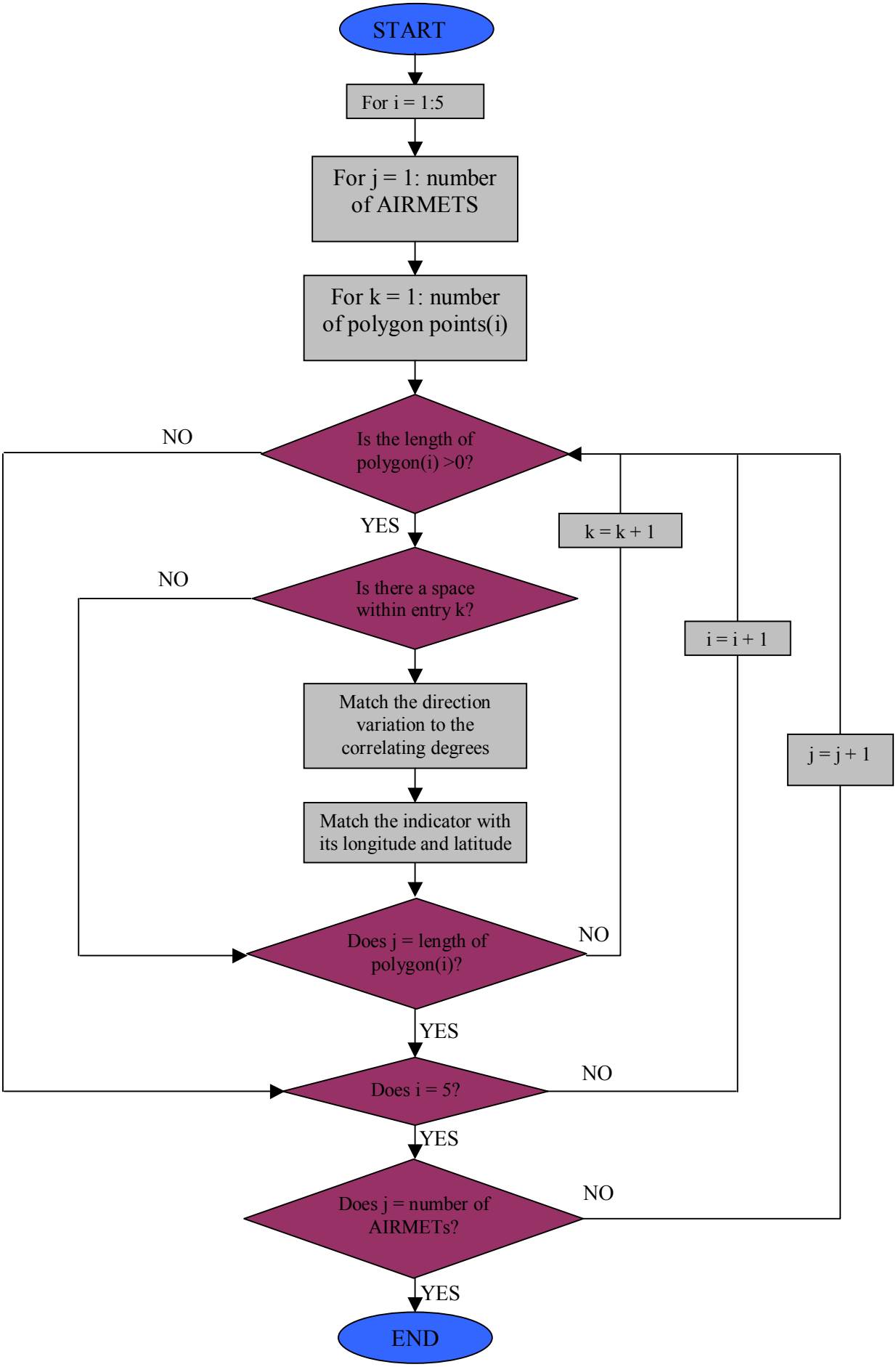

Figure 3.4: Diagram of Logic for FindNAV Program

3.2.2 Plotting Trajectories of Flight Paths and AIRMETs

The AIRMETs were initially plotted in two dimensions over a map of the northeastern portion of the United States to ensure that the program was running correctly. Subsequently, a program created by Hojong Baik (Baik, 2003) was utilized for plotting the flight trajectories on a separate graph. This program plotted the flight trajectories utilizing a set of 30 waypoints between the origin and destination. Each waypoint consisted of a longitude, latitude, and altitude. The program was much more detailed than required for the scope of this research. It was initially created to be able to assess the fuel usage, flight time, and trajectory for any origin and destination airports. Since the origin-destination pairs in this research are limited to three cities, this coding was altered to plot the trajectories for these three flight paths without prompt from the user of the program. Also, the only output required was the graphical representation of the flight path; therefore all the other information was eliminated to make the program more time efficient.

The flight trajectory program (Baik, 2003) was designed to plot the flight trajectories in three-dimensions; therefore no additional assessment was required in regards to this. However, the two dimensional AIRMET objects needed to be expanded to three dimensions in order to be able to determine intersection points. This was accomplished by plotting one of the AIRMET polygons at the minimum altitude and another one at the maximum altitude, and then a surface was created by connecting the two polygons.

Once these two programs were successfully running individually they were combined into a single program that combined the two plots. This was necessary to be able to determine the intersection points. An example of this plot in two-dimensions and three-dimensions is shown in Figure 3.5a and Figure 3.5b. The red lines are the three flight trajectory paths, and the teal, magenta, and blue objects represent three forecasted icing conditions present in the correlating AIRMET.

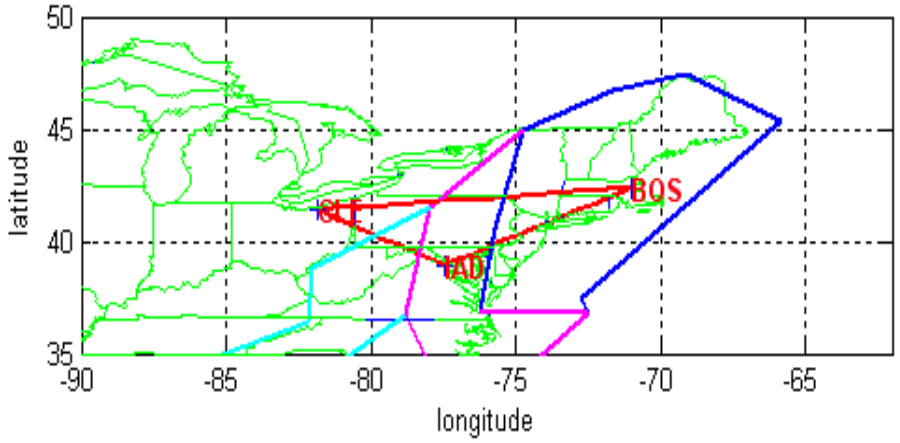

Figure 3.5a: Two-Dimensional Graph of Flight Trajectory and AIRMET

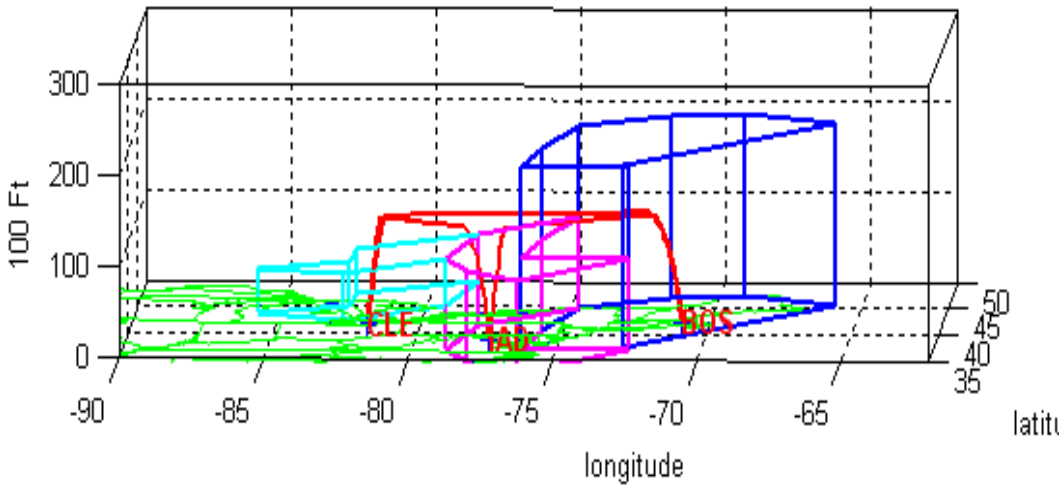

Figure 3.5b: Three-Dimensional Graph of Flight Trajectory and AIRMET

This program can be viewed in Appendix F, and Figure 3.6 is a diagram of the logic for this process.

The next phase of plotting flight trajectories and AIRMETs was to determine the intersection points at the maximum flying altitude of 12,000 feet. This task required multiple programs to reach the final results. Each of these programs was written for each flight path (BOS to CLE, BOS to IAD, and CLE to IAD). The initial program was written to determine when there were intersections in any of the five possible polygons

for each AIRMET. The program was written for five polygons because this is the maximum number of polygons present within an AIRMET for the data within this research. There is no known command in Matlab to determine intersections in regards to three dimensions; therefore the three dimensions were broken up to aid in the assessment. This was accomplished by first only incorporating the two dimensions of longitude and latitude through the use of the inpolygon command, and then if there was an intersection based on these two components the altitude was incorporated. If a two-dimensional intersection was determined the intersecting waypoints within the flight trajectory path were inspected to determine if they also fell between the minimum and maximum altitude for the corresponding AIRMET condition.

The same process as described for the 12,000 feet cruising altitude program was implemented to determine intersections for the maximum flying altitude of 10,000 feet, 8,000 feet, and 6,000 feet. The only difference was the flight trajectory altitude being read into the command. The program for BOS to CLE at a maximum flying altitude of 12,000 feet can be viewed in Appendix G, and Figure 3.7 is a diagram of the logic for this process.

Some output issues needed to be addressed. The output could possibly indicate that a single AIRMET intersected the flight trajectory path multiple times. This was due to the fact that up to five polygons could be present, each indicating a different icing forecast, for each AIRMET; and out of these five polygons each could have up to 30 way points from within the flight trajectory indicating an intersection. In order to eliminate this redundancy an additional program was written to remove all but a single reference to each AIRMET. It was also determined that often times in the data set, an AIRMET could be listed up to three times containing the same information. This would falsify the results if each of these were considered an intersection. Therefore, this program also had to determine if there was an exact match between the current and previous AIRMET, and if a match was found the program eliminated the repetitive output. Finally, to keep the data in a useful structure, this program summed the amount of intersections for each month

and wrote the totals to a separate file for storage. This program can be viewed in Appendix H.

To aid in the determination of the reliability measure, the data needed to be classified further. Therefore, the intersections for each flight trajectory were classified into time periods that were equal to the start time and end time of the AIRMET. AIRMETs are issued on Zulu time, which is five hours ahead of Eastern Standard Time and four hours ahead of Eastern Daylight Savings Time. For the years of 2002 and 2003 daylight savings time started on April 6th and 7th respectively. Therefore, the remainder of the month of April and the month of May are in this category. This program involved searching through a list of the possible start times and end times that were relevant to the 0600 to 2000 time period. The date of the specific AIRMET was analyzed to determine if four or five hours needed to be subtracted from the time to obtain the local time. The new local time period was then placed into its corresponding time category and stored. The total number of AIRMETs in each period were summed, and this number was written to a file for storage. This program was run for each individual month (14 times) at the varying maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet), and it output the intersecting AIRMETs start time and end time on a daily basis. An example of the program for the first two weeks of November 2001 for the flight path of BOS to CLE at the cruising altitude of 12,000 feet can be viewed in Appendix I.

There is also a prospect that when a flight does not intersect an AIRMET it is due to the fact that it is flying above the adverse condition. In order to be able to verify how often this occurred, a program was created to output the flights in which the maximum cruising altitude was above the AIRMETs maximum altitude. This was accomplished by evaluating a list of all of the AIRMETs that did not intersect with the flight trajectories. Each of the thirty waypoints in the flight trajectory were tested in comparison to the maximum altitude of the AIRMET. If there was a point when the flight trajectory altitude was greater than the AIRMET maximum altitude that AIRMET was recorded. Each AIRMET could be recorded multiple times, up to 30, based on the number of occurrences. Therefore, an addition to this program compared the lines of output to look

for duplications, and it eliminated the redundancies in an additional file. This program can be viewed in Appendix J, and Figure 3.8 is a diagram of the logic for this process.

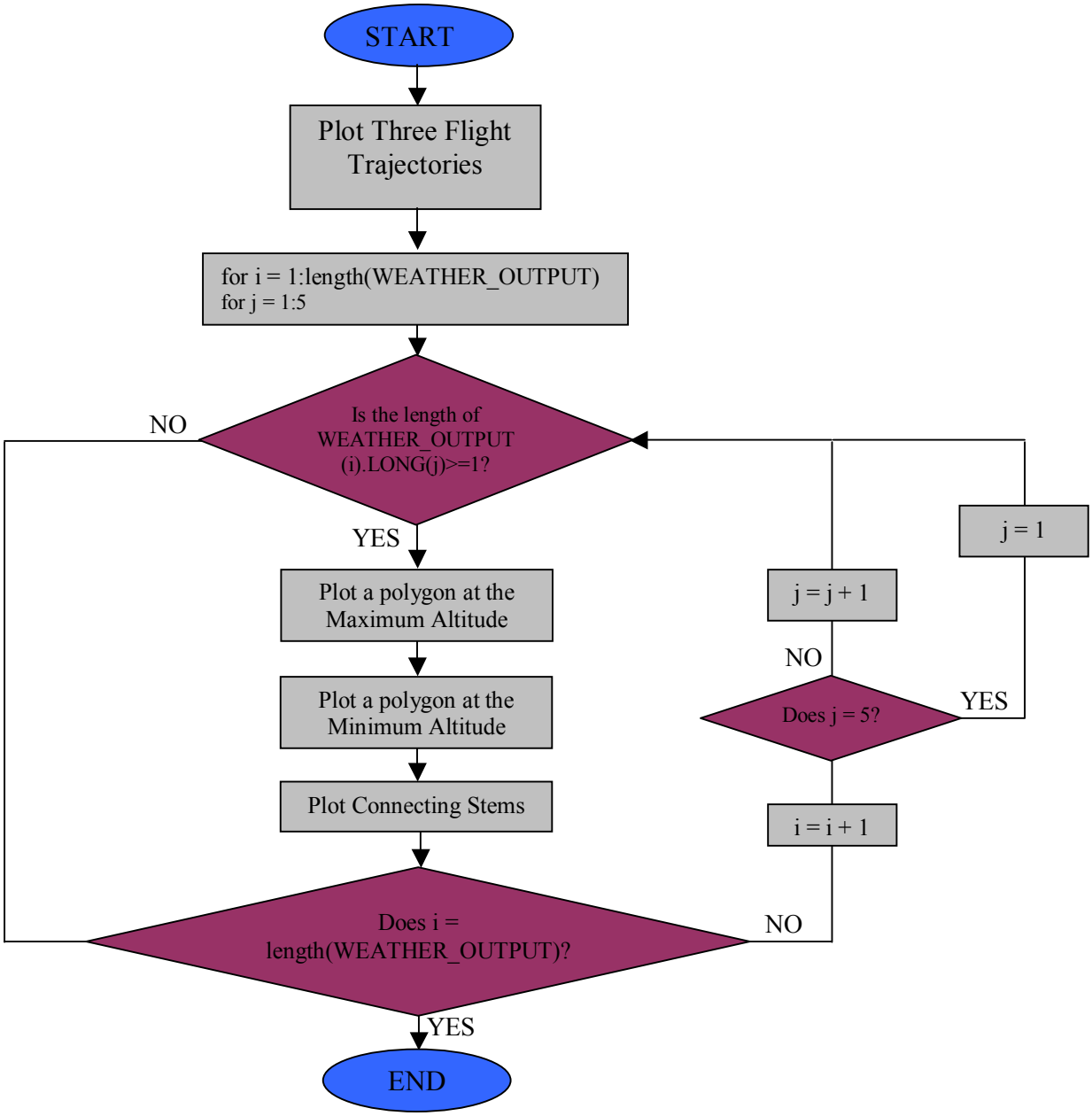

Figure 3.6: Diagram of Logic for Plotting Program

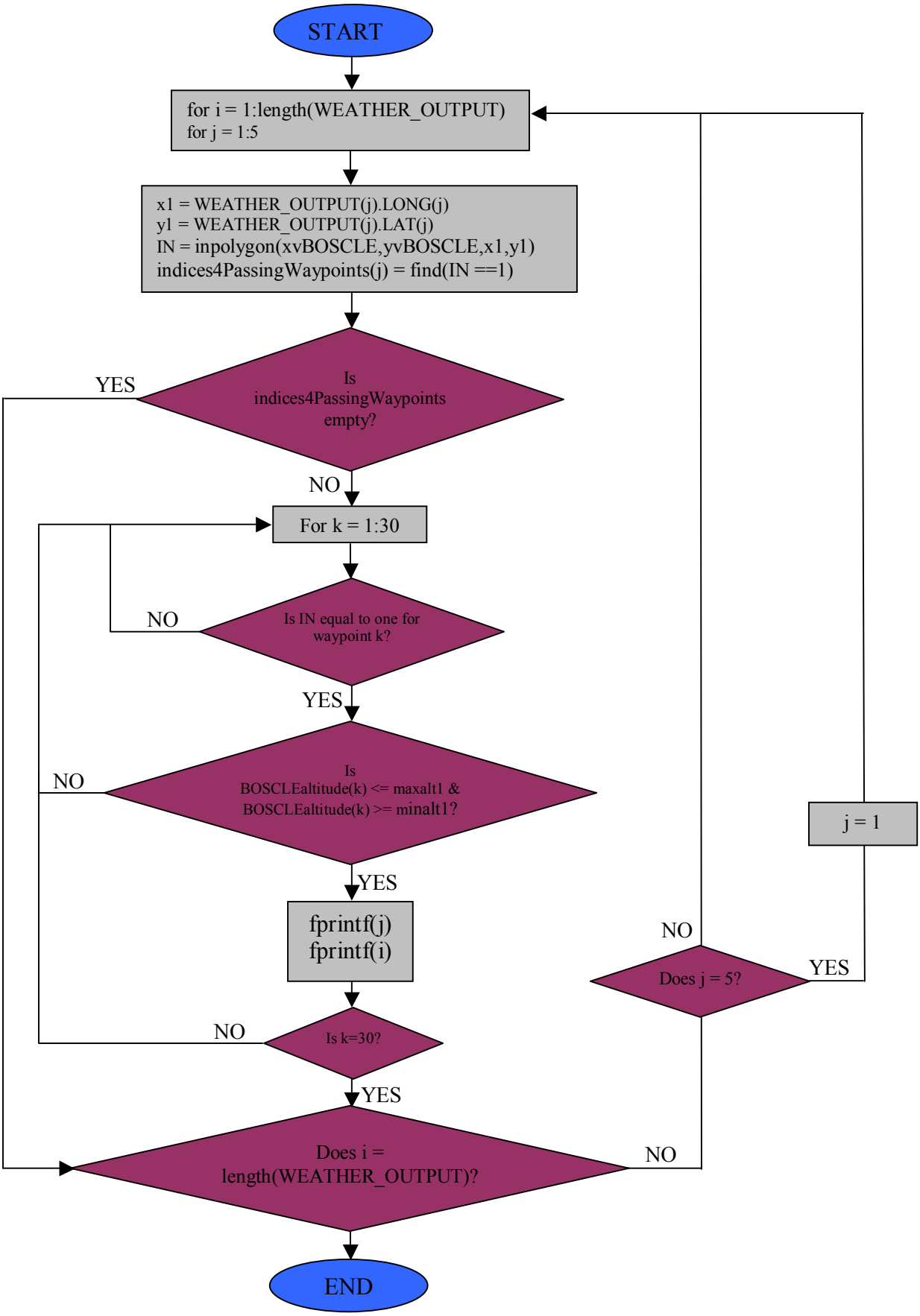

Figure 3.7: Diagram of Logic for Intersection Program

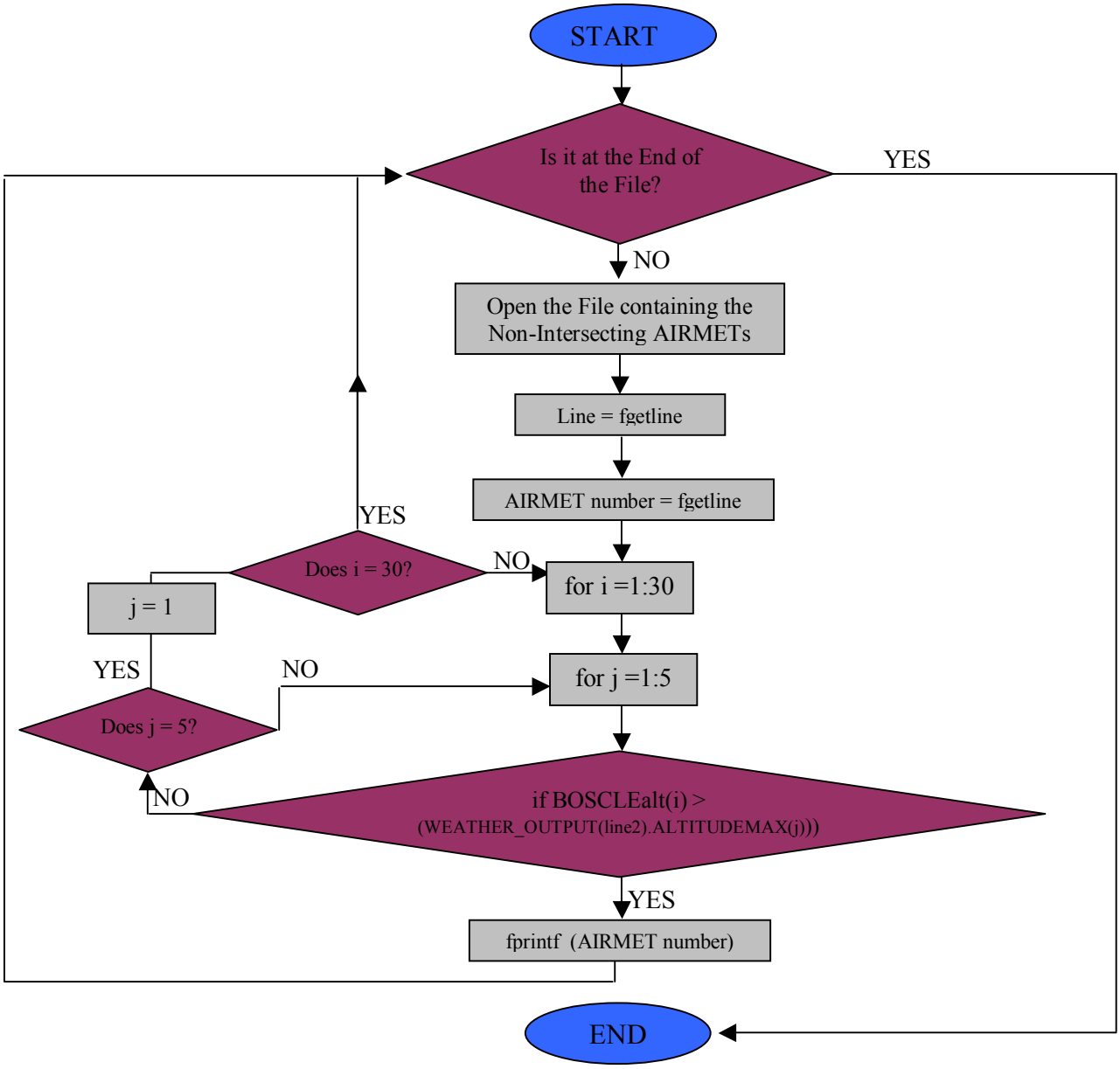

Figure 3.8: Diagram of Logic for Finding Flights Flying Over AIRMETs

3.2.3 Analysis of Reliability based on Icing AIRMETs

Once the intersection points of the AIRMET polygon and flight path were established it was possible to determine how often in a semimonthly, three-week, and monthly period the pre specified aircraft flights would not be made within the morning, afternoon, and evening periods. Since AIRMETs are issued on a six-hour increment, the weather at every given hour is not precisely known, however the projected weather was used for this research. Each AIRMET has a start time and end time; therefore each hour of the day could still be evaluated through the incorporation of this information. If a specific intersecting icing AIRMET was issued from 9:45 am until 4:00 pm the flight was considered a "no-go" for the seven hours incorporated within that time frame. This methodology was used because it is known that the flights from the specified origin-destination pairs were at least one hour in duration. Therefore, if icing conditions were encountered within that time period the flight should not be attempted.

The semimonthly, three-week, and monthly time periods were used in order to reduce the bias of the estimates due to the limited sample size of the data. The credibility of the reliability measures would have been compromised by reducing the time periods further due to having only two years worth of data, which corresponds to only two data points for each hour. The semimonthly, three-week, and monthly periods were derived by dividing up the weeks incorporated within the months of November to May. The dividing points and number of inclusive days for each of these time periods can be seen in Table 3.2, Table 3.3, and Table 3.4.

Table 3.2: Number of Days in Monthly Time Periods

Monthly Period	Number of Days
November	30
December	31
January	31
February	28
March	31
April	30
May	31

Table 3.3: Number of Days in Three-Week Time Periods

3-Week Period	Number of Days
Nov 1 to Nov 21	21
Nov 22 to Dec 12	21
Dec 13 to Jan 3	21
Jan 4 to Jan 24	21
Jan 25 to Feb 14	21
Feb 15 to Mar 7	21
Mar 8 to Mar 28	21
Mar 29 to Apr 18	21
Apr 19 to May 9	21
May 10 to May 31	22

Table 3.4: Number of Days in Semimonthly Time Periods

Semimonthly Period	Number of Days
Nov 1 to Nov 15	15
Nov 16 to Nov 30	15
Dec 1 to Dec 15	15
Dec 16 to Dec 31	16
Jan 1 to Jan 15	15
Jan 16 to Jan 31	16
Feb 1 to Feb 15	15
Feb 16 to Feb 28	13
Mar 1 to Mar 15	15
Mar 16 to Mar 31	16
Apr 1 to Apr 15	15
Apr 16 to Apr 30	15
May 1 to May 15	15
May 16 to May 31	16

The daily intersecting AIRMETs data that were stored with respect to the start time and end time of the conditions were transferred to a spreadsheet that would sum the number of intersecting AIRMETs for any given hour over the semimonthly, three-week, and monthly period. Frequently the AIRMETs were issued at time periods that overlapped the previous AIRMET. Therefore, for each day it needed to be verified that each hour slot had no more then one intersection listed. Each month was established individually,

and shown in Figure 3.10 is the month of November for both years data (2001 and 2002) for the flight path BOS to CLE at a maximum cruising altitude of 12,000 feet. Once the data were entered for all the monthly periods, it was shifted to determine the sum within the predetermined three-week and semimonthly periods. At this point all the required information was available to determine the effect of icing on the dispatch reliability of small aircraft.

1-Nov	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000
1	1	1	1	1										
2														
3	1	1	1	1										
4														
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1										
7														
8										1	1	1	1	1
9				1	1	1	1	1	1	1				
10					1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1				
12				1	1	1	1	1	1	1				
13														
14				1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1				
16														
17														
18										1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	1	1				
22														
23														
24				1	1	1	1	1	1	1	1	1	1	1
25				1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1										
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	1	1	1	1	1	1	1	1	1	1				
29														
30										1	1	1	1	1
SUM	**12**	**12**	**12**	**17**	**14**	**14**	**14**	**14**	**14**	**17**	**11**	**11**	**11**	**11**
2-Nov	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5				1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1				
8														
9				1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20														
21				1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29														
30	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SUM	**24**	**24**	**24**	**27**	**27**	**27**	**27**	**27**	**27**	**27**	**26**	**26**	**26**	**26**
TOTAL	**36**	**36**	**36**	**44**	**41**	**41**	**41**	**41**	**41**	**44**	**37**	**37**	**37**	**37**

Figure 3.9: Intersecting AIRMETs Placed in Time Slot for November (Monthly), BOS to CLE-12,000

Chapter 4- Results and Analysis

This chapter provides detailed results from the procedures outlined in chapter three. The main focus is on the results from the reliability analysis; however results from the Matlab programs for the data parsing and graphical representations are also included.

4.1 Data Parsing Results

A sample of the initially collected data is provided in Figure 4.1. This sample provides the first set of AIRMETs on November 1, 2001. The time period is 0245 to 0900 Zulu time, and there is one icing (Zulu) AIRMET, one turbulence (Tango) AIRMET, and one low visibility (Sierra) AIRMET.

```
NWS SRRS PRODUCTS FOR:
        2001110100 to 2001113023

WAUS1 KBOS 010233
WA1Z
 BOSZ WA 010245
AIRMET ZULU FOR ICE AND FRZLVL VALID UNTIL 010900
.
AIRMET ICE...ME NH VT MA RI CT NY LO PA NJ AND CSTL WTRS
FROM 60NW PQI TO YSJ TO ACK TO CYN TO ETX TO BUF TO YOW TO 60NW
PQI
OCNL MOD RIME/MXD ICGICIP BTN FRZLVL AND FL220. FRZLVL 020-060.
CONDS ENDG SW SYR-HTO LN 07-09Z CONTG RMNDR BYD 09Z THRU 15Z.
.
FRZLVL...SFC-040 N OF YSC-50E ENE SLPG TO 080-120 SW DXO-AIR-ORF
LN.
....

WAUS1 KBOS 010233
WA1T
BOST WA 010245
AIRMET TANGO FOR TURB VALID UNTIL 010900
.
AIRMET TURB...NY LO PA OH LE
FROM MSS TO ALB TO EWC TO CVG TO FWA TO DXO TO YYZ TO MSS
OCNL MOD TURB BLW 060 DUE TO INCRG SWLY LOW LVL WNDS. CONDS CONTG
BYD 09Z THRU 15Z.
....

WAUS1 KBOS 010245
BOSS WA 010245
AIRMET SIERRA FOR IFR AND MTN OBSCN VALID UNTIL 010900
.
AIRMET IFR...ME NH VT NY MA
FROM HUL TO 50E BGR TO 40SW ENE TO 30S ALB TO MSS TO HUL
OCNL CIG BLW 010/VIS BLW 3SM IN CLDS/PCPN BR/FG. CONDS SPRDG EWD
AND CONTG BYD 09Z ENDG NY PTN 09-11Z CONTG RMNDR THRU 15Z.
.
AIRMET MTN OBSCN...ME NH VT MA NY
FROM 70NNW PQI TO MLT TO CON TO ALB TO SYR TO MSS TO 70NNW PQI
MTNS OCNL OBSC IN CLDS/PCPN BR/FG. CONDS SPRDG NEWD AND CONTG BYD
09Z ENDG NY PTN 11-13Z CONTG RMNDR THRU 15Z.
....
```

Figure 4.1: Sample of Initial Collected Data

The first program in the data collection and data parsing phase was written to extract the icing AIRMETs and place them in a separate file. Therefore, the data in Figure 4.1 would be reduced to the data in Figure 4.2.

```
WAUS1 KBOS 010233
WA1Z
 BOSZ WA 010245
AIRMET ZULU FOR ICE AND FRZLVL VALID UNTIL 010900
.
AIRMET ICE...ME NH VT MA RI CT NY LO PA NJ AND CSTL WTRS
FROM 60NW PQI TO YSJ TO ACK TO CYN TO ETX TO BUF TO YOW TO 60NW
PQI
OCNL MOD RIME/MXD ICGICIP BTN FRZLVL AND FL220. FRZLVL 020-060.
CONDS ENDG SW SYR-HTO LN 07-09Z CONTG RMNDR BYD 09Z THRU 15Z.
.
FRZLVL...SFC-040 N OF YSC-50E ENE SLPG TO 080-120 SW DXO-AIR-ORF
LN.
....
```

Figure 4.2: Sample of Data with Only Icing AIRMETs

Next the relevant information needed to be parsed out of the icing AIRMET. This data included the start time and end time, the polygon points, and the minimum and maximum altitude. The data in Figure 4.2 would be reduced to the data in Figure 4.3a. A translation of the relevant information is shown in Figure 4.3b.

```
010245
010900

60NW PQI YSJ ACK CYN ETX BUF YOW 60NW PQI
220
020
```

Figure 4.3a: Sample of Relevant Data from Icing AIRMET

```
Start Time: 01 = day of month, 0245 = time of day
End Time:  01 = day of month, 0900 = time of day

NAVAID Points for Polygon Boundaries
Maximum Altitude (220 = 22,000 Feet)
Minimum Altitude (020 = 2,000 Feet)
```

Figure 4.3b: Translation of Relevant Data from Icing AIRMET

Figure 4.4 is the corresponding structure array in Matlab that is used for plotting purposes. There is enough space in this structure array for up to five different polygons

at a point in time because this is the maximum number of polygons encountered in the data set used. In the example case there is only a single polygon.

```
   STARTTIME: '010245'
     ENDTIME: '010900'
   POLYGON1: {'60NW PQI'  'YSJ'  'ACK'  'CYN'  'ETX'  'BUF'  'YOW'  '60NWPQI'}
ALTITUDEMAX1: {'220'}
ALTITUDEMIN1: {'020'}
   POLYGON2: [ ]
ALTITUDEMAX2: [ ]
ALTITUDEMIN2: [ ]
   POLYGON3: [ ]
ALTITUDEMAX3: [ ]
ALTITUDEMIN3: [ ]
   POLYGON4: [ ]
ALTITUDEMAX4: [ ]
ALTITUDEMIN4: [ ]
   POLYGON5: [ ]
ALTITUDEMAX5: [ ]
ALTITUDEMIN5: [ ]
```

Figure 4.4: Sample of Matlab Structure Array

The final step in the data collection and data parsing phase was to transform the polygon points into (longitude, latitude) points. Once this was accomplished the structure array was expanded to include these points, as shown in Figure 4.5.

```
STARTTIME: '010245'
      ENDTIME: '010900'
    POLYGON1: {'60NW PQI'  'YSJ'  'ACK'  'CYN'  'ETX'  'BUF'  'YOW'  '60NWPQI'}
 ALTITUDEMAX1: {'220'}
 ALTITUDEMIN1: {'020'}
    POLYGON2: [ ]
 ALTITUDEMAX2: [ ]
 ALTITUDEMIN2: [ ]
    POLYGON3: [ ]
 ALTITUDEMAX3: [ ]
 ALTITUDEMIN3: [ ]
    POLYGON4: [ ]
 ALTITUDEMAX4: [ ]
 ALTITUDEMIN4: [ ]
    POLYGON5: [ ]
 ALTITUDEMAX5: [ ]
 ALTITUDEMIN5: [ ]
      LAT1: [47.4761 45.4073 41.2818 39.8173 40.5810 42.9290 45.4417 47.4761]
     LONG1: [-69.1400 -65.8702 -70.0267 -74.4316 -75.6840 -78.6463 -75.8969 -69.1400]
      LAT2: [ ]
     LONG2: [ ]
      LAT3: [ ]
     LONG3: [ ]
      LAT4: [ ]
     LONG4: [ ]
      LAT5: [ ]
     LONG5: [ ]
```

Figure 4.5: Sample of Matlab Structure Array with Longitude and Latitude

4.2 Graphical Representation Results

An example of the graphical representation of the flight trajectories and AIRMETs was provided in chapter three. However, one of the objectives of the graphical representation phase was to locate the flights that did not intersect an icing AIRMET, and determine how many of these flights flew over the adverse icing conditions for at least a portion of their flight. The results for the three different flight paths are shown in Table 4.1.

Table 4.1: Flights Over Maximum Altitude of AIRMET

	Number of Flights Over Maximum Altitude of AIRMET	Total Number of Non-Intersecting AIRMETS	Proportion of Non-Intersecting Flights Flying Over AIRMET
BOS to CLE	78	555	0.141
BOS to IAD	214	831	0.258
CLE to IAD	161	705	0.228

4.3 Analysis of Reliability based on Icing AIRMETs

The procedure in chapter three was implemented to place the data into a format to obtain the dispatch reliability measures for small aircraft in icing conditions. In actuality, the dispatch unreliability was being determined due to concluding the number of "no-go" flights. The unreliability measures were found from the following equation, which divides the total number of times a flight path intersects an AIRMET by the total number of data points for the time period.

$$Unreliability\ Measure = \frac{Total\ Number\ of\ Intersecting\ AIRMETs\ for\ Time\ Period}{(Number\ of\ Days\ in\ Time\ Period \times 2 \times Number\ of\ Hour\ Periods)}$$

The number two in the denominator represents that there are two data points for each hour period due to collecting two years worth of AIRMET forecasts. To determine the unreliability measure for the semimonthly period of November 1st to November 14th for the morning hours (0600 to 1000) the variables would be as follows:

Number of Days in Time Period = 14
Number of Hour Periods = 4 (0600to0700, 0700to0800, 0800to0900, and 0900to1000)

The number of times the flight path of BOS to CLE intersected AIRMETs is shown in Table 4.2. Within the figure Nov 1a is referring to the first year worth of data, and Nov 1b is referring to the second year worth of data.

Table 4.2: Number of Intersecting AIRMETs for BOS to CLE at 12,000 feet for Semimonthly Period

	0600-0700	0700-0800	0800-0900	0900-1000
Nov 1a	5	5	5	6
Nov 2a	12	12	12	13
Hour Total	17	17	17	19
TOTAL	70			

Therefore, the unreliability measure would be:

$$Unreliability\ Measure = \frac{70}{14 * 2 * 4} = 0.583$$

Many scenarios needed to be incorporated within the analysis to capture all the combination of variables within this research. The variables consist of: three different flight paths, four different cruising altitudes, and three different time periods. Therefore, frequency plots of the dispatch unreliability of small aircraft due to icing conditions were created for the thirty-six scenarios. The thirty-six scenarios include the three flight paths, each for semimonthly, three-week, and monthly time periods, and each for a maximum cruising altitude of 12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet. Multiple worksheets, one for each graph, were created in Microsoft Excel to generate the subsequent graphs. Figure 4.6 is the worksheet used to determine the unreliability measure for the BOS to CLE flight for the monthly period at a maximum cruising altitude of 12,000 feet. Within the figure Month A is referring to the first year worth of data, and Month B is referring to the second year worth of data. The unreliability measures are highlighted in yellow. The remaining spreadsheets can be found in Appendix L.

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	10	10	10	13	10	10	10	10	10	13	11	11	11	11	150
November B	24	24	24	26	25	25	25	25	25	26	25	25	25	25	349
Hour Total	34	34	34	39	35	35	35	35	35	39	36	36	36	36	499
TOTAL	141				175					183					499
Fraction	0.5875				0.583333333					0.61					
December A	17	17	17	23	21	21	21	21	21	23	19	20	20	20	281
December B	22	22	22	25	24	24	24	24	24	25	22	22	22	22	324
Hour Total	39	39	39	48	45	45	45	45	45	48	41	42	42	42	605
TOTAL	165				225					215					605
Fraction	0.665322581				0.725806452					0.693548387					
January A	21	22	22	24	22	22	22	22	22	22	20	20	20	19	300
January B	24	24	24	25	24	24	24	24	24	26	23	23	23	23	335
Hour Total	45	46	46	49	46	46	46	46	46	48	43	43	43	42	635
TOTAL	186				230					219					635
Fraction	0.75				0.741935484					0.706451613					
February A	18	18	18	21	16	16	16	16	16	18	15	15	15	15	233
February B	19	19	19	21	16	16	16	16	16	20	18	18	18	17	249
Hour Total	37	37	37	42	32	32	32	32	32	38	33	33	33	32	482
TOTAL	153				160					169					482
Fraction	0.683035714				0.571428571					0.603571429					
March A	21	23	23	26	21	21	21	21	21	24	21	21	21	20	305
March B	16	16	16	18	18	18	18	18	18	23	21	21	21	21	263
Hour Total	37	39	39	44	39	39	39	39	39	47	42	42	42	41	568
TOTAL	159				195					214					568
Fraction	0.641129032				0.629032258					0.690322581					
April A	19	19	19	21	17	17	17	17	17	19	19	19	19	19	258
April B	21	21	21	21	18	18	18	18	19	22	19	19	19	19	273
Hour Total	40	40	40	42	35	35	35	35	36	41	38	38	38	38	531
TOTAL	162				176					193					531
Fraction	0.675				0.586666667					0.643333333					
May A	12	13	13	18	13	14	14	14	14	17	15	15	15	15	202
May B	18	18	18	18	16	16	16	16	16	19	15	15	15	15	231
Hour Total	30	31	31	36	29	30	30	30	30	36	30	30	30	30	433
TOTAL	128				149					156					433
Fraction	0.516129032				0.480645161					0.503225806					

Figure 4.6: BOS to CLE Unreliability Measures for Monthly Period – 12,000 feet

The unreliability measures determined for the monthly period for the flight path of BOS to CLE for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.7 to 4.10.

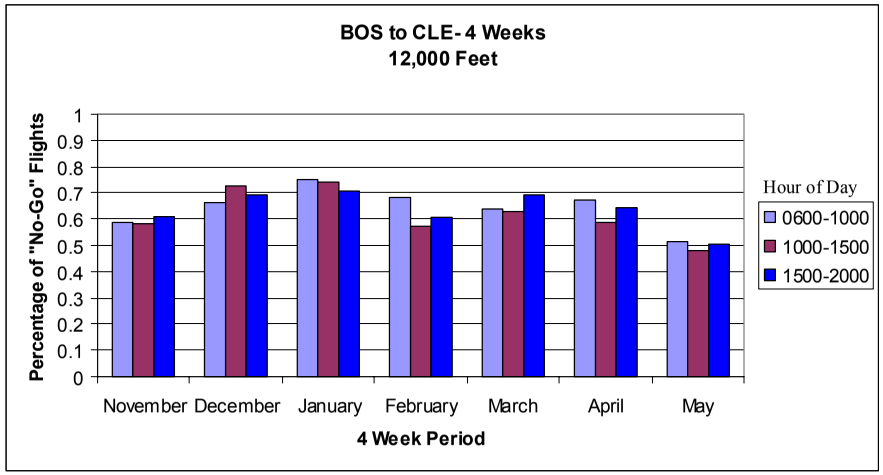

Figure 4.7: Unreliability Measures for BOS to CLE, Monthly Period – 12,000 Feet

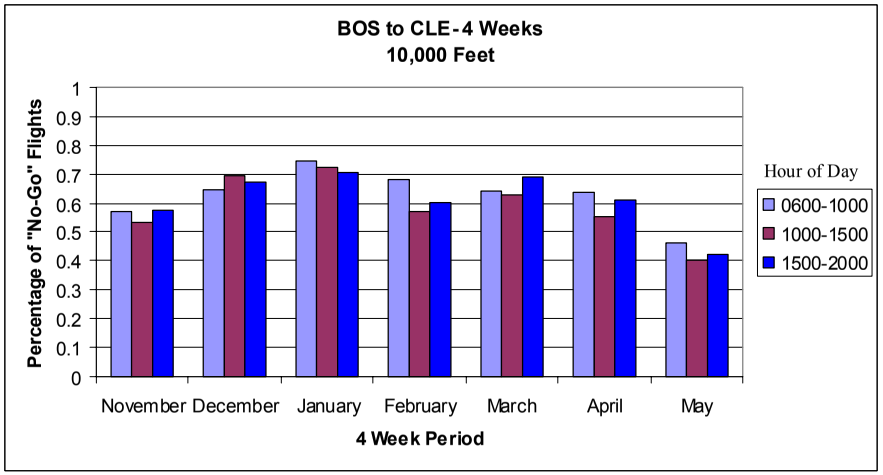

Figure 4.8: Unreliability Measures for BOS to CLE, Monthly Period–10,000 Feet

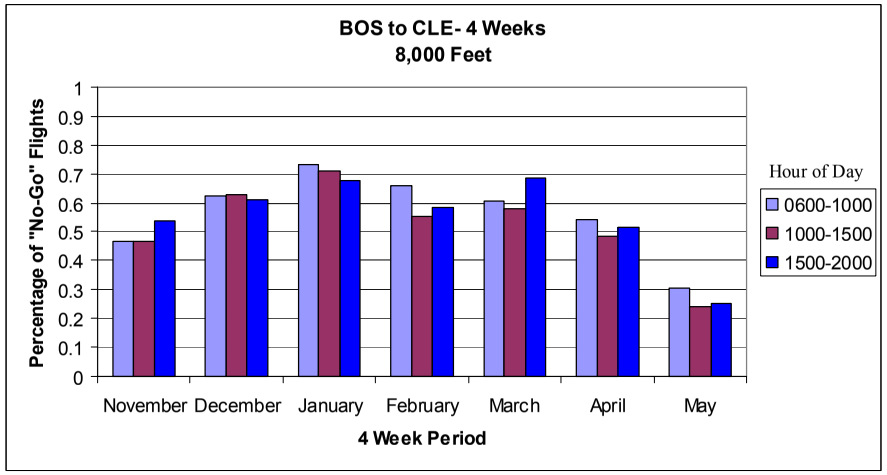

Figure 4.9: Unreliability Measures for BOS to CLE, Monthly Period – 8,000 Feet

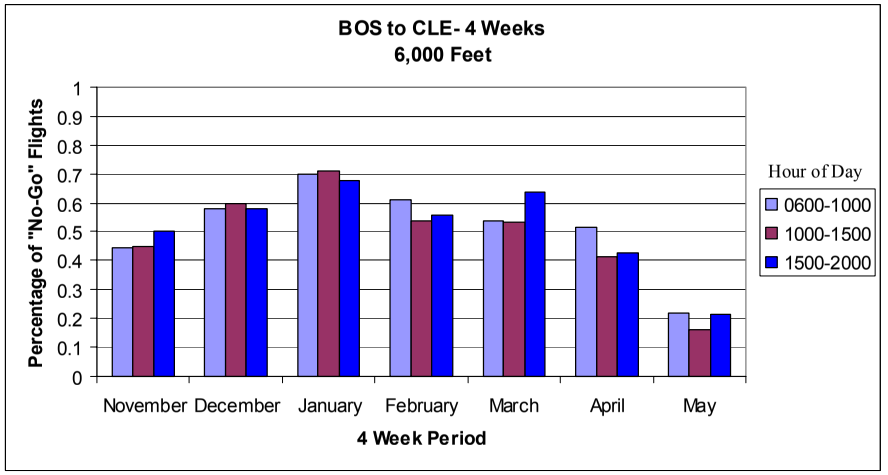

Figure 4.10: Unreliability Measures for BOS to CLE, Monthly Period – 6,000 Feet

The unreliability measures determined for the monthly period for the flight path of BOS to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.11 to 4.14.

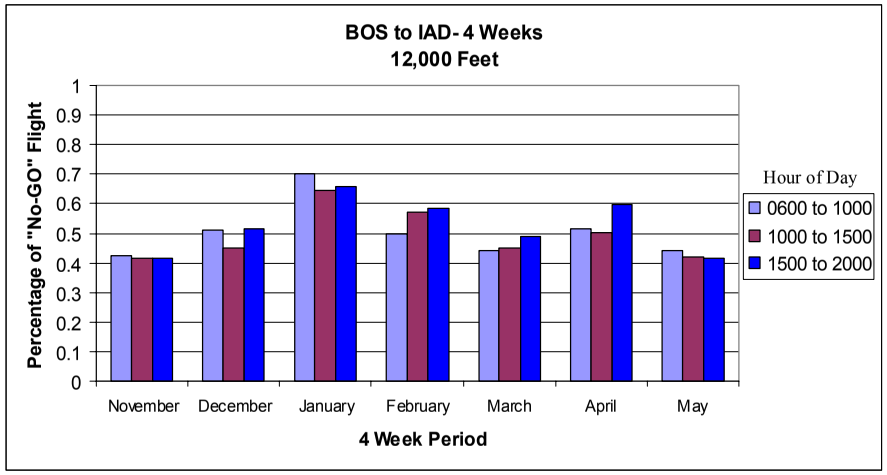

Figure 4.11: Unreliability Measures for BOS to IAD, Monthly Period – 12,000 Feet

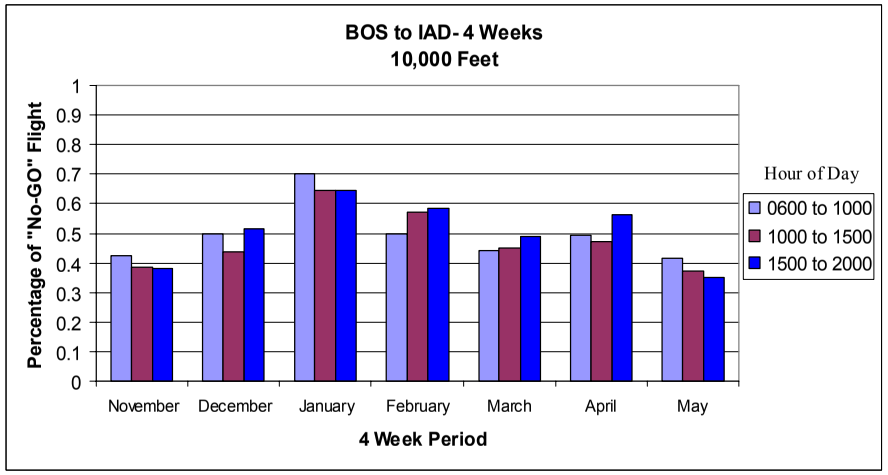

Figure 4.12: Unreliability Measures for BOS to IAD, Monthly Period – 10,000 Feet

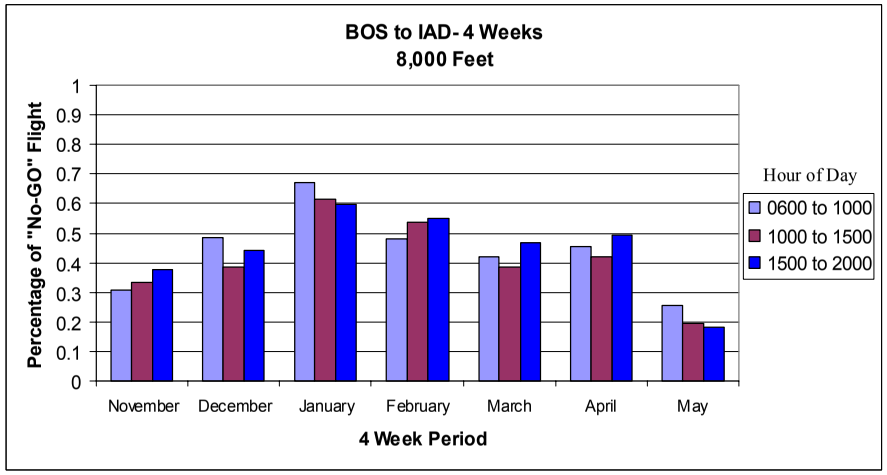

Figure 4.13: Unreliability Measures for BOS to IAD, Monthly Period – 8,000 Feet

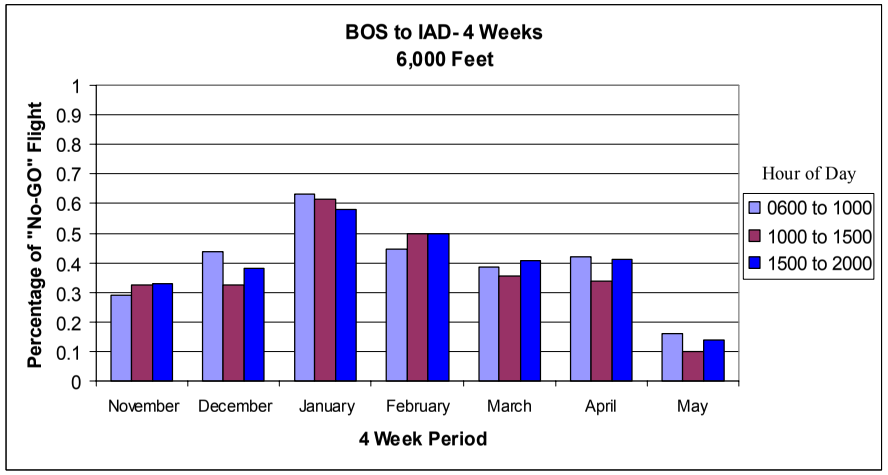

Figure 4.14: Unreliability Measures for BOS to IAD, Monthly Period – 6,000 Feet

The unreliability measures determined for the monthly period for the flight path of CLE to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.15 to 4.18.

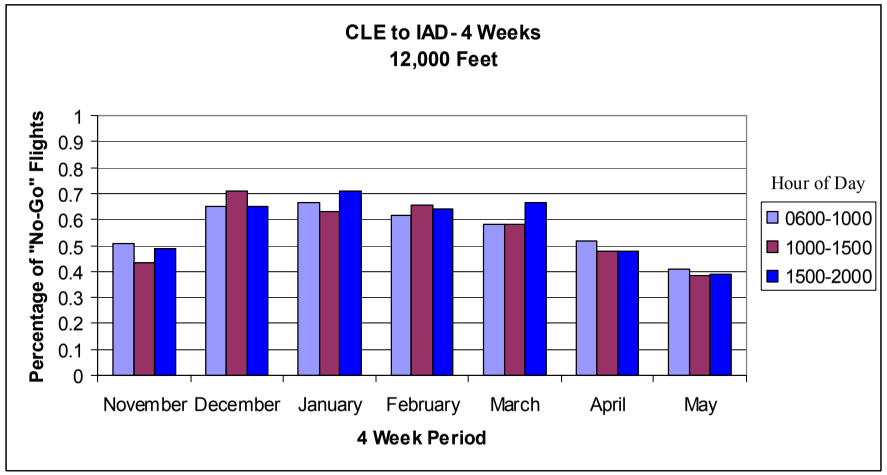

Figure 4.15: Unreliability Measures for CLE to IAD, Monthly Period – 12,000 Feet

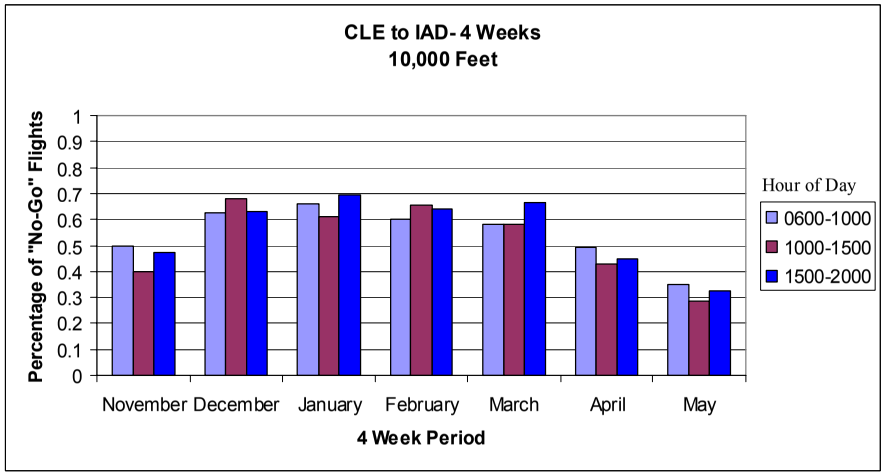

Figure 4.16: Unreliability Measures for CLE to IAD, Monthly Period – 10,000 Feet

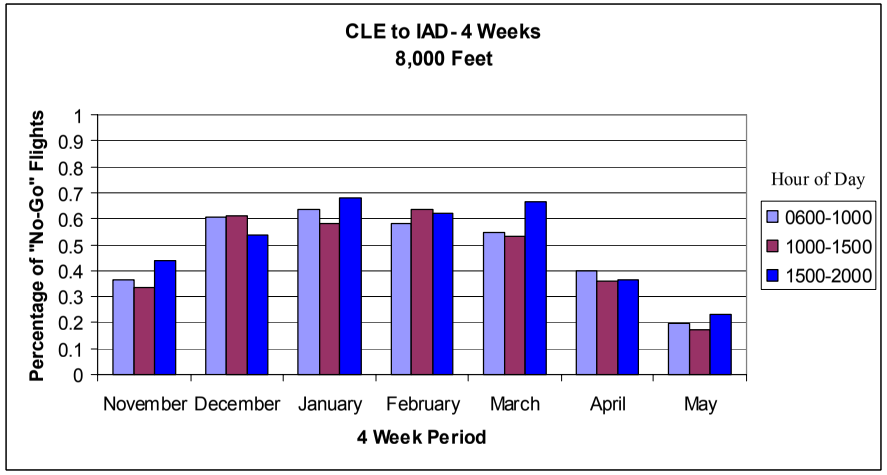

Figure 4.17: Unreliability Measures for CLE to IAD, Monthly Period – 8,000 Feet

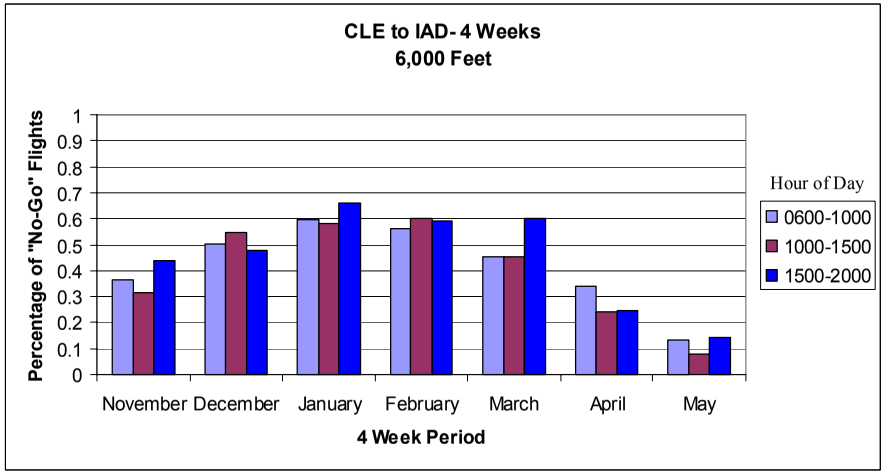

Figure 4.18: Unreliability Measures for CLE to IAD, Monthly Period – 6,000 Feet

The unreliability measures determined for the three-week period for the flight path of BOS to CLE for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.19 to 4.22.

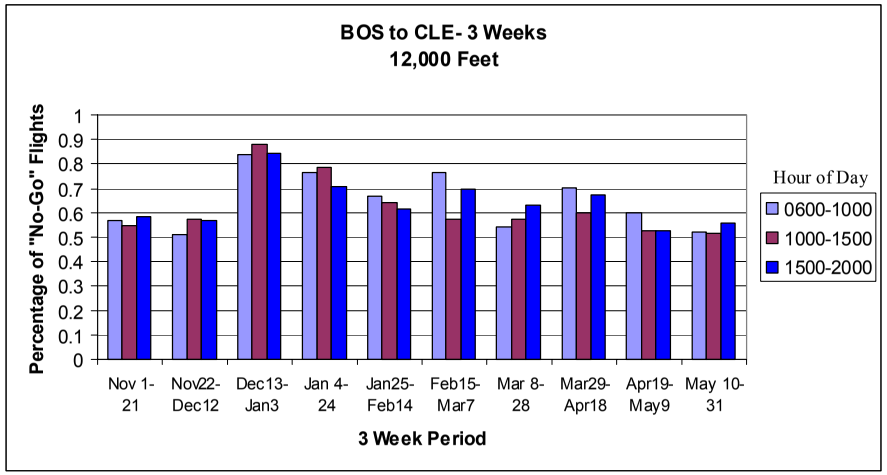

Figure 4.19: Unreliability Measures for BOS to CLE, 3-Week Period – 12,000 Feet

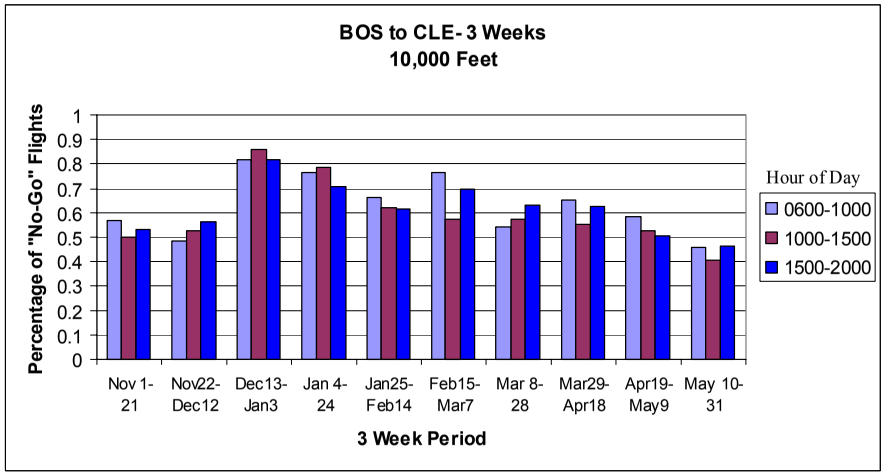

Figure 4.20: Unreliability Measures for BOS to CLE, 3-Week Period – 10,000 Feet

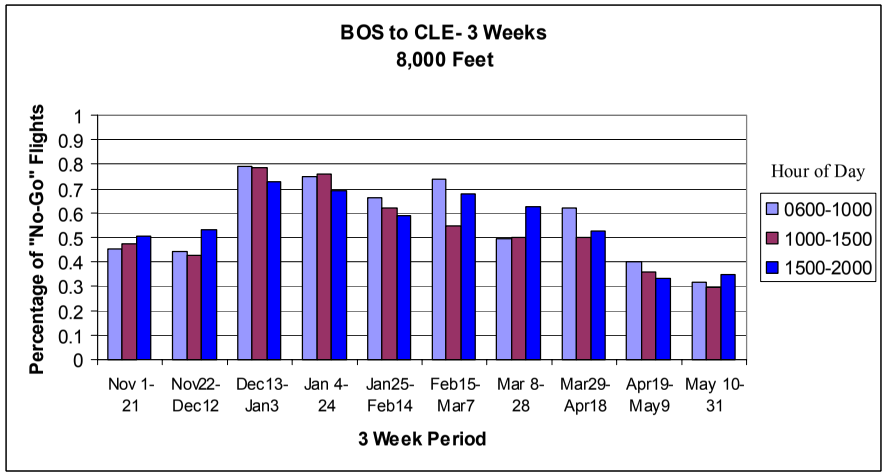

Figure 4.21: Unreliability Measures for BOS to CLE, 3-Week Period – 8,000 Feet

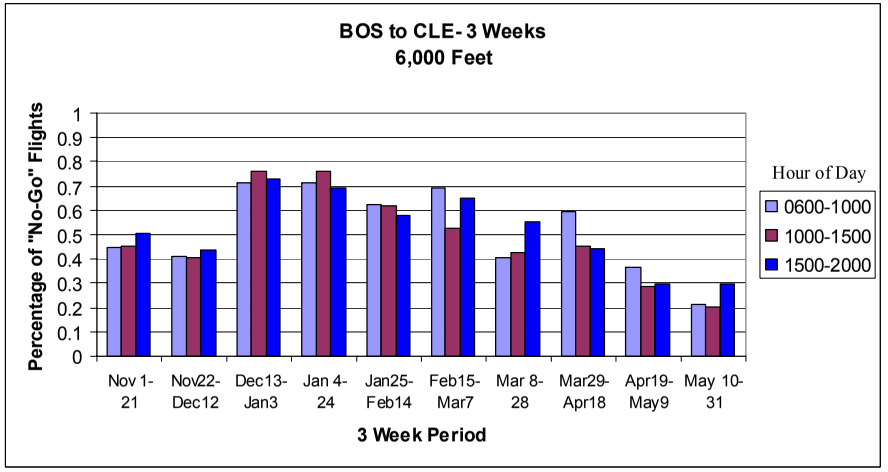

Figure 4.22: Unreliability Measures for BOS to CLE, 3-Week Period – 6,000 Feet

The unreliability measures determined for the three-week period for the flight path of BOS to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.23 to 4.26.

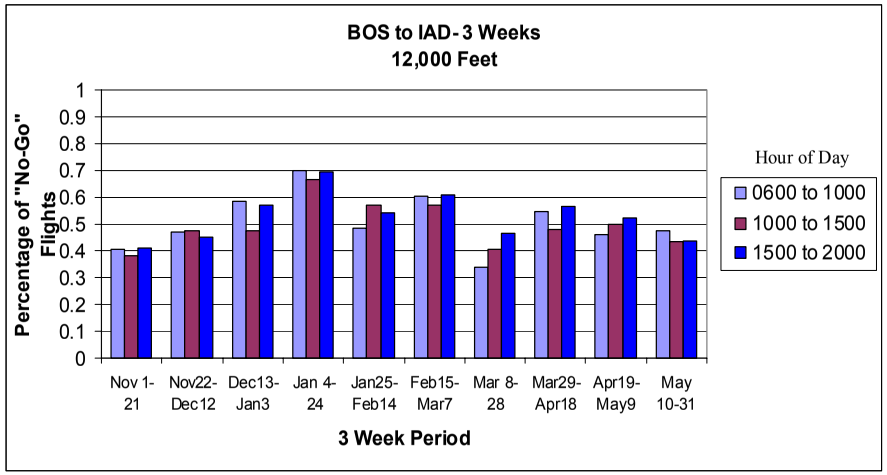

Figure 4.23: Unreliability Measures for BOS to IAD, 3-Week Period – 12,000 Feet

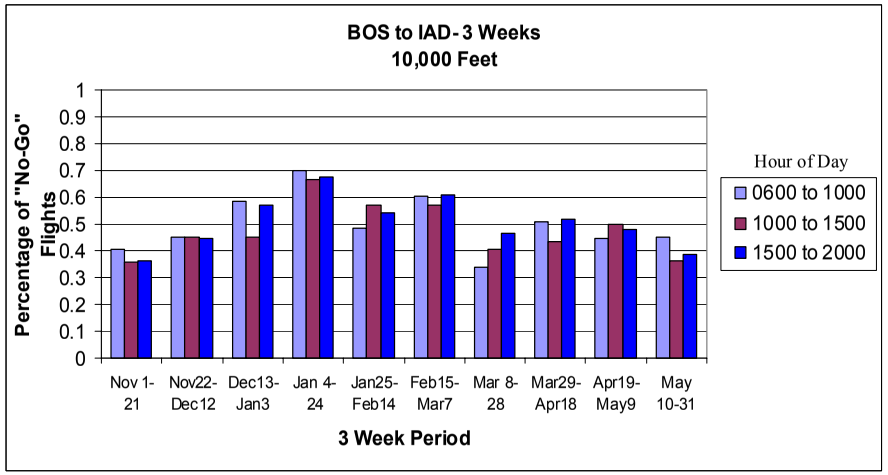

Figure 4.24: Unreliability Measures for BOS to IAD, 3-Week Period – 10,000 Feet

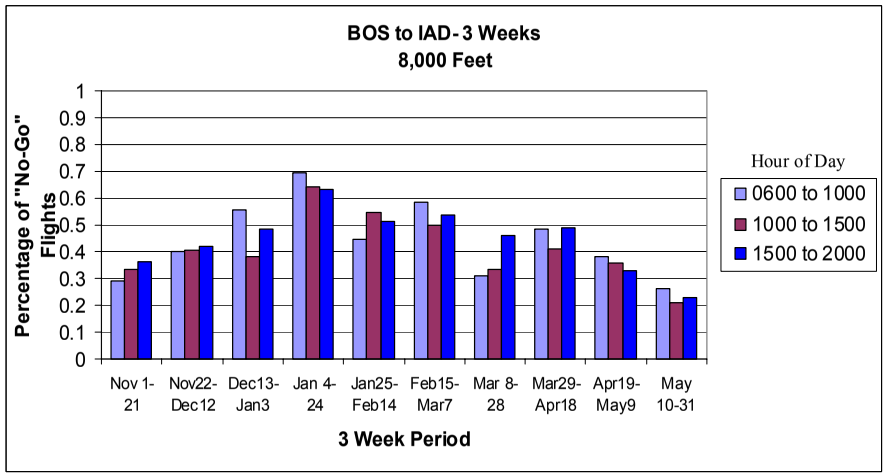

Figure 4.25: Unreliability Measures for BOS to IAD, 3-Week Period – 8,000 Feet

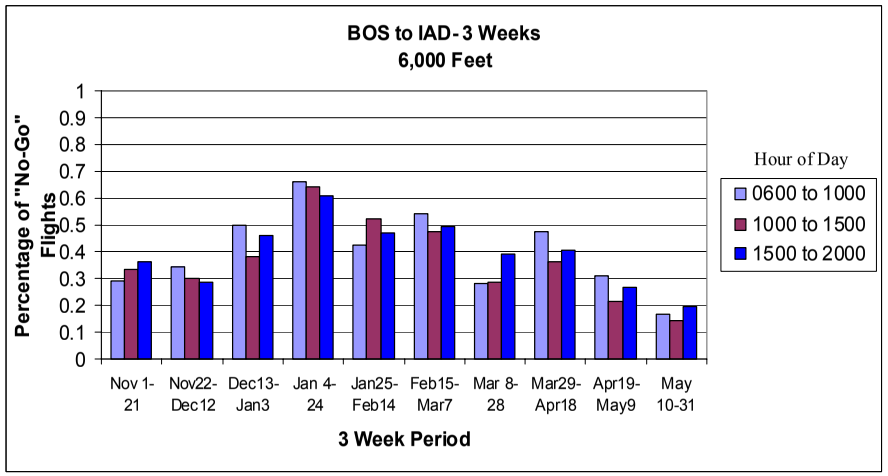

Figure 4.26: Unreliability Measures for BOS to IAD, 3-Week Period – 6,000 Feet

The unreliability measures determined for the three-week period for the flight path of CLE to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.27 to 4.30.

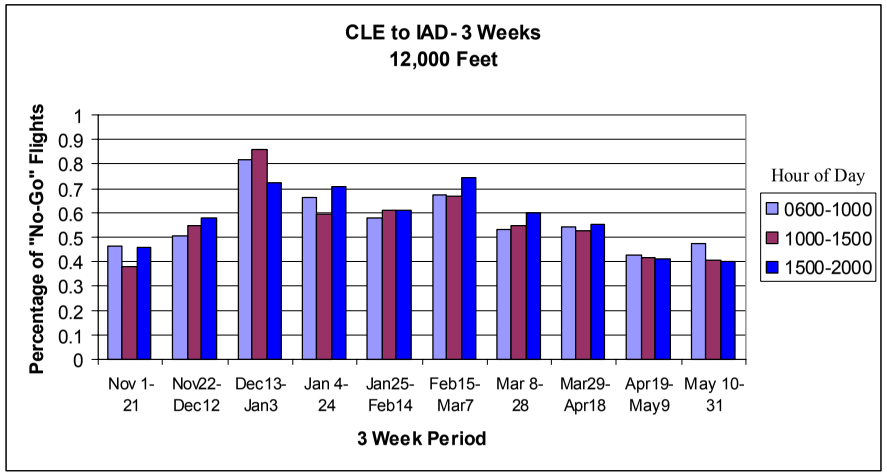

Figure 4.27: Unreliability Measures for CLE to IAD, 3-Week Period – 12,000 Feet

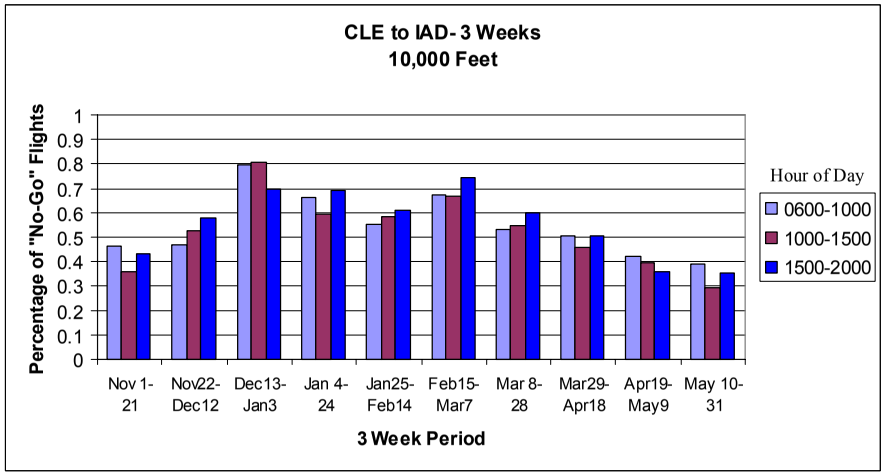

Figure 4.28: Unreliability Measures for CLE to IAD, 3-Week Period – 10,000 Feet

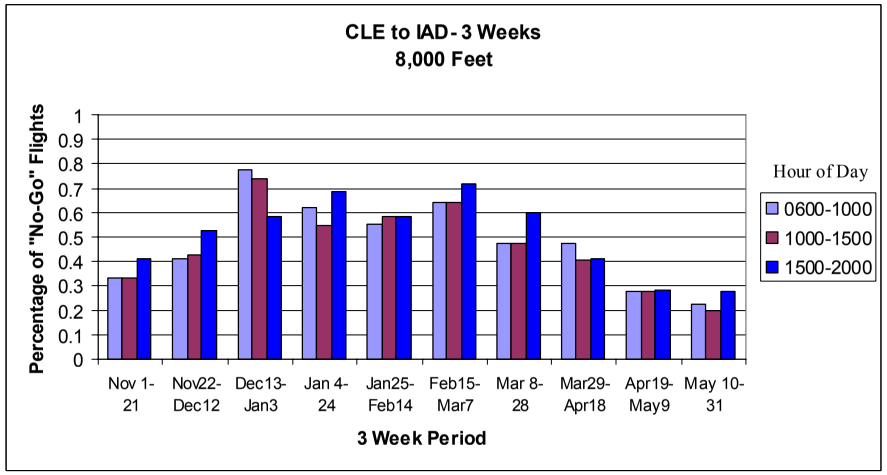

Figure 4.29: Unreliability Measures for CLE to IAD, 3-Week Period – 8,000 Feet

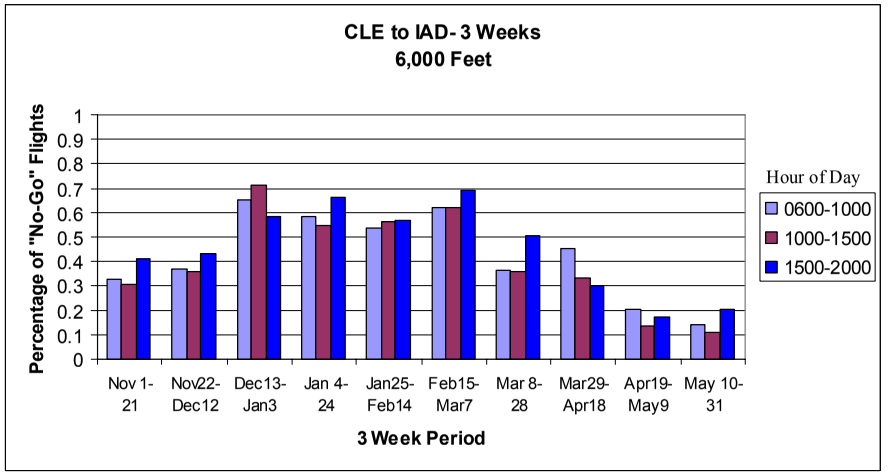

Figure 4.30: Unreliability Measures for CLE to IAD, 3-Week Period – 6,000 Feet

The unreliability measures determined for the semimonthly period for the flight path of BOS to CLE for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.31 to 4.34.

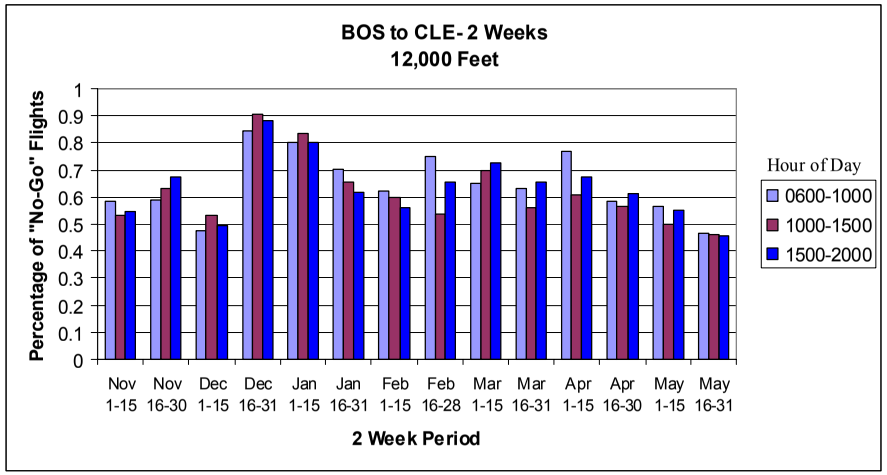

Figure 4.31: Unreliability Measures for BOS to CLE, 2-Week Period – 12,000 Feet

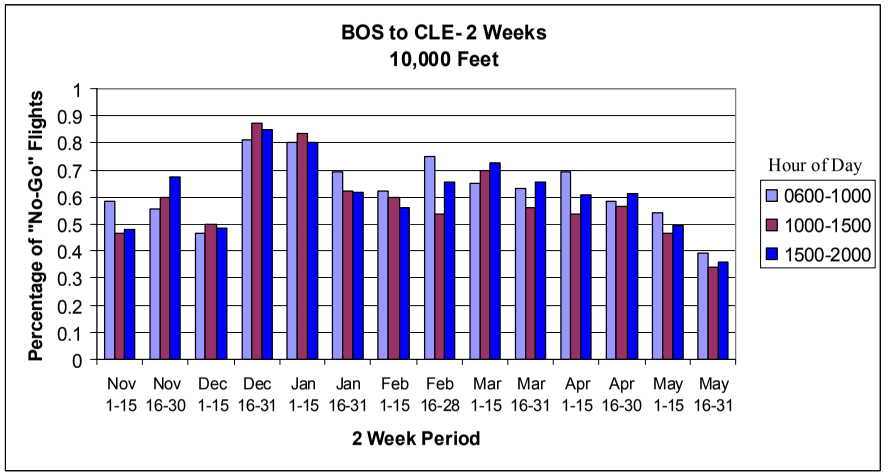

Figure 4.32: Unreliability Measures for BOS to CLE, 2-Week Period – 10,000 Feet

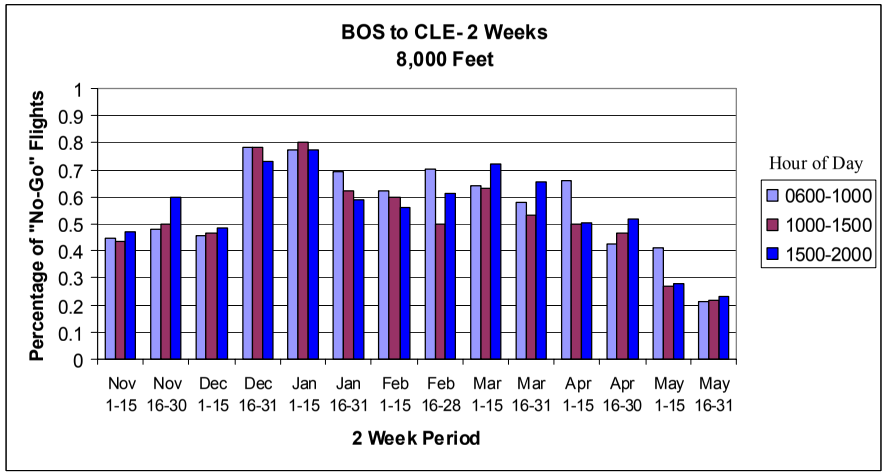

Figure 4.33: Unreliability Measures for BOS to CLE, 2-Week Period – 8,000 Feet

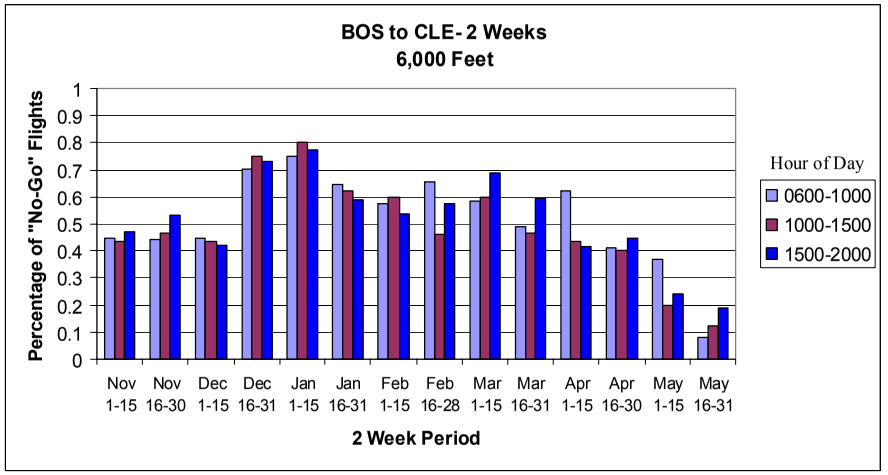

Figure 4.34: Unreliability Measures for BOS to CLE, 2-Week Period – 6,000 Feet

The unreliability measures determined for the semimonthly period for the flight path of BOS to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.35 to 4.38.

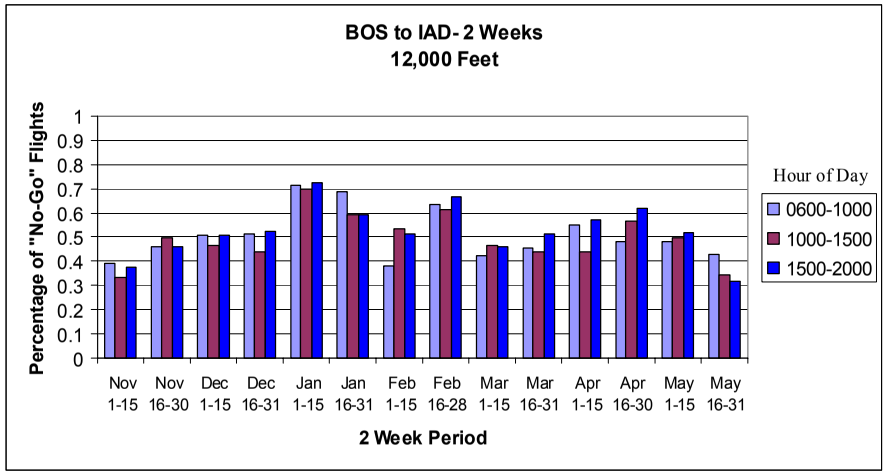

Figure 4.35: Unreliability Measures for BOS to IAD, 2-Week Period – 12,000 Feet

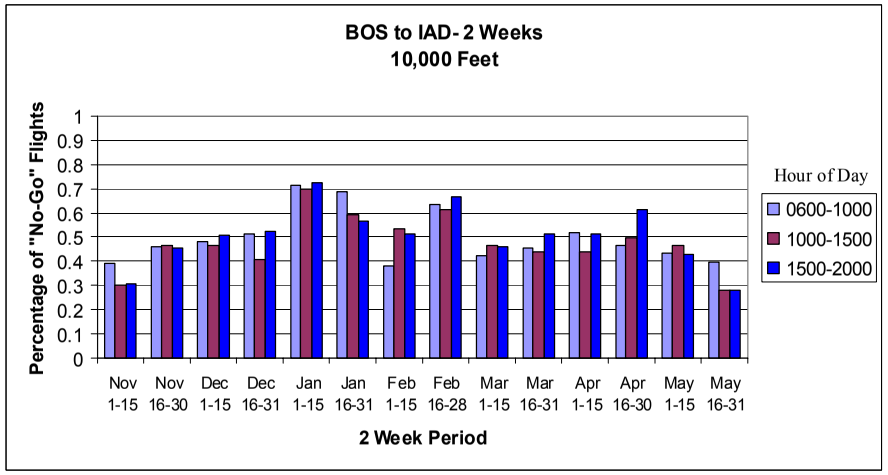

Figure 4.36: Unreliability Measures for BOS to IAD, 2-Week Period – 10,000 Feet

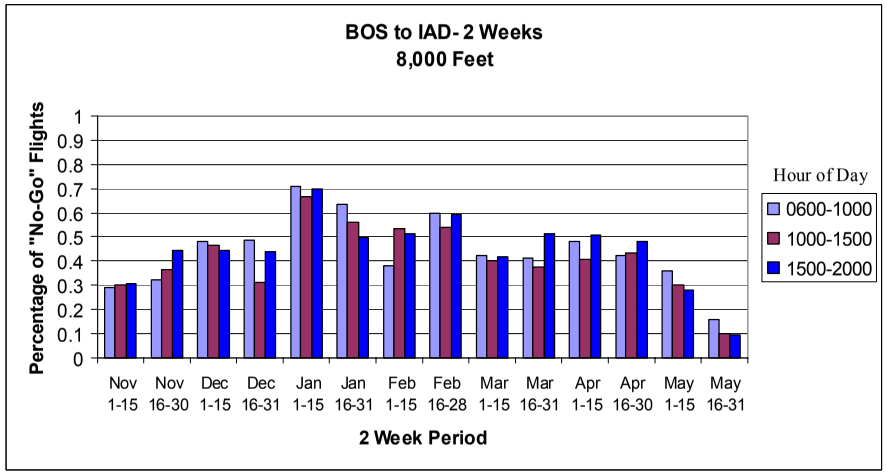

Figure 4.37: Unreliability Measures for BOS to IAD, 2-Week Period – 8,000 Feet

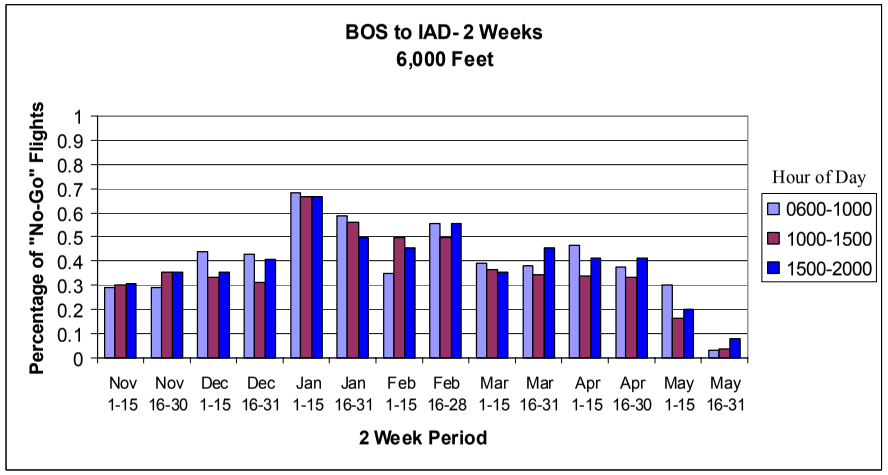

Figure 4.38: Unreliability Measures for BOS to IAD, 2-Week Period – 6,000 Feet

The unreliability measures determined for the semimonthly period for the flight path of CLE to IAD for the various maximum cruising altitudes (12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet) were converted into graphical form. The graphs are shown in Figures 4.39 to 4.42.

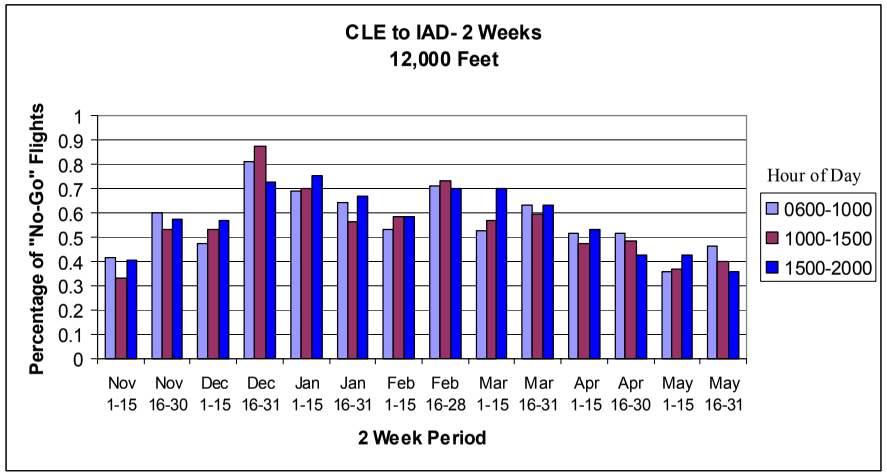

Figure 4.39: Unreliability Measures for CLE to IAD, 2-Week Period – 12,000 Feet

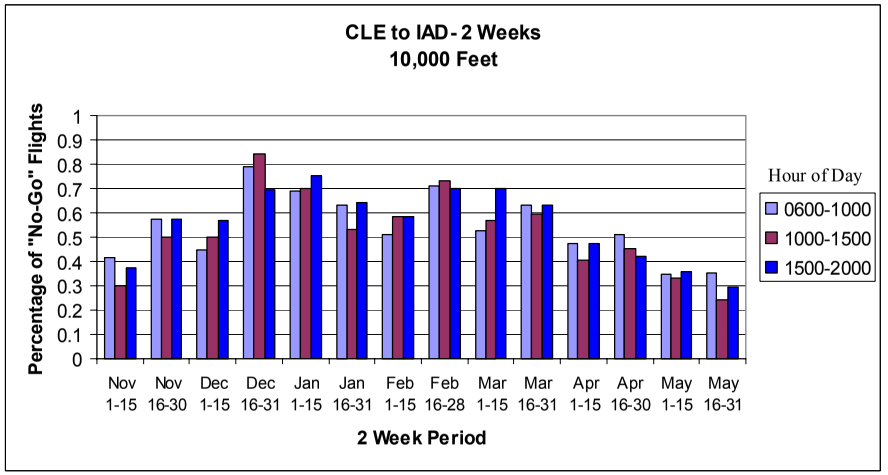

Figure 4.40: Unreliability Measures for CLE to IAD, 2-Week Period – 10,000 Feet

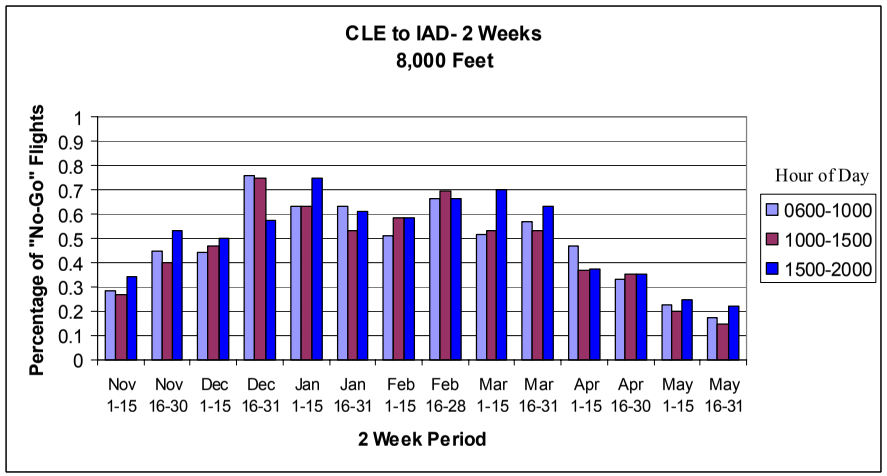

Figure 4.41: Unreliability Measures for CLE to IAD, 2-Week Period – 8,000 Feet

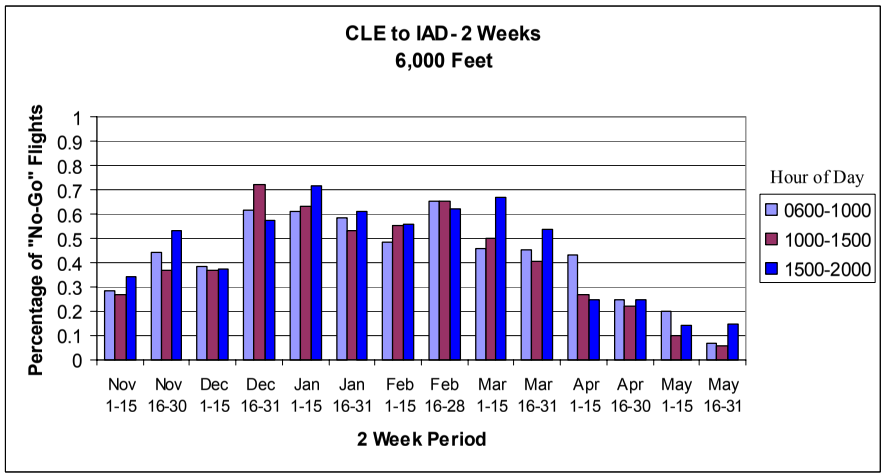

Figure 4.42: Unreliability Measures for CLE to IAD, 2-Week Period – 6,000 Feet

To obtain an overall probability for the dispatch unreliability of small aircraft due to the weather condition of icing an average of the three flight paths at each maximum cruising altitude was determined. The overall probability of a "no-go" for all three flight paths at the normal cruising altitude of 12,000 feet is 56.8%. When the cruising altitude is reduced to 10,000 feet, 8,000 feet, and 6,000 feet the probability of a "no-go" for all three flight paths reduces to 54.6%, 48.5%, and 43.7% respectively. The semimonthly, three-week, and monthly time periods all correspond to these same values since the sum of all the intersecting AIRMETs is equal.

4.4 Verification of Results

The results were verified to ensure the reliability of the programming used within this research. A five percent sample of the total number of AIRMETs, 164 out of 3,280, was selected to verify the feasibility of the programming that was incorporated within this thesis. Random numbers were generated through the use of Microsoft Excel to select the AIRMETs to be verified.

The verification procedure was completed by hand, and it utilized the same method that was incorporated within the Matlab programs. Initially, the AIRMETs and the flight trajectories were checked to determine if there were intersections based on longitude and latitude. If there were intersections, these points were recorded. Next, the points in the flight trajectory that intersected the AIRMET in regards to longitude and latitude were examined to locate points that also intersected the altitude of the AIRMET. The flight paths that had points intersecting based on all three dimensions were recorded. The final step in the verification process was to ensure that the intersecting flight paths determined by hand correlated to the flight paths output by the programs. The results confirmed that for each of the 164 AIRMETs verified by hand, the results were equivalent to the results from the programs. The results of the verification are shown in Table 4.3.

Verification was also completed through the determination of the overall probability of a "no-go". It was known that the overall probability should be independent of the semimonthly, three-week, and monthly time period. The total number of intersecting AIRMETs should be equivalent for all of the time periods when the semimonthly, three-week, and monthly period are summed. To ensure that each intersecting AIRMET had been included within each time period, the overall "no-go" probabilities were compared. The verification provided confirmation that these values were the same independent of the time period.

Table 4.3: Verification Results

No.	Lat/Long	Altitude	Cities	Match
525	yes	yes	BOS/CLE CLE/IAD	YES
1053	no	---	---	YES
2833	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1126	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
351	yes	yes	BOS/CLE CLE/IAD	YES
1565	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2234	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1711	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
3275	no	---	---	YES
2831	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2568	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
468	yes	yes	BOS/IAD	YES
2305	yes	yes	CLE/IAD	YES
2509	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
298	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2789	yes	yes	BOS/CLE CLE/IAD	YES
886	no	---	---	YES
404	no	---	---	YES

No.	Lat/Long	Altitude	Cities	Match
1950	no	---	---	YES
2965	yes	yes	BOS/CLE CLE/IAD	YES
960	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
520	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2493	yes	yes	BOS/CLE BOS/IAD	YES
1450	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1925	no	---	---	YES
1608	yes	yes	BOS/IAD CLE/IAD	YES
1247	yes	yes	BOS/CLE CLE/IAD	YES
1578	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2247	yes	yes	BOS/CLE CLE/IAD	YES
396	no	---	---	YES
1504	yes	no	---	YES
297	yes	yes	BOS/IAD CLE/IAD	YES
665	no	---	---	YES
1870	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1837	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
142	no	---	---	YES

No.	Lat/Long	Altitude	Cities	Match
1865	yes	yes	BOS/CLE CLE/IAD	YES
2298	yes	yes	BOS/CLE	YES
3076	no	---	---	YES
2969	yes	yes	BOS/CLE CLE/IAD	YES
2528	yes	yes	BOS/CLE CLE/IAD	YES
3073	no	---	---	YES
145	no	---	---	YES
1479	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
487	yes	yes	BOS/IAD CLE/IAD	YES
1331	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2181	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
3237	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1227	no	---	---	YES
1900	no	---	---	YES
2557	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2155	yes	yes	BOS/CLE CLE/IAD	YES
194	no	---	---	YES
1918	yes	yes	BOS/CLE	YES

No.	Lat/Long	Altitude	Cities	Match
2874	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2440	yes	yes	BOS/CLE CLE/IAD	YES
1779	yes	no	---	YES
1051	no	---	---	YES
2443	yes	yes	BOS/CLE	YES
986	yes	yes	BOS/CLE BOS/IAD	YES
883	no	---	---	YES
1860	yes	yes	BOS/CLE BOS/IAD	YES
550	yes	yes	BOS/CLE CLE/IAD	YES
309	yes	yes	BOS/CLE BOS/IAD	YES
1758	no	---	---	YES
37	no	---	---	YES
1821	no	---	---	YES
2339	no	---	---	YES
2239	yes	yes	BOS/CLE CLE/IAD	YES
2147	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1635	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2870	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES

No.	Lat/Long	Altitude	Cities	Match
2846	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
778	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1816	no	---	---	YES
1106	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1260	no	---	---	YES
1585	no	---	---	YES
3279	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
981	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
816	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1919	yes	yes	BOS/CLE CLE/IAD	YES
2410	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1227	no	---	---	YES
2930	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2913	yes	yes	BOS/CLE CLE/IAD	YES
1511	yes	no	---	YES
516	yes	yes	BOS/CLE BOS/IAD	YES
629	yes	yes	BOS/CLE CLE/IAD	YES

No.	Lat/Long	Altitude	Cities	Match
2447	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
735	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
959	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
805	yes	yes	BOS/CLE BOS/IAD	YES
3015	yes	yes	BOS/CLE BOS/IAD	YES
1292	no	---	---	YES
3113	yes	yes	BOS/CLE BOS/IAD	YES
3102	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2213	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
373	yes	yes	BOS/CLE	YES
1312	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
342	yes	yes	BOS/CLE CLE/IAD	YES
1222	no	---	---	YES
1628	no	---	---	YES
974	yes	no	---	YES
719	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
296	yes	yes	BOS/IAD CLE/IAD	YES

No.	Lat/Long	Altitude	Cities	Match
691	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
434	yes	yes	BOS/CLE BOS/IAD	YES
2427	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1371	no	---	---	YES
598	yes	yes	BOS/CLE CLE/IAD	YES
746	yes	yes	BOS/CLE BOS/IAD	YES
1571	yes	yes	CLE/IAD	YES
2006	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1588	no	---	---	YES
88	no	---	---	YES
3037	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1064	yes	yes	BOS/CLE CLE/IAD	YES
671	yes	yes	BOS/CLE	YES
2882	yes	yes	BOS/CLE CLE/IAD	YES
745	yes	no	---	YES
1122	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
3108	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
819	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1914	yes	yes	BOS/IAD CLE/IAD	YES
2550	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1433	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1582	yes	no	---	YES
1962	no	---	---	YES

No.	Lat/Long	Altitude	Cities	Match
2078	no	---	---	YES
642	no	---	---	YES
1278	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1517	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
272	no	---	---	YES
2788	yes	yes	BOS/CLE CLE/IAD	YES
1795	yes	yes	BOS/CLE CLE/IAD	YES
2521	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
2202	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
238	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
264	no	---	---	YES
2840	no	---	---	YES
1267	yes	yes	BOS/CLE CLE/IAD	YES
1877	yes	yes	BOS/CLE CLE/IAD	YES
3115	yes	yes	BOS/CLE BOS/IAD	YES
2404	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1944	no	---	---	YES
1852	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
167	yes	yes	BOS/CLE	YES
1497	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
463	yes	yes	BOS/IAD CLE/IAD	YES
15	yes	yes	BOS/CLE BOS/IAD	YES
2307	yes	yes	BOS/CLE CLE/IAD	YES

No.	Lat/Long	Altitude	Cities	Match
1768	no	---	---	YES
2516	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1155	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
3041	yes	yes	BOS/CLE BOS/IAD	YES
3208	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
1089	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES

No.	Lat/Long	Altitude	Cities	Match
2512	yes	yes	BOS/CLE CLE/IAD	YES
1489	no	---	---	YES
1856	yes	yes	BOS/CLE BOS/IAD	YES
1490	no	---	---	YES
2518	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES
3209	yes	yes	BOS/CLE BOS/IAD CLE/IAD	YES

Chapter 5- Conclusion

5.1 Summary

There are many factors that need to be assessed before the feasibility of a SATS system can be determined. This thesis touches on one of these factors, the dispatch reliability of small aircraft with respect to one weather condition: icing. To determine the dispatch reliability of small aircraft with respect to icing two years of data were utilized for a reduced set of origin and destination pairs. These data consisted of AIRMETs for the months of November to May because it was determined that the probability of icing would be low in the remaining months. The principal challenge throughout this research was determining the "no-go" conditions from the text reports acquired from the NOAA. The raw data provided by the NOAA consisted of much information that was not relevant to the scope of this project. The initial task was to parse out the required information, which consisted of the start time and end time of the icing AIRMETs, the polygon that outlined the shape of the icing AIRMETs, and the minimum and maximum altitude of the icing AIRMETs.

After the relevant data had been stored, the flight trajectory and AIRMETs were simultaneously plotted to determine intersection points. The graph of all of the AIRMETs rotated over a stationary graph of the three flight paths. To determine intersections between the AIRMET and a flight path the analysis was broken into two components. First, intersections were determined based on the two-dimensions of longitude and latitude. Then, the AIRMETs intersecting the flight path in regards to longitude and latitude were analyzed for intersections based on altitude.

The results were allocated to specific time periods to allow them to be more relevant to small aircraft flights. It was assumed that a vast majority of small aircraft flights occur between the hours of 0600 to 2000; therefore these were the times considered in this

research. This time period was divided into three subsections to indicate morning (0600-1000), afternoon (1000-1500), and evening (1500-2000). The dispatch reliability of each of these subsections was analyzed over a semimonthly, three-week, and monthly period.

Aircraft with un-pressurized cabins can fly at a maximum cruising altitude of 12,000 feet, and while this altitude cannot be increased without special provisions it can be decreased. If a small aircraft pilot was aware of icing conditions, but could safely fly below these conditions to their predetermined location this would be acceptable. However, the aircraft must account for the terrain along the flight path and sustain adequate clearance. To account for this, flight paths that intersected icing conditions were reevaluated in a stepwise fashion by decreasing the altitude to 10,000 feet, 8,000 feet, and 6,000 feet until a non-intersection was determined.

The results indicate that the probability of a "no-go" is quite substantial; however by decreasing the cruising altitude the probability of a "no-go" is correspondingly reduced. The overall probability of a "no-go" for all three flight paths at the cruising altitudes of 12,000 feet, 10,000 feet, 8,000 feet, and 6,000 feet are 56.8%, 54.6%, 48.5%, and 43.7% respectively. This research was conducted based on criteria that correlated to high probabilities of icing conditions. Foremost, the geographic location of the analysis was the northeastern portion of the United States, and this area is very prone to in-flight icing conditions during the winter months. Next, if icing was forecast along the predetermined flight path, the aircraft was considered a "no-go" independent of the severity of the icing.

5.2 Future Research

There were restrictions within the scope of this research mostly attributed to time and monetary constraints. The credibility of the results could be increased through future research. This model could be extended to incorporate additional weather conditions, a more extensive data set, additional geographic locations, a precise minimum altitude, and an increase in scope to incorporate alternatives within the flight path and time of departure.

This thesis focused on icing. However, there are additional weather conditions that affect the ability of a small aircraft to safely make a predetermined flight. To determine the overall weather related dispatch reliability of small aircraft each of these need to be assessed and included within the weather related dispatch reliability measure.

The credibility of the results of this research could be improved through the collection of a larger data set. The data set used consisted of two years worth of data for the months November to May. Increasing the sample size would provide more reliable results, and eliminate biases based on extremes present in any given year. The entire year of data could also be included, and this will be a necessity for some of the additional weather conditions. Though icing is more prevalent in the winter months other weather conditions are independent of the time of year. The data could also be expanded to include SIGMETs, as well as the AIRMETs utilized in this research. SIGMETs correspond to more severe weather conditions, but if an AIRMET is not issued correlating to the severe weather the SIGMET could indicate an additional "no-go" condition.

This research could be enhanced through the incorporation of a larger geographic area. The geographic area for this thesis was derived due to the tendencies for icing to exist in the northeastern portion of the United States, but this could be extended for a more reliable representation of the entire United States.

This research could be further advanced by eliminating the use of the lowest minimum altitude point as the overall minimum altitude. The minimum altitude is often a rough line that is connecting various points, but for this thesis the minimum point was determined and used as the minimum altitude across the span of the polygon. If each individual point were incorporated the dispatch reliability would be more true to the conditions.

This model can be improved by incorporating deviations around the dangerous weather. An aircraft may have alternatives if adverse weather exists along the predetermined flight path. A diversion may be implemented that allows the flight to redirect its path around the weather condition. This diversion will normally need to be within certain limitations in regards to the overall mileage or flight time.

The overall scope of the research could be extended to determine the improvement in the dispatch reliability of small aircraft when the departure time becomes flexible. Passengers may be willing to delay their departure for a certain time period before finding an alternative transportation method. This delay will need to be within the passenger's tolerance.

To allow for the results to be more realistic this research could be expanded to incorporate the severity of the weather condition in the "go" or "no-go" determination. If the weather condition does not pose an imminent threat to the aircraft the pilot may choose to fly through the conditions.

This research has provided a valid foundation for future research in regards to the dispatch reliability of small aircraft due to icing conditions. Additional research could be completed to improve the creditability of the results, as well as lead to a more complete understanding of the dispatch reliability of small aircraft due to all types of weather phenomena over the entire United States. This study will enable future researchers to extend this concept into a complete representation of the dispatch reliability due to all weather conditions and geographic locations.

References

[1] Baik, Hojong. (2003). *flightPathGeneration_SATS.m* (Version 6.5.1) [Matlab Program]. Blacksburg, VA.

[2] Bernstein, Ben C., Omeron, Tiffany A., Politovich, Marcia K., & McDonough Frank. (1998). Surface weather features associated with freezing precipitation and severe in-flight aircraft icing. *Atmospheric Research*, 46, 57-73.

[3] Boeing. (2003). Current Market Outlook 2003. Retrieved April 27, 2004, from http://www.boeing.com/commercial/cmo/pdf/CMO2003.pdf

[4] Chavan, Harish. (2003). *A Heuristic Approach to Solve Air Taxi Scheduling Problem.* Thesis. Virginia Polytechnic Institute and State University.

[5] Civil Aviation Authority. (2000). *Aircraft Icing Handbook*. Retrieved March 11, 2004 from http://www.caa.govt.nz/fulltext/safety_booklets/ aircraft_icing_handbook.pdf

[6] Ding, Y., Rong, J., & Valasek, J. (2003). *Automation Capabilities Analysis Methodology for Non-Controlled Airports.* Paper presented at the AIAA Modeling and Simulation Technologies Conference and Exhibit, Austin, TX.

[7] Holmes, Dr. Bruce J. *Small Aircraft Transportation System. A Vision for 21st Century Transportation Alternatives.* Retrieved on March 15, 2004, from http://www.smithairfield.com/Documents/forcasts/Small%20Aircraft%20 Transportation%20System.doc

[8] Kempthorne, Oscar, & Folks, Leroy. (1971). *Probability, Statistics, and Data Analysis.* Ames: The Iowa State University Press.

[9] Long, Dou, Lee, David, Johnson, Jesse, & Kostiuk, Peter. (2001). *A Small Aircraft Transportation System (SATS) Demand Model.* Retrieved on April 11, 2004, from http://techreports.larc.nasa.gov/ltrs/PDF/2001/cr/NASA-2001-cr210874.pdf

[10] McGrath, Robert N. (2002). A study of NASA's vision for the future of air travel. *Technological Forecasting and Social Change*, 69, 173-193.

[11] McGrath, Robert N., & Young, Seth B. (2002). NASA's small aircraft costs versus automobile costs and the economic value of traveler time. *Technovation,* 22, 325-336.

[12] NASA & NCAM. (2002). *Small Aircraft Transportation System (SATS) Joint Sponsored Research and Development Agreement.* Retrieved on November 20, 2003 from, http://ncam-sats.org/JSRDA.doc.

[13] Ross, Sheldon M. (2000). *Introduction to Probability Models* (7th Ed.). San Diego: Harcourt/Academic Press, 2000.

[14] Spirkovska, Lilly, & Lodha, Suresh K. (2002). AWE: aviation weather data visualization environment. *Computers & Graphics*, 26, 169-191.

[15] Taha, Hamdy A. (1997). *Operations Research An Introduction* (6^{th} Ed.). Upper Saddle River: Prentice Hall, 1997.

[16] Teodorovic D. B. (1988). *Airline Operations Research.* New York: Gordon and Breach Science Publishers.

[17] U.S. Flight Standards Service. (1975). *Aviation Weather.* Washington, DC : Department of Transportation, Federal Aviation.

[18] U.S. Department of Transportation. (1996). *Advisory Circular. Subject: Effect of Icing on Aircraft Control and Airplane Deice and Anti-ice Systems.* Washington, DC: Department of Transportation, Federal Aviation Administration.

APPENDIX A: Translation of an AIRMET

Example of an Icing AIRMET

```
WA1Z
 BOSZ WA 011445
AIRMET ZULU UPDT 2 FOR ICE AND FRZLVL VALID UNTIL 012100
.
AIRMET ICE...ME NH AND CSTL WTRS
FROM 70NW PQI TO 30NNE PQI TO 50WSW YSJ TO 150ENE ACK TO ENE TO
YSC TO 70NW PQI
OCNL MOD RIME/MXD ICGICIP BTN FRZLVL AND FL200. FRZLVL 015-040 N
OF YQB-MLT LN...080 40E YSC-BGR LN...100 40SW YSC-30ENE ENE LN.
CONDS ENDG 13-15Z SW OF YSC-BGR LN. CONDS CONTG BYD 21Z ENDG BY
03Z.
.
FRZLVL...015-040 NE OF YQB-MLT LN.
         RPDLY RSG TO 080 40E YSC-BGR LN.
         SLPG 100 40SW YSC-30ENE ENE LN
         SLPG 120-130 RMNDR.
```

Translation of the Example Icing AIRMET

Zulu AIRMET in the Boston Zone on the 1st of the month at 2:45 pm.
Icing AIRMET updated 2 times for icing and freezing level until the 1st at 9:00pm.
Icing AIRMET for Maine, New Hampshire, and Coastal Waters.
Boundary Points for the icing polygon based on NAV AID points and variations from these points.
Occasional to Moderate Rime/Mixed Icing in clouds between freezing level and 20,000 feet.
Boundaries of the freezing level.
Describes where and when the conditions are ending.
Detailed description of the freezing level parameters.

APPENDIX B: Initial Zulu AIRMET Parsing Program for BOS Region

```
% Initial Zulu AIRMET Parsing Program for BOS Region

% Programmed by: Melinda Gates (04/2004)
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
%This program parses out the icing AIRMETS from the complete data set
%received from the NOAA
%-------------------------------------------------------------------------------

Data = fopen('data.txt','rt'); %Open File
AirmetsData = fopen('AirmetsData.txt','w');

while (not(feof(Data)))  % while not at the end of the file
   line=fgetl(Data);  % get a line
      if findstr(line,'DATA')  % if find the string WA1Z in line
         fprintf(AirmetsData,' %s\n',line); %print to file
            while isempty(findstr ('NEW MONTH', line)) %while not finding 'WAUS1'
               line=fgetl(Data);  %get a line
               if findstr(line, 'BOSZ') %If find icing AIRMET indicator for BOS region
                  fprintf(AirmetsData,'%s\n',line);  %print to file
                  while isempty(strfind(line, '....'))
                     line=fgetl(Data);  %get a line
                     fprintf(AirmetsData,'%s\n',line);
                  end;
               end;
            end;
      end;
end;

fclose(AirmetsData);
fclose(Data);
```

APPENDIX C: Detailed Extraction of Data from AIRMET

```
% Detailed Extraction of Data from AIRMET

% Programmed by: Melinda Gates and Nicolas Hinze (04/2004)
%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
%This program extracts the start time and end time of the AIRMETS, the
%polygon NAVAID points, and the minimum and maximum altitude.

clear all
clc


Data = fopen('Data.txt','rt');
AirmetsParsedData = fopen('AirmetsParsedData.txt','w');
numberoffreezing = 0;
Counter = 0;

%PARSE START TIMES, END TIMES, AND POLYGONS FROM AIRMET DATA
%-------------------------------------------------------------------------
while (not(feof(Data)))
   line=fgetl(Data);
   if findstr(line,'DATA')
      disp(line);
   end
   if findstr(line,'DATA')
      fprintf(AirmetsParsedData,'\n%s\n',line);
      while isempty(findstr('NEW MONTH', line))
         line=fgetl(Data);
         if strfind(line,'BOSZ')
            if((line(1, length(line)-2:length(line)))== 'AMD')
               starttime=(line(1, length(line)-9:length(line)));
               fprintf(AirmetsParsedData,'\n\n%s\n',starttime);
            elseif((line(1, length(line)-2:length(line)))== 'COR')
               starttime=(line(1, length(line)-9:length(line)));
               fprintf(AirmetsParsedData,'\n\n%s\n',starttime);
            else
               starttime = (line(1, length(line)-5:length(line)));
               fprintf(AirmetsParsedData,'\n\n%s\n',starttime);
            end;
            Counter = Counter + 1;
            polygon_counter = 0;
            altitude_counter = 0;
            WEATHER_OUTPUT(Counter).STARTTIME = (starttime);

            while isempty(strfind(line, '....'))
               line=fgetl(Data);
               if findstr(line, 'AIRMET ZULU')
                  endtime = line(1, length(line)-5:length(line));
                  fprintf(AirmetsParsedData,'%s\n',endtime);
               end;
               WEATHER_OUTPUT(Counter).ENDTIME = (endtime);

               if findstr(line, 'AIRMET ICE')
                  polygon_counter = polygon_counter + 1;
                  altitude_counter = altitude_counter + 1;
                  k=0;
                  fprintf(AirmetsParsedData,'\n');
```

```
while isempty(findstr(line, 'FROM'))
   line=fgetl(Data);
end;
FROM = findstr (line, 'FROM');
FROM2 = FROM + 5;
TO = findstr (line, ' TO');
TO2 = TO-1;
TO3 = TO + 3;
TO_WEATHER_OUTPUT = TO + 4;

if (length (TO2)>=1)
   initialindicator = line(1, FROM2:TO2);
   fprintf(AirmetsParsedData,'%s',initialindicator);
   if (polygon_counter == 1);
      WEATHER_OUTPUT(Counter).POLYGON1(1) = {line(1, FROM2:TO2)};
   elseif (polygon_counter ==2);
      WEATHER_OUTPUT(Counter).POLYGON2(1) = {line(1, FROM2:TO2)};
   elseif  (polygon_counter ==3);
      WEATHER_OUTPUT(Counter).POLYGON3(1) = {line(1, FROM2:TO2)};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(1) = {line(1, FROM2:TO2)};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(1) = {line(1, FROM2:TO2)};
   end;
else
   initialindicator = line(1, FROM2:length(line));
   fprintf(AirmetsParsedData,'%s',initialindicator);
   if (polygon_counter == 1);
      WEATHER_OUTPUT(Counter).POLYGON1(1) = {line(1, FROM2:length(line))};
   elseif (polygon_counter ==2);
      WEATHER_OUTPUT(Counter).POLYGON2(1) = {line(1, FROM2:length(line))};
   elseif  (polygon_counter ==3);
      WEATHER_OUTPUT(Counter).POLYGON3(1) = {line(1, FROM2:length(line))};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(1) = {line(1, FROM2:length(line))};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(1) = {line(1, FROM2:length(line))};
   end;
end;
while k<=9
   k=k+1;
   if ((length (TO2)==k) & isempty(strmatch(line(1, length(line)-2:length(line)),...
      ' TO','exact')))
      indicator = line(1, TO3(k):length(line));
      fprintf(AirmetsParsedData,'%s',indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1)= {line(1, TO_WEATHER_OUTPUT(k):length(line))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1)= {line(1, TO_WEATHER_OUTPUT(k):length(line))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1) = {line(1, TO_WEATHER_OUTPUT(k):length(line))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, TO_WEATHER_OUTPUT(k):length(line))};
      else
         WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
         .POLYGON5)+1) = {line(1, TO_WEATHER_OUTPUT(k):length(line))};
      end;
```

```
   elseif ((length (TO2)==k+1) & (strmatch(line(1, length(line)-2:length(line)),...
   ' TO','exact'))==1)
      indicator = line(1, TO3(k):TO2(k+1));
      fprintf(AirmetsParsedData,'%s',indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      else
         WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
         .POLYGON5)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      end;
   elseif (length (TO2)>=k+1)
      indicator = line(1, TO3(k):TO2(k+1));
      fprintf(AirmetsParsedData,'%s',indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      else
         WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
         .POLYGON5)+1) = {line(1, TO_WEATHER_OUTPUT(k):TO2(k+1))};
      end;
   end;
end;

line=fgetl(Data);
while isempty(findstr(line, 'OCNL')) & isempty(findstr(line, 'CANCEL'))...
& isempty (findstr(line, 'CONTG'))& isempty (findstr(line, 'CNCL'))
   m=0;
   line2TO = findstr (line, ' TO');
   line2TO2 = line2TO-1;
   line2TO3 = line2TO + 3;
   line2TO3_WEATHER_OUTPUT = line2TO + 4;

   if (isempty(findstr(line,' TO')) & (strmatch(line(1, 1:3),'TO ','exact'))==1)
      line2indicator1 = line(1, 4:length(line));
      fprintf(AirmetsParsedData,' %s',line2indicator1);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1) = {line2indicator1};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line2indicator1};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
```

```
      .POLYGON3)+1) = {line2indicator1};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
      .POLYGON4)+1) = {line2indicator1};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
      .POLYGON5)+1) = {line2indicator1};
   end;
elseif (length (line2TO2)>= 1 & (strmatch(line(1,1:3),'TO '))==1)
   line2indicator1 = line(1, 4:line2TO2);
   fprintf(AirmetsParsedData,' %s',line2indicator1);
   if (polygon_counter == 1);
      WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
      .POLYGON1)+1) = {line2indicator1};
   elseif (polygon_counter ==2);
      WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
      .POLYGON2)+1) = {line2indicator1};
   elseif  (polygon_counter ==3);
      WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
      .POLYGON3)+1) = {line2indicator1};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
      .POLYGON4)+1) = {line2indicator1};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
      .POLYGON5)+1) = {line2indicator1};
   end;
elseif (length (line2TO2)>=1 & isempty(strmatch(line(1,1:3),'TO ')))
   line2indicator1 = line(1, 1:line2TO2);
   fprintf(AirmetsParsedData,' %s',line2indicator1);
   if (polygon_counter == 1);
      WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
      .POLYGON1)+1) = {line2indicator1};
   elseif (polygon_counter ==2);
      WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
      .POLYGON2)+1) = {line2indicator1};
   elseif  (polygon_counter ==3);
      WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
      .POLYGON3)+1) = {line2indicator1};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
      .POLYGON4)+1) = {line2indicator1};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
      .POLYGON5)+1) = {line2indicator1};
   end;
elseif (isempty(findstr(line,' TO')) & isempty(strmatch(line(1,1:3),'TO ')))
   fprintf(AirmetsParsedData,' %s',line);
   if (polygon_counter == 1);
      WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
      .POLYGON1)+1) = {line};
   elseif (polygon_counter ==2);
      WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
      .POLYGON2)+1) = {line};
   elseif  (polygon_counter ==3);
      WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
      .POLYGON3)+1) = {line};
   elseif  (polygon_counter ==4);
      WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
      .POLYGON4)+1) = {line};
   else
      WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
```

```
      .POLYGON5)+1) = {line};
   end;
end;

while m<=8
   m=m+1;
   if (length (line2TO2)==m)& isempty(strmatch(line(1, length(line)-2:length(line)),...
   ' TO','exact'))
      line2indicator = line(1, line2TO3(m):length(line));
      fprintf(AirmetsParsedData,'%s',line2indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):length(line))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):length(line))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):length(line))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):length(line))};
      else
         WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
         .POLYGON5)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):length(line))};
      end;
   elseif (length (line2TO2)>=m+1)...
   & (strmatch(line(1, length(line)-2:length(line)),' TO','exact'))==1
      line2indicator = line(1, line2TO3(m):line2TO2(m+1));
      fprintf(AirmetsParsedData,'%s',line2indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter))...
         .POLYGON1)+1 = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      else
         WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
         .POLYGON5)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      end
   elseif (length (line2TO2)>=m+1)
      line2indicator = line(1, line2TO3(m):line2TO2(m+1));
      fprintf(AirmetsParsedData,'%s',line2indicator);
      if (polygon_counter == 1);
         WEATHER_OUTPUT(Counter).POLYGON1(length(WEATHER_OUTPUT(Counter)...
         .POLYGON1)+1)= {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif (polygon_counter ==2);
         WEATHER_OUTPUT(Counter).POLYGON2(length(WEATHER_OUTPUT(Counter)...
         .POLYGON2)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif  (polygon_counter ==3);
         WEATHER_OUTPUT(Counter).POLYGON3(length(WEATHER_OUTPUT(Counter)...
         .POLYGON3)+1)  = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      elseif  (polygon_counter ==4);
         WEATHER_OUTPUT(Counter).POLYGON4(length(WEATHER_OUTPUT(Counter)...
         .POLYGON4)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
      else
```

```
            WEATHER_OUTPUT(Counter).POLYGON5(length(WEATHER_OUTPUT(Counter)...
            .POLYGON5)+1) = {line(1, line2TO3_WEATHER_OUTPUT(m):line2TO2(m+1))};
         end;
      end;
   end;
   line=fgetl(Data);
end;
%PARSE MAX ALTITUDE AND MIN ALTITUDE FROM AIRMET DATA
%-----------------------------------------------------------------------------------------------
if findstr (line, 'ICGICIP FM')
   numberoffreezing = numberoffreezing + 1;
   startfreezinglevelindicator = findstr(line, ' FM');
   startfreezinglevel = startfreezinglevelindicator + 4;
   if findstr (line, ' TO ')
      endfreezinglevelindicator = findstr(line, ' TO ');
      endfreezinglevel = endfreezinglevelindicator - 1;
      freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
      startmaximumindicator = findstr(line, ' TO ');
      startmaximum = startmaximumindicator + 4;
      endmaximumindicator = findstr(line, '.');
      endmaximum = endmaximumindicator - 1;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      if findstr (maxfreezinglevel,'FL');
         startmaximum = startmaximumindicator + 6;
         endmaximumindicator = findstr(line, '.');
         endmaximum = endmaximumindicator - 1;
         maxfreezinglevel = line(1, startmaximum:endmaximum);
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      else
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      end;
   elseif findstr (line, ' AND ')
      endfreezinglevelindicator = findstr(line, ' AND ');
      endfreezinglevel = endfreezinglevelindicator - 1;
      freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
      startmaximumindicator = findstr(line, ' AND ');
      startmaximum = startmaximumindicator + 5;
      endmaximumindicator = findstr(line, '.');
      endmaximum = endmaximumindicator - 1;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      if findstr (maxfreezinglevel,'FL');
         startmaximum = startmaximumindicator + 7;
         endmaximumindicator = findstr(line, '.');
         endmaximum = endmaximumindicator - 1;
         maxfreezinglevel = line(1, startmaximum:endmaximum);
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      else
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      end;
   end;
   if strmatch (freezinglevel,'FRZLVL ')==1;
      if findstr(line, '.') & length(line)>endmaximumindicator(1)+6;
         if findstr (line, 'FRZLVL SLPG')
            line = fgetl(Data);
            freezinglevelnumeric = line(1,4:6);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         end;
      end;
   end;
```

```
elseif findstr (line, 'ICGICIP FRZ')
   numberoffreezing = numberoffreezing + 1;
   startfreezinglevelindicator = findstr(line, ' FR');
   startfreezinglevel = startfreezinglevelindicator + 1;
   endfreezinglevelindicator = findstr(line, ' TO ');
   endfreezinglevel = endfreezinglevelindicator - 1;
   freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
   startmaximumindicator = findstr(line, ' TO ');
   startmaximum = startmaximumindicator + 4;
   endmaximumindicator = findstr(line, '.');
   endmaximum = endmaximumindicator - 1;
   maxfreezinglevel = line(1, startmaximum:endmaximum);
   if findstr (maxfreezinglevel,'FL');
      startmaximum = startmaximumindicator + 6;
      endmaximumindicator = findstr(line, '.');
      endmaximum = endmaximumindicator - 1;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   else
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   end;
   if strmatch (freezinglevel,'FRZLVL ')==1;
      if findstr(line, '.') & length(line)>endmaximumindicator(1)+6;
         if findstr (line, 'FRZLVL AT OR NR')
            findnumericFL = findstr (line, 'FRZLVL AT OR NR');
            freezinglevelnumeric = line(1,findnumericFL+16:...
            findnumericFL+18);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif findstr (line, 'FRZLVL NR SFC')
            freezinglevelnumeric = line(1,endmaximumindicator+12:...
            endmaximumindicator+14);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif findstr (line, 'FRZLVL SFC')
            freezinglevelnumeric = line(1,endmaximumindicator+9:...
            endmaximumindicator+11);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif strmatch(line(1,endmaximumindicator(1)+2:...
         endmaximumindicator(1)+8),'FRZLVLS')==1
            freezinglevelnumeric = line(1,endmaximumindicator+10:...
            endmaximumindicator+12);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif strmatch(line(1,endmaximumindicator(1)+2:...
         endmaximumindicator(1)+7),'FRZLVL')==1
            freezinglevelnumeric = line(1,endmaximumindicator...
            +9:endmaximumindicator+11);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         end;
```

```
        end;
    end;
elseif findstr (line, ' ICGICIP FL')
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, ' FL');
    startfreezinglevel = startfreezinglevelindicator + 3;
    endfreezinglevelindicator = findstr(line, ' TO ');
    endfreezinglevel = endfreezinglevelindicator - 1;
    freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
    startmaximumindicator = findstr(line, ' TO ');
    startmaximum = startmaximumindicator + 4;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
        startmaximum = startmaximumindicator + 6;
        endmaximumindicator = findstr(line, '.');
        endmaximum = endmaximumindicator - 1;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
        fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    else
        fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    end;
    freezinglevelnumeric = freezinglevel;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'ICGICIP BLW TN FRZLVL AND ')
    numberoffreezing = numberoffreezing + 1;
    startmaximumindicator = findstr(line, ' AND ');
    startmaximum = startmaximumindicator + 5;
    endmaximum = startmaximum+2;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    freezinglevelnumeric = '000';
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr(line, ' ICGICIP BLW ')
    numberoffreezing = numberoffreezing + 1;
    startmaximumindicator = findstr(line, ' BLW ');
    startmaximum = startmaximumindicator + 5;
    if strmatch(line(1, startmaximumindicator+9:...
    startmaximumindicator+11),'AND')==1;
        endmaximum = startmaximum+2;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
        if findstr (maxfreezinglevel,'FL');
            startmaximum = startmaximumindicator + 7;
            endmaximumindicator = findstr(line, '.');
            endmaximum = endmaximumindicator - 1;
            maxfreezinglevel = line(1, startmaximum:endmaximum);
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        else
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        end;
        freezinglevelnumeric = '000';
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    elseif isempty(findstr(line,'.'))==1
        endmaximum = startmaximum + 3;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
        if findstr (maxfreezinglevel,'FL');
            startmaximum = startmaximumindicator + 7;
            endmaximum = startmaximum+2;
            maxfreezinglevel = line(1, startmaximum:endmaximum);
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        else
```

```
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        end;
        freezinglevelnumeric = '000';
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
        endmaximumindicator = findstr(line, '.');
        endmaximum = endmaximumindicator - 1;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
        if findstr (line, 'WITH MOST ICG BLW 080')
            maxfreezinglevel = line(1, startmaximum:startmaximum+2);
        end;
        if findstr (line, 'ERN HALF OF AREA')
            maxfreezinglevel = line(1, startmaximum:startmaximum+2);
        end;
        if findstr (line, 'NRN SXNS AND BLW 090 SRN SXNS.')
            maxfreezinglevel = line(1, startmaximum:startmaximum+2);
        end;
        if findstr (line, '-')
            maxfreezinglevel = line(1, startmaximum:startmaximum+2);
        end;
        if findstr (line, ' TO ')
            maxfreezinglevel = line(1,startmaximum:startmaximum+2);
        end;
        if findstr (maxfreezinglevel,'FL');
            startmaximum = startmaximumindicator + 7;
            endmaximumindicator = findstr(line, '.');
            endmaximum = endmaximumindicator - 1;
            maxfreezinglevel = line(1, startmaximum:endmaximum);
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        else
            fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
        end;
        freezinglevelnumeric = '000';
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif findstr(line, 'ICGICIP 120.')
    freezinglevel = '000';
    startmaximumindicator = findstr(line, 'CIP ');
    startmaximum = startmaximumindicator + 4;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    freezinglevelnumeric = freezinglevel;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr(line, 'ICGICIP 060')
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, 'CIP ');
    startfreezinglevel = startfreezinglevelindicator + 4;
    endfreezinglevelindicator = findstr(line, ' TO ');
    endfreezinglevel = endfreezinglevelindicator - 1;
    freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
    startmaximumindicator = findstr(line, ' TO ');
    startmaximum = startmaximumindicator + 4;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
        startmaximum = startmaximumindicator + 6;
        endmaximumindicator = findstr(line, '.');
        endmaximum = endmaximumindicator - 1;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
```

```
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   else
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   end;
   freezinglevelnumeric = freezinglevel;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr(line, ' ICGIC BLW ')
   numberoffreezing = numberoffreezing + 1;
   startmaximumindicator = findstr(line, ' BLW ');
   startmaximum = startmaximumindicator + 5;
   if strmatch(line(1, startmaximumindicator+9:...
   startmaximumindicator+11),'AND')==1;
      endmaximum = startmaximum+2;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      if findstr (maxfreezinglevel,'FL');
         startmaximum = startmaximumindicator + 7;
         endmaximumindicator = findstr(line, '.');
         endmaximum = endmaximumindicator - 1;
         maxfreezinglevel = line(1, startmaximum:endmaximum);
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      else
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      end;
      freezinglevelnumeric = '000';
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   elseif findstr (line, ...
   'OCNL MOD RIME ICGIC BLW 080 NRN HLF AND BLW 160 SRN HLF AREA.')
      endmaximum = startmaximum(1) + 2;
      maxfreezinglevel = line(1,startmaximum:endmaximum);
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      freezinglevelnumeric = '000';
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   else
      endmaximumindicator = findstr(line, '.');
      endmaximum = endmaximumindicator - 1;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      if findstr (maxfreezinglevel,'FL');
         startmaximum = startmaximumindicator + 7;
         endmaximumindicator = findstr(line, '.');
         endmaximum = endmaximumindicator - 1;
         maxfreezinglevel = line(1, startmaximum:endmaximum);
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      else
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      end;
      freezinglevelnumeric = '000';
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   end;
elseif strmatch(line(1, length(line)-13:length(line)), 'ABV FRZLVL AND')==1
   freezinglevel = 'FRZLVL';
   line = fgetl(Data);
   maxfreezinglevel = line(1,1:findstr(line,'.')-1);
   if findstr (maxfreezinglevel,'FL');
      maxfreezinglevel = line(1, 3:findstr(line,'.')-1);
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   else
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
   end;
   if strmatch (freezinglevel,'FRZLVL ')==1;
      if findstr(line, '.') & length(line)>endmaximumindicator(1)+6;
         freezinglevelnumericstart = findstr(line, '.')+9;
         freezinglevelnumericend = findstr(line, '.')+11;
```

```
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            if findstr (freezinglevelnumeric,'FL');
               freezinglevelnumeric = freezinglevelnumeric(3:5);
               fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
            else
               fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
            end;
         end;
      end;
   elseif (findstr(line, ' FROM ') & findstr(line, ' TO '));
      numberoffreezing = numberoffreezing + 1;
      startfreezinglevelindicator = findstr(line, ' FROM ');
      startfreezinglevel = startfreezinglevelindicator + 6;
      endfreezinglevelindicator = findstr(line, ' TO ');
      endfreezinglevel = endfreezinglevelindicator - 1;
      freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
      startmaximumindicator = findstr(line, ' TO ');
      startmaximum = startmaximumindicator + 4;
      endmaximumindicator = findstr(line, '.');
      endmaximum = endmaximumindicator - 1;
      maxfreezinglevel = line(1, startmaximum:endmaximum);
      if findstr (maxfreezinglevel,'FL');
         startmaximum = startmaximumindicator + 6;
         endmaximumindicator = findstr(line, '.');
         endmaximum = endmaximumindicator - 1;
         maxfreezinglevel = line(1, startmaximum:endmaximum);
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      else
         fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      end;
      if strmatch (freezinglevel,'FRZLVL ')==1;
         if findstr(line, '.') & length(line)>endmaximumindicator(1)+6;
            if length(line)>endmaximumindicator(1)+15
               freezinglevelnumericstart = endmaximumindicator+9;
               freezinglevelnumericend = endmaximumindicator+11;
               freezinglevelnumeric = line(1,freezinglevelnumericstart:...
               freezinglevelnumericend);
               if freezinglevelnumeric == 'SFC'
                  freezinglevelnumeric = '000';
               end;
               if findstr (freezinglevelnumeric,'FL');
                  freezinglevelnumeric = freezinglevelnumeric(3:5);
                  fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
               else
                  fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
               end;
            end;
         end;
      end;
   elseif findstr(line, 'CNCL AIRMET.')
      numberoffreezing = numberoffreezing + 1;
      maxfreezinglevel = '000';
      fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
      freezinglevelnumeric = '000';
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   elseif findstr(line,'CANCEL AIRMET.')
      numberoffreezing = numberoffreezing + 1;
      maxfreezinglevel = '000';
```

```
    fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    freezinglevelnumeric = '000';
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
  elseif (findstr(line, 'AND PRECIP BTN FRZLVL AND FL220.'))
    numberoffreezing = numberoffreezing +1;
    startfreezinglevelindicator = findstr(line, ' BTN ');
    startfreezinglevel = startfreezinglevelindicator + 5;
    endfreezinglevelindicator = findstr(line, ' AND FL220.');
    endfreezinglevel = endfreezinglevelindicator - 1;
    freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
    startmaximumindicator = findstr(line, ' AND FL220.');
    startmaximum = startmaximumindicator + 5;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
       startmaximum = startmaximumindicator + 7;
       endmaximumindicator = findstr(line, '.');
       endmaximum = endmaximumindicator - 1;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    else
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    end;
    if strmatch (freezinglevel,'FRZLVL ')==1;
       line = fgetl(Data);
       freezinglevelnumeric = line(1,8:10);
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
  elseif (findstr(line, ' BTN ') & findstr(line,' AND '));
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, ' BTN ');
    startfreezinglevel = startfreezinglevelindicator + 5;
    endfreezinglevelindicator = findstr(line, ' AND ');
    endfreezinglevel = endfreezinglevelindicator - 1;
    freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
    if (findstr(freezinglevel, '-'))
       freezinglevel = freezinglevel(1:3);
    end;
    if findstr (line, 'ICE BTN SFC AND 070 E CLE-EKN.')
       startmaximumindicator = findstr(line, ' AND ');
       startmaximum = startmaximumindicator + 5;
       endmaximumindicator = findstr(line, ' E ');
       endmaximum = endmaximumindicator - 1;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
    elseif findstr(line, '.')
       startmaximumindicator = findstr(line, ' AND ');
       startmaximum = startmaximumindicator + 5;
       endmaximumindicator = findstr(line, '.');
       endmaximum = endmaximumindicator - 1;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
    elseif strmatch(line(1, endfreezinglevelindicator+5:...
    endfreezinglevelindicator+6),'FL')==1
       startmaximumindicator = findstr(line, ' AND ');
       startmaximum = startmaximumindicator + 7;
       endmaximum = startmaximum + 2;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
    else
       startmaximumindicator = findstr(line, ' AND ');
```

```
   startmaximum = startmaximumindicator + 5;
   endmaximum = startmaximum + 2;
   maxfreezinglevel = line(1, startmaximum:endmaximum);
end;
if findstr (maxfreezinglevel,'FL');
   startmaximum = startmaximumindicator + 7;
   endmaximumindicator = findstr(line, '.');
   endmaximum = endmaximumindicator - 1;
   maxfreezinglevel = line(1, startmaximum:endmaximum);
   fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
else
   fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
end;
if strmatch (freezinglevel,'FRZLVL ')==1;
   if findstr(line, '.') & length(line)>endmaximumindicator(1)+6;
      if strmatch (line(1, length(line)-14:length(line)),...
      'FRZLVL AT OR NR')==1
         line = fgetl(Data);
         freezinglevelnumeric = line(1,1:3);
         if freezinglevelnumeric == 'SFC'
            freezinglevelnumeric = '000';
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      elseif findstr (line, 'CONDS MOVG EWD')
         line=fgetl(Data);
         line=fgetl(Data);
         line=fgetl(Data);
         if findstr (line, 'FRZLVL...')
            freezinglevelnumeric = line(1,10:12);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      elseif findstr (line, '     CONDS')
         line=fgetl(Data);
         line=fgetl(Data);
         line=fgetl(Data);
         if findstr (line, 'FRZLVL...')
            freezinglevelnumeric = line(1,10:12);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      elseif strmatch (line(1, length(line)-9:length(line)),...
         'CONDS MOVG')==1
         line=fgetl(Data);
         line=fgetl(Data);
         line=fgetl(Data);
         line=fgetl(Data);
         if findstr (line, 'FRZLVL...')
            freezinglevelnumeric = line(1,10:12);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      elseif strmatch (line(1, length(line)-13:length(line)),...
         'MULT FRZLV BTN')==1
         line=fgetl(Data);
         freezinglevelnumeric = line(1:3);
```

```
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'TIL 04Z FRQ MOD')
    line=fgetl(Data);
    line=fgetl(Data);
    freezinglevelnumeric = line(1:3);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1, length(line)-10:length(line)), ...
    'CONDS SLOLY')==1
    freezinglevelnumeric = 'SFC';
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL BLW')
    freezinglevelnumeric = line(1, endmaximumindicator+13:...
    endmaximumindicator+15);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL AT OR NR THE')
    line = fgetl(Data);
    freezinglevelnumeric = line(1,1:3);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1,length(line)-10:length(line)), ...
    'CONDS SPRDG')==1
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    if findstr (line, 'FRZLVL...')
       freezinglevelnumeric = line(1,10:12);
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1,length(line)-12:length(line)), ...
    'ALSO OCNL MOD')==1
    line = fgetl(Data);
    line = fgetl(Data);
    FRZLVLline3indicator = findstr(line, 'FRZLVL');
    FRZLVLline3 = FRZLVLline3indicator + 7;
    freezinglevelnumeric = line(1,FRZLVLline3:FRZLVLline3+2);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL AT OR NR')
    FRZLVLATORNR = findstr(line, ' NR ');
    freezinglevelnumeric = line(1,FRZLVLATORNR+4:FRZLVLATORNR+6);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
```

```
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1, length(line)-11:length(line)), ...
'FRZLVL AT OR')==1
    line = fgetl(Data);
    freezinglevelnumericstart = findstr (line, 'NR ')+3;
    freezinglevelnumericend = freezinglevelnumericstart+2;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL AT OR')
    freezinglevelnumericstart = endmaximumindicator+15;
    freezinglevelnumericend = endmaximumindicator+17;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'CONDS ENDG S OF A')
    freezinglevelnumeric = 'SFC';
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL MULT FRZLVL')
    line = fgetl(Data);
    freezinglevelnumericstart = 1;
    freezinglevelnumericend = 3;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif findstr (line, 'FRZLVL MULT')
    line = fgetl(Data);
    if strmatch (line(1, 1:11), 'FRZLVLS BLW')==1
       freezinglevelnumericstart = 13;
       freezinglevelnumericend = 15;
       freezinglevelnumeric = ...
       line(1,freezinglevelnumericstart:freezinglevelnumericend);
    else
       freezinglevelnumericstart = 1;
       freezinglevelnumericend = 3;
       freezinglevelnumeric = ...
       line(1,freezinglevelnumericstart:freezinglevelnumericend);
    end;
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
```

```
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif strmatch(line(1,length(line)-16:length(line)),...
    'FRZLVL NRN HLF AT')==1
    line = fgetl(Data);
    freezinglevelnumeric = line(1,7:9);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr(line,
'OCNL MOD MXD/CLR ICGICIP BTN FRZLVL AND FL220. MULT FRZLVLS')
    line = fgetl(Data);
    freezinglevelnumeric = line(1,1:3);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch(line(1,length(line)-14:length(line)),...
    'MULT FRZLVL BTN')==1
    line = fgetl(Data);
    freezinglevelnumeric = line(1,1:3);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'MULT FRZLVLS BLW')
    line = fgetl(Data);
    freezinglevelnumericstart = 1;
    freezinglevelnumericend = 3;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif strmatch(line(1,endmaximumindicator(1)+2:...
    endmaximumindicator(1)+4),'020')==1
    freezinglevelnumeric = '020';
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch(line(1,endmaximumindicator(1)+2:...
endmaximumindicator(1)+4),'SFC')==1
    freezinglevelnumeric = '000';
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1, length(line)-13:length(line)),...
'FRZLVL NRN PTN')==1
    line = fgetl(Data);
    freezinglevelnumeric = line(1,1:3);
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif length(line)>endmaximumindicator(1)+10 ...
& strmatch(line(1, endmaximumindicator(1)+2:endmaximumindicator(1)+8),...
'FRZLVL ','exact')==1;
    if strmatch(line(1, endmaximumindicator(1)+9:...
    endmaximumindicator(1)+10),'NR')==1;
       freezinglevelnumericstart = endmaximumindicator+12;
       freezinglevelnumericend = endmaximumindicator+14;
```

```
      freezinglevelnumeric = ...
      line(1,freezinglevelnumericstart:freezinglevelnumericend);
   else
      freezinglevelnumericstart = endmaximumindicator+9;
      freezinglevelnumericend = endmaximumindicator+11;
      freezinglevelnumeric = ...
      line(1,freezinglevelnumericstart:freezinglevelnumericend);
   end;
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   if findstr (freezinglevelnumeric,'FL');
      freezinglevelnumeric = freezinglevelnumeric(3:5);
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   else
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   end;
elseif findstr (line, 'MULT FRZLVL SFC-')
   freezinglevelnumeric == 'SFC';
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL SFC')
   freezinglevelnumericstart = endmaximumindicator(1)+8;
   freezinglevelnumericend = endmaximumindicator(1)+10;
   freezinglevelnumeric = line(1,freezinglevelnumericstart:...
   freezinglevelnumericend);
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'FRZLVL..SFC-')
   freezinglevelnumericstart = endmaximumindicator(1)+10;
   freezinglevelnumericend = endmaximumindicator(1)+12;
   freezinglevelnumeric = line(1,freezinglevelnumericstart:...
   freezinglevelnumericend);
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'CONDS DVLPG 06Z AND')
   freezinglevelnumeric = 'SFC';
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr (line, 'CONDS OVR WRN')
   line = fgetl(Data);
   line = fgetl(Data);
   line = fgetl(Data);
   if length(line)==1
      line = fgetl(Data);
   end;
   freezinglevelnumeric = line(1,19:21);
   if freezinglevelnumeric == 'SFC'
      freezinglevelnumeric = '000';
   end;
   fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif strmatch (line(1,length(line)-14:length(line)), ...
'CONDS CONTG BYD')==1
   line = fgetl(Data);
```

```
    line = fgetl(Data);
    if isempty (findstr(line,'AIRMET ICE...'))==1
       line = fgetl(Data);
       if length(line)>7
          if findstr (line, 'FRZLVL...03Z...AT OR NR')
             NRindicator = findstr(line, ' NR ');
             freezinglevelnumeric = line(1, NRindicator+...
             4:NRindicator+6);
          else
             freezinglevelnumeric = line(1,10:12);
          end;
       end;
    else
       freezinglevelnumeric = 'SFC';
    end;
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
 elseif strmatch (line(1,length(line)-10:length(line)), ...
    'CONDS CONTG')==1
    line = fgetl(Data);
    line = fgetl(Data);
    if isempty (findstr(line,'AIRMET ICE...'))==1
       line = fgetl(Data);
       if length(line)>7
          if findstr (line, 'FRZLVL...03Z...AT OR NR')
             NRindicator = findstr(line, ' NR ');
             freezinglevelnumeric = line(1, ...
             NRindicator+4:NRindicator+6);
          else
             freezinglevelnumeric = line(1,10:12);
          end;
       end;
    else
       freezinglevelnumeric = 'SFC';
    end;
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
 elseif strmatch (line(1,length(line)-18:length(line)), ...
 'CONDS CONTG BYD 09Z')==1
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    if length(line)>7
       if findstr (line, 'FRZLVL...03Z...AT OR NR')
          NRindicator = findstr(line, ' NR ');
          freezinglevelnumZric = line(1, NRindicator+4:NRindicator+6);
       else
```

```
            freezinglevelnumeric = line(1,10:12);
        end;
    end;
    if freezinglevelnumeric == 'SFC'
        freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
        freezinglevelnumeric = freezinglevelnumeric(3:5);
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif strmatch (line(1, length(line)-16:length(line)), 'CONDS ENDG FM THE')
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    line = fgetl(Data);
    freezinglevelnumeric = line(1,14:16);
    if freezinglevelnumeric == 'SFC'
        freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
        freezinglevelnumeric = freezinglevelnumeric(3:5);
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif findstr (line, 'MULT FRZLVLS') & length(line)>...
endmaximumindicator(1)+13;
    if strmatch(line(1,endmaximumindicator+17),'-')==1
        freezinglevelnumericstart = endmaximumindicator+15;
        freezinglevelnumericend = endmaximumindicator+16;
        freezinglevelnumeric = ...
        line(1,freezinglevelnumericstart:freezinglevelnumericend);
    else
        freezinglevelnumericstart = endmaximumindicator+15;
        freezinglevelnumericend = endmaximumindicator+17;
        freezinglevelnumeric = ...
        line(1,freezinglevelnumericstart:freezinglevelnumericend);
        if freezinglevelnumeric == 'SFC'
            freezinglevelnumeric = '000';
        end;
    end;
    if findstr (freezinglevelnumeric,'FL');
        freezinglevelnumeric = freezinglevelnumeric(3:5);
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif findstr (line,'FRZLVLS') & length(line)>endmaximumindicator(1)+11;
    if strmatch(line(1, length(line)-2:length(line)),'BLW')==1
        line = fgetl(Data);
        freezinglevelnumericstart = 1;
        freezinglevelnumericend = 3;
        freezinglevelnumeric = ...
        line(1,freezinglevelnumericstart:freezinglevelnumericend);
    else
        freezinglevelnumericstart = endmaximumindicator+10;
        freezinglevelnumericend = endmaximumindicator+12;
        freezinglevelnumeric = ...
        line(1,freezinglevelnumericstart:freezinglevelnumericend);
```

```
    end;
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif length(line)>endmaximumindicator(1)+15 & strmatch(line(1, ...
endmaximumindicator(1)+7:endmaximumindicator(1)+12),'FRZLVL',...
'exact')==1;
    freezinglevelnumericstart = endmaximumindicator+14;
    freezinglevelnumericend = endmaximumindicator+16;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif length(line)>endmaximumindicator(1)+11 ...
& strmatch(line(1, length(line)-5:length(line)),'FRZLVL','exact')==1;
    line = fgetl(Data);
    freezinglevelnumericstart = 1;
    freezinglevelnumericend = 3;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif length(line)>endmaximumindicator(1)+11 & strmatch ...
(line(1, endmaximumindicator(1)+7:endmaximumindicator(1)+12),...
'FRZLVL','exact')==1;
    line = fgetl(Data);
    freezinglevelnumericstart = 1;
    freezinglevelnumericend = 3;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    if findstr (freezinglevelnumeric,'FL');
       freezinglevelnumeric = freezinglevelnumeric(3:5);
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    else
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif length(line)==findstr(line,'.')+7;
    line=fgetl(Data);
    freezinglevelnumericstart = 1;
```

```
            freezinglevelnumericend = 3;
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            if findstr (freezinglevelnumeric,'FL');
               freezinglevelnumeric = freezinglevelnumeric(3:5);
               fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
            else
               fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
            end;
         end;
      elseif findstr(line, '.') & line(length(line))=='.'
         if strmatch (line(1, length(line)-28:length(line)), ...
            'ICGICIP BTN FRZLVL AND FL220.')==1
            line = fgetl(Data);
            line = fgetl(Data);
            line = fgetl(Data);
            freezinglevelnumeric = line(1,10:12);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         else
            line=fgetl(Data);
            freezinglevelnumericstart = 8;
            freezinglevelnumericend = 10;
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         end;
      elseif isempty(findstr(line,'.'))
         line=fgetl(Data);
         line2FRZLVL = findstr(line,'FRZLVL');
         freezinglevelnumericstart = line2FRZLVL + 7;
         freezinglevelnumericend = freezinglevelnumericstart + 2;
         freezinglevelnumeric = line(1,freezinglevelnumericstart:...
         freezinglevelnumericend);
         if freezinglevelnumeric == 'SFC'
            freezinglevelnumeric = '000';
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);

      end;
   else
      if findstr (freezinglevel,'FL');
         freezinglevelnumeric = freezinglevel(3:5);
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      elseif strmatch (freezinglevel,'SFC')==1;
         freezinglevelnumeric = freezinglevel;
         freezinglevelnumeric = '000';
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      else
         freezinglevelnumeric = freezinglevel;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      end;
   end;
elseif (findstr(line, ' BTN ') & findstr(line,' TO '));
```

```
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, ' BTN ');
    startfreezinglevel = startfreezinglevelindicator + 5;
    endfreezinglevelindicator = findstr(line, ' TO ');
    endfreezinglevel = endfreezinglevelindicator - 1;
    freezinglevel = line(1,startfreezinglevel:endfreezinglevel);
    startmaximumindicator = findstr(line, ' TO ');
    startmaximum = startmaximumindicator + 4;
    endmaximum = startmaximum + 2;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
       startmaximum = startmaximumindicator + 6;
       endmaximumindicator = findstr(line, '.');
       endmaximum = endmaximumindicator - 1;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    else
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    end;
    if strmatch (freezinglevel,'FRZLVL')==1;
       if findstr(line, 'FRZLVLS');
          freezinglevelnumericstart = endmaximumindicator+10;
          freezinglevelnumericend = endmaximumindicator+12;
          freezinglevelnumeric = line(1,freezinglevelnumericstart:...
          freezinglevelnumericend);
          if freezinglevelnumeric == 'SFC'
             freezinglevelnumeric = '000';
          end;
          fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
       end;
    else
       freezinglevelnumeric = freezinglevel;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
 elseif findstr(line, ' ABV ') & findstr(line, ' TO ')
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, ' ABV ');
    startfreezinglevel = startfreezinglevelindicator + 5;
    endfreezinglevelindicator = findstr(line, ' TO ');
    endfreezinglevel = endfreezinglevelindicator-1;
    freezinglevel = line(1, startfreezinglevel:endfreezinglevel);
    if (findstr(freezinglevel, '-'))
       freezinglevel = freezinglevel(1:3);
    end;
    startmaximumindicator = findstr(line, ' TO ');
    startmaximum = startmaximumindicator + 4;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
       startmaximum = startmaximumindicator + 6;
       endmaximumindicator = findstr(line, '.');
       endmaximum = endmaximumindicator - 1;
       maxfreezinglevel = line(1, startmaximum:endmaximum);
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    else
       fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    end;
    if strmatch (freezinglevel,'FRZLVL')==1;
       if strmatch(line(1,length(line)-9:length(line)),'FRZLVL BLW')==1;
          line = fgetl(Data);
          freezinglevelnumericstart = 1;
```

```
    freezinglevelnumericend = 3;
    freezinglevelnumeric = line(1,freezinglevelnumericstart:...
    freezinglevelnumericend);
    if freezinglevelnumeric == 'SFC'
       freezinglevelnumeric = '000';
    end;
    fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
elseif findstr(line, '.') & length(line)>endmaximumindicator(1)+2;
    if length(line)>endmaximumindicator(1)+10 & strmatch(line(1, ...
       endmaximumindicator(1)+2:endmaximumindicator(1)+8),'FRZLVL ',...
       'exact')==1;
       freezinglevelnumericstart = endmaximumindicator+9;
       freezinglevelnumericend = endmaximumindicator+11;
       freezinglevelnumeric = line(1,freezinglevelnumericstart:...
       freezinglevelnumericend);
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    elseif strmatch(line(1,endmaximumindicator+2:...
    endmaximumindicator+4), 'SFC')==1
       freezinglevelnumeric = line (1, endmaximumindicator+2:...
       endmaximumindicator+4);
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    elseif strmatch (line(1, length(line)-12:length(line)), 'MULTI FRZLVLS')==1
       line = fgetl(Data);
       if strmatch (line(1, 1:3), 'BLW')==1
          freezinglevelnumeric = line(1,5:7);
       end;
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    elseif length(line)>endmaximumindicator(1)+11 & findstr (line,'FRZLVLS');
       freezinglevelnumericstart = endmaximumindicator+10;
       freezinglevelnumericend = endmaximumindicator+12;
       freezinglevelnumeric = line(1,freezinglevelnumericstart:...
       freezinglevelnumericend);
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    elseif length(line)==findstr(line,'.') + 7;
       line=fgetl(Data);
       if strmatch (line(1, 1:4),'SLPG', 'exact')==1
          freezinglevelnumericstart = 6;
          freezinglevelnumericend = 8;
          freezinglevelnumeric = line(1,...
          freezinglevelnumericstart:freezinglevelnumericend);
       else
          freezinglevelnumericstart = 1;
          freezinglevelnumericend = 3;
          freezinglevelnumeric = line(1,...
          freezinglevelnumericstart:freezinglevelnumericend);
       end;
       if freezinglevelnumeric == 'SFC'
          freezinglevelnumeric = '000';
       end;
       fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
```

```
            end;
        elseif findstr(line, '.') & length(line)-1=='.'
            line=fgetl(Data)
            freezinglevelnumericstart = 8;
            freezinglevelnumericend = 10;
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
                freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
        end;

    else
        freezinglevelnumeric = freezinglevel;
        fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
    end;
elseif findstr(line, ' ABV ') & findstr(line, ' AND ')
    numberoffreezing = numberoffreezing + 1;
    startfreezinglevelindicator = findstr(line, ' ABV ');
    startfreezinglevel = startfreezinglevelindicator + 5;
    endfreezinglevelindicator = findstr(line, ' AND ');
    endfreezinglevel = endfreezinglevelindicator-1;
    freezinglevel = line(1, startfreezinglevel:endfreezinglevel);
    if (findstr(freezinglevel, '-'))
        freezinglevel = freezinglevel(1:3);
    end;
    startmaximumindicator = findstr(line, ' AND ');
    startmaximum = startmaximumindicator + 5;
    endmaximumindicator = findstr(line, '.');
    endmaximum = endmaximumindicator - 1;
    maxfreezinglevel = line(1, startmaximum:endmaximum);
    if findstr (maxfreezinglevel,'FL');
        endmaximumindicator = findstr(line, '.');
        startmaximum = endmaximumindicator(1)-3;
        endmaximum = endmaximumindicator(1) - 1;
        maxfreezinglevel = line(1, startmaximum:endmaximum);
        fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    else
        fprintf(AirmetsParsedData,'\n%s',maxfreezinglevel);
    end;
    if strmatch (freezinglevel,'FRZLVL')==1;
        if findstr(line, '.') & length(line)>endmaximumindicator(1)+2;
            if strmatch(line(1,length(line)-10:length(line)),'FRZLVL SLPG')==1;
                line = fgetl(Data);
                freezinglevelnumericstart = 1;
                freezinglevelnumericend = 3;
                freezinglevelnumeric = line(1,freezinglevelnumericstart:...
                freezinglevelnumericend);
                if freezinglevelnumeric == 'SFC'
                    freezinglevelnumeric = '000';
                end;
                fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
            elseif (length(line)>endmaximumindicator(1)+10 & strmatch(line(1,...
            endmaximumindicator(1)+2:endmaximumindicator(1)+8),'FRZLVL ', 'exact')==1);
                freezinglevelnumericstart = endmaximumindicator+9;
                freezinglevelnumericend = endmaximumindicator+11;
                freezinglevelnumeric = line(1,freezinglevelnumericstart:...
                freezinglevelnumericend);
                if freezinglevelnumeric == 'SFC'
                    freezinglevelnumeric = '000';
                end;
```

```
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif strmatch(line(1,length(line)-6:length(line)),'FRZLVLS')==1
            line = fgetl(Data);
            freezinglevelnumeric = line(1,1:3);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif findstr (line,'FRZLVLS') & length(line)>endmaximumindicator(1)+11;
            freezinglevelnumericstart = endmaximumindicator+10;
            freezinglevelnumericend = endmaximumindicator+12;
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         elseif length(line)==findstr(line,'.')+7;
            line=fgetl(Data);
            freezinglevelnumericstart = 1;
            freezinglevelnumericend = 3;
            freezinglevelnumeric = line(1,freezinglevelnumericstart:...
            freezinglevelnumericend);
            if freezinglevelnumeric == 'SFC'
               freezinglevelnumeric = '000';
            end;
            fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
         end;
      elseif findstr(line, '.') & length(line)-1=='.'
         line=fgetl(Data);
         freezinglevelnumericstart = 8;
         freezinglevelnumericend = 10;
         freezinglevelnumeric = line(1,freezinglevelnumericstart:...
         freezinglevelnumericend);
         if freezinglevelnumeric == 'SFC'
            freezinglevelnumeric = '000';
         end;
         fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
      end;

   else
      freezinglevelnumeric = freezinglevel;
      fprintf(AirmetsParsedData,'\n%s',freezinglevelnumeric);
   end;
end;
 if (altitude_counter == 1);
   str2num(maxfreezinglevel);
   str2num(freezinglevelnumeric);
   WEATHER_OUTPUT(Counter).ALTITUDEMAX1 = (maxfreezinglevel);
   WEATHER_OUTPUT(Counter).ALTITUDEMIN1 = (freezinglevelnumeric);
elseif (altitude_counter ==2);
   str2num(maxfreezinglevel);
   str2num(freezinglevelnumeric);
   WEATHER_OUTPUT(Counter).ALTITUDEMAX2 = {maxfreezinglevel};
   WEATHER_OUTPUT(Counter).ALTITUDEMIN2 = {freezinglevelnumeric};
elseif  (altitude_counter ==3);
   str2num(maxfreezinglevel);
   str2num(freezinglevelnumeric);
   WEATHER_OUTPUT(Counter).ALTITUDEMAX3 = {maxfreezinglevel};
   WEATHER_OUTPUT(Counter).ALTITUDEMIN3 = {freezinglevelnumeric};
elseif  (altitude_counter ==4);
   str2num(maxfreezinglevel);
```

```
                    str2num(freezinglevelnumeric);
                    WEATHER_OUTPUT(Counter).ALTITUDEMAX4 = {maxfreezinglevel};
                    WEATHER_OUTPUT(Counter).ALTITUDEMIN4 = {freezinglevelnumeric};
                 else
                    str2num(maxfreezinglevel);
                    str2num(freezinglevelnumeric);
                    WEATHER_OUTPUT(Counter).ALTITUDEMAX5 = {maxfreezinglevel};
                    WEATHER_OUTPUT(Counter).ALTITUDEMIN5 = {freezinglevelnumeric};
                 end;
              end;
           end;
        end;
        if findstr(line, 'NEW MONTH')
           disp(num2str(Counter));
        end;
     end;
  end;
end;


fclose(AirmetsParsedData);
fclose(Data);
save WEATHER_OUTPUT WEATHER_OUTPUT
```

APPENDIX D: AMD and COR for BOSCLE 12,000 Feet

```
% AMDandCORforBOSCLE12,000Feet

% Programmed by: Melinda Gates (08/2004)
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%This program corrected the end times for AIRMETS that had corrections or
%amendments issued to keep AIRMETS from being counted two times.
%--------------------------------------------------------------------------

clear all;

DataRead = fopen('StarttimeandEndtimeNoRepeatBOSCLE.txt','rt');
AMDdata = fopen('DataWithoutAMDboscle.txt','w');

%Get information for AIRMET number 1

AIRMETnumber = fgetl(DataRead);
Starttime = fgetl(DataRead);
Endtime = fgetl(DataRead);
Space = fgetl(DataRead);

%Get information for AIRMET number 2

while(not(feof(DataRead)))
AIRMETnumber2 = fgetl(DataRead);
Starttime2 = fgetl(DataRead);
Endtime2 = fgetl(DataRead);
Space2 = fgetl(DataRead);

%If an AMD or COR was made to AIRMET number 1 rewrite
%the end time of it as the start time of the AMD or
%COR AIRMET.

if findstr('AMD', Starttime2) & Endtime==Endtime2
   Endtime = Starttime2(1:6);
end;
if findstr('COR', Starttime2) & Endtime==Endtime2
   Endtime = Starttime2(1:6);
end;

   fprintf(AMDdata, '%s\n', AIRMETnumber);
   fprintf(AMDdata, '%s\n', Starttime(1:6));
   fprintf(AMDdata, '%s\n', Endtime);
   fprintf(AMDdata, '%s\n', Space);

   AIRMETnumber = AIRMETnumber2;
   Starttime = Starttime2;
   Endtime = Endtime2;
   Space = Space2;
end;

fclose(DataRead);
fclose(AMDdata);
```

APPENDIX E: FindNAV Program

```
% FindNAV Program

% Programmed by: Melinda Gates and Nicolas Hinze (04/2004)
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%This program matches the NAVAID points to their corresponding longitude
%and latitude values.
%-------------------------------------------------------------------------

clear all;

DataParsingProgramFINAL

load NAV_LIST

for j = 1:length(WEATHER_OUTPUT)
   breakout = 0;

%For Polygon 1

   if length(WEATHER_OUTPUT(j).POLYGON1)>0
      for k = 1:length(WEATHER_OUTPUT(j).POLYGON1)

%Locates if there is a variation from the NAVAID point, and stores the
%values as direction.

         if sum(isletter(char(WEATHER_OUTPUT(j).POLYGON1(k))))<...
         length(char(WEATHER_OUTPUT(j).POLYGON1(k)))
            distpoint = char(WEATHER_OUTPUT(j).POLYGON1(k));
            direction = char(WEATHER_OUTPUT(j).POLYGON1(k));
            if sum(isspace(char(WEATHER_OUTPUT(j).POLYGON1(k))))>0
               if sum(isletter(distpoint(1:3)))==0;
                  distpoint = str2num(distpoint(1:3));
                  if sum(isletter(direction(4:6)))==3;
                     direction = direction (4:6);
                  elseif sum(isletter(direction(4:5)))==2;
                     direction = direction(4:5);
                  else
                     direction = direction(4:4);
                  end
               elseif sum(isletter(distpoint(1:2)))==0;
                  distpoint = str2num(distpoint(1:2));
                  if sum(isletter(direction(3:5)))==3;
                     direction = direction (3:5);
                  elseif sum(isletter(direction(3:4)))==2;
                     direction = direction(3:4);
                  else
                     direction = direction(3:3);
                  end
               else
                  distpoint = str2num(distpoint(1:1));
                  if sum(isletter(direction(2:4)))==3;
                     direction = direction (2:4);
                  elseif sum(isletter(direction(2:3)))==2;
                     direction = direction(2:3);
                  else
                     direction = direction(2:2);
                  end
               end
```

```
        else
            if sum(isletter(distpoint(1:3)))==0;
                distpoint = str2num(distpoint(1:3));
                if sum(isletter(direction(4:length(direction))))==3;
                    direction = direction (4:length(direction));
                elseif sum(isletter(direction(4:length(direction))))==2;
                    direction = direction(4:length(direction));
                else
                    direction = direction(4:length(direction));
                end
            elseif sum(isletter(distpoint(1:2)))==0;
                distpoint = str2num(distpoint(1:2));
                if sum(isletter(direction(3:length(direction))))==3;
                    direction = direction (3:length(direction));
                elseif sum(isletter(direction(3:length(direction))))==2;
                    direction = direction(3:length(direction));
                else
                    direction = direction(3:length(direction));
                end
            else
                distpoint = str2num(distpoint(1:1));
                if sum(isletter(direction(2:length(direction))))==3;
                    direction = direction (2:length(direction));
                elseif sum(isletter(direction(2:length(direction))))==2;
                    direction = direction(2:length(direction));
                else
                    direction = direction(2:length(direction));
                end
            end
        end

%Matches Direction to correct numerical value deviation.

        if strcmp(direction,'N')==1
            directiondegrees = 0;
        elseif strcmp(direction,'S')==1
            directiondegrees = 180;
        elseif strcmp(direction,'E')==1
            directiondegrees = 90;
        elseif strcmp(direction,'W')==1
            directiondegrees = 270;
        elseif strcmp(direction,'NNE')==1
            directiondegrees = 22.5;
        elseif strcmp(direction,'NE')==1
            directiondegrees = 45;
        elseif strcmp(direction,'ENE')==1
            directiondegrees = 67.5;
        elseif strcmp(direction,'ESE')==1
            directiondegrees = 112.5;
        elseif strcmp(direction,'SE')==1
            directiondegrees = 135;
        elseif strcmp(direction,'SSE')==1
            directiondegrees = 157.5;
        elseif strcmp(direction,'SSW')==1
            directiondegrees = 202.5;
        elseif strcmp(direction,'SW')
            directiondegrees = 225;
        elseif strcmp(direction,'WSW')==1
            directiondegrees = 247.5;
        elseif strcmp(direction,'WNW')==1
            directiondegrees = 292.5;
        elseif strcmp(direction,'NW')==1
```

```
            directiondegrees = 315;
        elseif strcmp(direction,'NNW')==1
            directiondegrees = 337.5;
        end

        NAVpoint = char(WEATHER_OUTPUT(j).POLYGON1(k));
        if sum(isspace(NAVpoint))>0
            NAVpoint = NAVpoint(length(NAVpoint)-2:length(NAVpoint));
            breakout = breakout+0;
        else
            NAVpoint = char(WEATHER_OUTPUT(j).POLYGON1(k+1));
            WEATHER_OUTPUT(j).POLYGON1(k)={strcat(char(WEATHER_OUTPUT(j).POLYGON1(k)), ...
            char(WEATHER_OUTPUT(j).POLYGON1(k+1)))};
            WEATHER_OUTPUT(j).POLYGON1(k+1)=[];
            breakout = breakout+1;
        end

%Matches NAVAID indicator to the longitude and latitude values.

        for i = 1:length(NAV_LIST)
            if length(NAV_LIST(i).ID) == length(NAVpoint)
                if NAV_LIST(i).ID == NAVpoint
                    NAVpointLAT= NAV_LIST(i).Latitude;
                    NAVpointLONG= NAV_LIST(i).Longitude;
                    break
                end
            end
        end
        [NEWLAT,NEWLONG] = reckon(NAVpointLAT,NAVpointLONG,nm2deg(distpoint),directiondegrees);
        WEATHER_OUTPUT(j).LAT1(k)= NEWLAT;
        WEATHER_OUTPUT(j).LONG1(k)= NEWLONG;

        if breakout >= 1
            if k <= length(WEATHER_OUTPUT(j).POLYGON1)-1
                k=k+1;
            else
                break;
            end
        end

    else
        for i = 1:length(NAV_LIST)
            if length(NAV_LIST(i).ID) == length(char(WEATHER_OUTPUT(j).POLYGON1(k)))
                if NAV_LIST(i).ID == char(WEATHER_OUTPUT(j).POLYGON1(k))
                    WEATHER_OUTPUT(j).LAT1(k)= NAV_LIST(i).Latitude;
                    WEATHER_OUTPUT(j).LONG1(k)= NAV_LIST(i).Longitude;
                    break
                end
            end
        end

        if breakout == 1
            if k <= length(WEATHER_OUTPUT(j).POLYGON1)-1
                k=k+1;
            else
                break;
            end
        end

    end

end
```

```
   end
   breakout = 0;

 %For Polygon 2

   if length(WEATHER_OUTPUT(j).POLYGON2)>0
      for k = 1:length(WEATHER_OUTPUT(j).POLYGON2)

%Locates if there is a variation from the NAVAID point, and stores the
%values as direction.

         if sum(isletter(char(WEATHER_OUTPUT(j).POLYGON2(k))))<...
         length(char(WEATHER_OUTPUT(j).POLYGON2(k)))
            distpoint = char(WEATHER_OUTPUT(j).POLYGON2(k));
            direction = char(WEATHER_OUTPUT(j).POLYGON2(k));
            if sum(isspace(char(WEATHER_OUTPUT(j).POLYGON2(k))))>0
               if sum(isletter(distpoint(1:3)))==0;
                  distpoint = str2num(distpoint(1:3));
                  if sum(isletter(direction(4:6)))==3;
                     direction = direction (4:6);
                  elseif sum(isletter(direction(4:5)))==2;
                     direction = direction(4:5);
                  else
                     direction = direction(4:4);
                  end
               elseif sum(isletter(distpoint(1:2)))==0;
                  distpoint = str2num(distpoint(1:2));
                  if sum(isletter(direction(3:5)))==3;
                     direction = direction (3:5);
                  elseif sum(isletter(direction(3:4)))==2;
                     direction = direction(3:4);
                  else
                     direction = direction(3:3);
                  end
               else
                  distpoint = str2num(distpoint(1:1));
                  if sum(isletter(direction(2:4)))==3;
                     direction = direction (2:4);
                  elseif sum(isletter(direction(2:3)))==2;
                     direction = direction(2:3);
                  else
                     direction = direction(2:2);
                  end
               end
            else
               if sum(isletter(distpoint(1:3)))==0;
                  distpoint = str2num(distpoint(1:3));
                  if sum(isletter(direction(4:length(direction))))==3;
                     direction = direction (4:length(direction));
                  elseif sum(isletter(direction(4:length(direction))))==2;
                     direction = direction(4:length(direction));
                  else
                     direction = direction(4:length(direction));
                  end
               elseif sum(isletter(distpoint(1:2)))==0;
                  distpoint = str2num(distpoint(1:2));
                  if sum(isletter(direction(3:length(direction))))==3;
                     direction = direction (3:length(direction));
                  elseif sum(isletter(direction(3:length(direction))))==2;
                     direction = direction(3:length(direction));
                  else
                     direction = direction(3:length(direction));
```

```
            end
        else
            distpoint = str2num(distpoint(1:1));
            if sum(isletter(direction(2:length(direction))))==3;
                direction = direction (2:length(direction));
            elseif sum(isletter(direction(2:length(direction))))==2;
                direction = direction(2:length(direction));
            else
                direction = direction(2:length(direction));
            end
        end
    end

%Matches Direction to correct numerical value deviation.

    if strcmp(direction,'N')==1
        directiondegrees = 0;
    elseif strcmp(direction,'S')==1
        directiondegrees = 180;
    elseif strcmp(direction,'E')==1
        directiondegrees = 90;
    elseif strcmp(direction,'W')==1
        directiondegrees = 270;
    elseif strcmp(direction,'NNE')==1
        directiondegrees = 22.5;
    elseif strcmp(direction,'NE')==1
        directiondegrees = 45;
    elseif strcmp(direction,'ENE')==1
        directiondegrees = 67.5;
    elseif strcmp(direction,'ESE')==1
        directiondegrees = 112.5;
    elseif strcmp(direction,'SE')==1
        directiondegrees = 135;
    elseif strcmp(direction,'SSE')==1
        directiondegrees = 157.5;
    elseif strcmp(direction,'SSW')==1
        directiondegrees = 202.5;
    elseif strcmp(direction,'SW')
        directiondegrees = 225;
    elseif strcmp(direction,'WSW')==1
        directiondegrees = 247.5;
    elseif strcmp(direction,'WNW')==1
        directiondegrees = 292.5;
    elseif strcmp(direction,'NW')==1
        directiondegrees = 315;
    elseif strcmp(direction,'NNW')==1
        directiondegrees = 337.5;
    end
    NAVpoint = char(WEATHER_OUTPUT(j).POLYGON2(k));
    if sum(isspace(NAVpoint))>0
        NAVpoint = NAVpoint(length(NAVpoint)-2:length(NAVpoint));
        breakout = breakout+0;
    else
        NAVpoint = char(WEATHER_OUTPUT(j).POLYGON2(k+1));
        WEATHER_OUTPUT(j).POLYGON2(k)={strcat(char(WEATHER_OUTPUT(j).POLYGON2(k)),...
        char(WEATHER_OUTPUT(j).POLYGON2(k+1)))};
        WEATHER_OUTPUT(j).POLYGON2(k+1)=[];
        breakout = breakout+1;
    end

%Matches NAVAID indicator to the longitude and latitude values.
```

```
            for i = 1:length(NAV_LIST)
               if length(NAV_LIST(i).ID) == length(NAVpoint)
                  if NAV_LIST(i).ID == NAVpoint
                     NAVpointLAT= NAV_LIST(i).Latitude;
                     NAVpointLONG= NAV_LIST(i).Longitude;
                     break
                  end
               end
            end
            [NEWLAT,NEWLONG] = reckon(NAVpointLAT,NAVpointLONG,nm2deg(distpoint),directiondegrees);
            WEATHER_OUTPUT(j).LAT2(k)= NEWLAT;
            WEATHER_OUTPUT(j).LONG2(k)= NEWLONG;

            if breakout >= 1
               if k <= length(WEATHER_OUTPUT(j).POLYGON2)-1
                  k=k+1;
               else
                  break;
               end
            end

         else

            for i = 1:length(NAV_LIST)
               if length(NAV_LIST(i).ID) == length(char(WEATHER_OUTPUT(j).POLYGON2(k)))
                  if NAV_LIST(i).ID == char(WEATHER_OUTPUT(j).POLYGON2(k))
                     WEATHER_OUTPUT(j).LAT2(k)= NAV_LIST(i).Latitude;
                     WEATHER_OUTPUT(j).LONG2(k)= NAV_LIST(i).Longitude;
                     break
                  end
               end
            end

            if breakout >= 1
               if k <= length(WEATHER_OUTPUT(j).POLYGON2)-1
                  k=k+1;
               else
                  break;
               end
            end
         end

      end
   end

   breakout = 0;

 %For Polygon 3

   if length(WEATHER_OUTPUT(j).POLYGON3)>0
      for k = 1:length(WEATHER_OUTPUT(j).POLYGON3)

%Locates if there is a variation from the NAVAID point, and stores the
%values as direction.

         if sum(isletter(char(WEATHER_OUTPUT(j).POLYGON3(k))))<...
         length(char(WEATHER_OUTPUT(j).POLYGON3(k)))
            distpoint = char(WEATHER_OUTPUT(j).POLYGON3(k));
            direction = char(WEATHER_OUTPUT(j).POLYGON3(k));
            if sum(isspace(char(WEATHER_OUTPUT(j).POLYGON3(k))))>0
               if sum(isletter(distpoint(1:3)))==0;
                  distpoint = str2num(distpoint(1:3));
```

```
            if sum(isletter(direction(4:6)))==3;
               direction = direction (4:6);
            elseif sum(isletter(direction(4:5)))==2;
               direction = direction(4:5);
            else
               direction = direction(4:4);
            end
         elseif sum(isletter(distpoint(1:2)))==0;
            distpoint = str2num(distpoint(1:2));
            if sum(isletter(direction(3:5)))==3;
               direction = direction (3:5);
            elseif sum(isletter(direction(3:4)))==2;
               direction = direction(3:4);
            else
               direction = direction(3:3);
            end
         else
            distpoint = str2num(distpoint(1:1));
            if sum(isletter(direction(2:4)))==3;
               direction = direction (2:4);
            elseif sum(isletter(direction(2:3)))==2;
               direction = direction(2:3);
            else
               direction = direction(2:2);
            end
         end
      else
         if sum(isletter(distpoint(1:3)))==0;
            distpoint = str2num(distpoint(1:3));
            if sum(isletter(direction(4:length(direction))))==3;
               direction = direction (4:length(direction));
            elseif sum(isletter(direction(4:length(direction))))==2;
               direction = direction(4:length(direction));
            else
               direction = direction(4:length(direction));
            end
         elseif sum(isletter(distpoint(1:2)))==0;
            distpoint = str2num(distpoint(1:2));
            if sum(isletter(direction(3:length(direction))))==3;
               direction = direction (3:length(direction));
            elseif sum(isletter(direction(3:length(direction))))==2;
               direction = direction(3:length(direction));
            else
               direction = direction(3:length(direction));
            end
         else
            distpoint = str2num(distpoint(1:1));
            if sum(isletter(direction(2:length(direction))))==3;
               direction = direction (2:length(direction));
            elseif sum(isletter(direction(2:length(direction))))==2;
               direction = direction(2:length(direction));
            else
               direction = direction(2:length(direction));
            end
         end
      end

%Matches Direction to correct numerical value deviation.

      if strcmp(direction,'N')==1
         directiondegrees = 0;
      elseif strcmp(direction,'S')==1
```

```
        directiondegrees = 180;
    elseif strcmp(direction,'E')==1
        directiondegrees = 90;
    elseif strcmp(direction,'W')==1
        directiondegrees = 270;
    elseif strcmp(direction,'NNE')==1
        directiondegrees = 22.5;
    elseif strcmp(direction,'NE')==1
        directiondegrees = 45;
    elseif strcmp(direction,'ENE')==1
        directiondegrees = 67.5;
    elseif strcmp(direction,'ESE')==1
        directiondegrees = 112.5;
    elseif strcmp(direction,'SE')==1
        directiondegrees = 135;
    elseif strcmp(direction,'SSE')==1
        directiondegrees = 157.5;
    elseif strcmp(direction,'SSW')==1
        directiondegrees = 202.5;
    elseif strcmp(direction,'SW')
        directiondegrees = 225;
    elseif strcmp(direction,'WSW')==1
        directiondegrees = 247.5;
    elseif strcmp(direction,'WNW')==1
        directiondegrees = 292.5;
    elseif strcmp(direction,'NW')==1
        directiondegrees = 315;
    elseif strcmp(direction,'NNW')==1
        directiondegrees = 337.5;
    end
    NAVpoint = char(WEATHER_OUTPUT(j).POLYGON3(k));
    if sum(isspace(NAVpoint))>0
        NAVpoint = NAVpoint(length(NAVpoint)-2:length(NAVpoint));
        breakout = breakout+0;
    else
        NAVpoint = char(WEATHER_OUTPUT(j).POLYGON3(k+1));
        WEATHER_OUTPUT(j).POLYGON3(k)={strcat(char(WEATHER_OUTPUT(j).POLYGON3(k)), ...
        char(WEATHER_OUTPUT(j).POLYGON3(k+1)))};
        WEATHER_OUTPUT(j).POLYGON3(k+1)=[];
        breakout = breakout+1;
    end

%Matches NAVAID indicator to the longitude and latitude values.

    for i = 1:length(NAV_LIST)
        if length(NAV_LIST(i).ID) == length(NAVpoint)
            if NAV_LIST(i).ID == NAVpoint
                NAVpointLAT= NAV_LIST(i).Latitude;
                NAVpointLONG= NAV_LIST(i).Longitude;
                break
            end
        end
    end
    [NEWLAT,NEWLONG] = reckon(NAVpointLAT,NAVpointLONG,nm2deg(distpoint),directiondegrees);
    WEATHER_OUTPUT(j).LAT3(k)= NEWLAT;
    WEATHER_OUTPUT(j).LONG3(k)= NEWLONG;

    if breakout >= 1
        if k <= length(WEATHER_OUTPUT(j).POLYGON3)-1
            k=k+1;
        else
            break;
```

```
            end
        end

    else
        for i = 1:length(NAV_LIST)
            if length(NAV_LIST(i).ID) == length(char(WEATHER_OUTPUT(j).POLYGON3(k)))
                if NAV_LIST(i).ID == char(WEATHER_OUTPUT(j).POLYGON3(k))
                    WEATHER_OUTPUT(j).LAT3(k)= NAV_LIST(i).Latitude;
                    WEATHER_OUTPUT(j).LONG3(k)= NAV_LIST(i).Longitude;
                    break
                end
            end
        end

        if breakout >= 1
            if k <= length(WEATHER_OUTPUT(j).POLYGON3)-1
                k=k+1;
            else
                break;
            end
        end
    end

  end
end

breakout = 0;

%For Polygon 4

if length(WEATHER_OUTPUT(j).POLYGON4)>0
  for k = 1:length(WEATHER_OUTPUT(j).POLYGON4)

%Locates if there is a variation from the NAVAID point, and stores the
%values as direction.

    if sum(isletter(char(WEATHER_OUTPUT(j).POLYGON4(k))))<...
    length(char(WEATHER_OUTPUT(j).POLYGON4(k)))
        distpoint = char(WEATHER_OUTPUT(j).POLYGON4(k));
        direction = char(WEATHER_OUTPUT(j).POLYGON4(k));
        if sum(isspace(char(WEATHER_OUTPUT(j).POLYGON4(k))))>0
            if sum(isletter(distpoint(1:3)))==0;
                distpoint = str2num(distpoint(1:3));
                if sum(isletter(direction(4:6)))==3;
                    direction = direction (4:6);
                elseif sum(isletter(direction(4:5)))==2;
                    direction = direction(4:5);
                else
                    direction = direction(4:4);
                end
            elseif sum(isletter(distpoint(1:2)))==0;
                distpoint = str2num(distpoint(1:2));
                if sum(isletter(direction(3:5)))==3;
                    direction = direction (3:5);
                elseif sum(isletter(direction(3:4)))==2;
                    direction = direction(3:4);
                else
                    direction = direction(3:3);
                end
            else
                distpoint = str2num(distpoint(1:1));
                if sum(isletter(direction(2:4)))==3;
```

```
            direction = direction (2:4);
        elseif sum(isletter(direction(2:3)))==2;
            direction = direction(2:3);
        else
            direction = direction(2:2);
        end
    end
else
    if sum(isletter(distpoint(1:3)))==0;
        distpoint = str2num(distpoint(1:3));
        if sum(isletter(direction(4:length(direction))))==3;
            direction = direction (4:length(direction));
        elseif sum(isletter(direction(4:length(direction))))==2;
            direction = direction(4:length(direction));
        else
            direction = direction(4:length(direction));
        end
    elseif sum(isletter(distpoint(1:2)))==0;
        distpoint = str2num(distpoint(1:2));
        if sum(isletter(direction(3:length(direction))))==3;
            direction = direction (3:length(direction));
        elseif sum(isletter(direction(3:length(direction))))==2;
            direction = direction(3:length(direction));
        else
            direction = direction(3:length(direction));
        end
    else
        distpoint = str2num(distpoint(1:1));
        if sum(isletter(direction(2:length(direction))))==3;
            direction = direction (2:length(direction));
        elseif sum(isletter(direction(2:length(direction))))==2;
            direction = direction(2:length(direction));
        else
            direction = direction(2:length(direction));
        end
    end
end

%Matches Direction to correct numerical value deviation.

if strcmp(direction,'N')==1
    directiondegrees = 0;
elseif strcmp(direction,'S')==1
    directiondegrees = 180;
elseif strcmp(direction,'E')==1
    directiondegrees = 90;
elseif strcmp(direction,'W')==1
    directiondegrees = 270;
elseif strcmp(direction,'NNE')==1
    directiondegrees = 22.5;
elseif strcmp(direction,'NE')==1
    directiondegrees = 45;
elseif strcmp(direction,'ENE')==1
    directiondegrees = 67.5;
elseif strcmp(direction,'ESE')==1
    directiondegrees = 112.5;
elseif strcmp(direction,'SE')==1
    directiondegrees = 135;
elseif strcmp(direction,'SSE')==1
    directiondegrees = 157.5;
elseif strcmp(direction,'SSW')==1
    directiondegrees = 202.5;
```

```
        elseif strcmp(direction,'SW')
           directiondegrees = 225;
        elseif strcmp(direction,'WSW')==1
           directiondegrees = 247.5;
        elseif strcmp(direction,'WNW')==1
           directiondegrees = 292.5;
        elseif strcmp(direction,'NW')==1
           directiondegrees = 315;
        elseif strcmp(direction,'NNW')==1
           directiondegrees = 337.5;
        end
        NAVpoint = char(WEATHER_OUTPUT(j).POLYGON4(k));
        if sum(isspace(NAVpoint))>0
           NAVpoint = NAVpoint(length(NAVpoint)-2:length(NAVpoint));
           breakout = breakout+0;
        else
           NAVpoint = char(WEATHER_OUTPUT(j).POLYGON4(k+1));
           WEATHER_OUTPUT(j).POLYGON4(k)={strcat(char(WEATHER_OUTPUT(j).POLYGON4(k)), ...
           char(WEATHER_OUTPUT(j).POLYGON4(k+1)))};
           WEATHER_OUTPUT(j).POLYGON4(k+1)=[];
           breakout = breakout+1;
        end

%Matches NAVAID indicator to the longitude and latitude values.

        for i = 1:length(NAV_LIST)
           if length(NAV_LIST(i).ID) == length(NAVpoint)
              if NAV_LIST(i).ID == NAVpoint
                 NAVpointLAT= NAV_LIST(i).Latitude;
                 NAVpointLONG= NAV_LIST(i).Longitude;
                 break
              end
           end
        end
        [NEWLAT,NEWLONG] = reckon(NAVpointLAT,NAVpointLONG,nm2deg(distpoint),directiondegrees);
        WEATHER_OUTPUT(j).LAT4(k)= NEWLAT;
        WEATHER_OUTPUT(j).LONG4(k)= NEWLONG;

        if breakout >= 1
           if k <= length(WEATHER_OUTPUT(j).POLYGON4)-1
              k=k+1;
           else
              break;
           end
        end

     else
        for i = 1:length(NAV_LIST)
           if length(NAV_LIST(i).ID) == length(char(WEATHER_OUTPUT(j).POLYGON4(k)))
              if NAV_LIST(i).ID == char(WEATHER_OUTPUT(j).POLYGON4(k))
                 WEATHER_OUTPUT(j).LAT4(k)= NAV_LIST(i).Latitude;
                 WEATHER_OUTPUT(j).LONG4(k)= NAV_LIST(i).Longitude;
                 break
              end
           end
        end

        if breakout >= 1
           if k <= length(WEATHER_OUTPUT(j).POLYGON4)-1
              k=k+1;
           else
              break;
```

```
            end
          end
        end

      end
    end

    breakout = 0;

%For Polygon 5

    if length(WEATHER_OUTPUT(j).POLYGON5)>0
       for k = 1:length(WEATHER_OUTPUT(j).POLYGON5)

%Locates if there is a variation from the NAVAID point, and stores the
%values as direction.

          if sum(isletter(char(WEATHER_OUTPUT(j).POLYGON5(k))))<...
          length(char(WEATHER_OUTPUT(j).POLYGON5(k)))
             distpoint = char(WEATHER_OUTPUT(j).POLYGON5(k));
             direction = char(WEATHER_OUTPUT(j).POLYGON5(k));
             if sum(isspace(char(WEATHER_OUTPUT(j).POLYGON5(k))))>0
                if sum(isletter(distpoint(1:3)))==0;
                   distpoint = str2num(distpoint(1:3));
                   if sum(isletter(direction(4:6)))==3;
                      direction = direction (4:6);
                   elseif sum(isletter(direction(4:5)))==2;
                      direction = direction(4:5);
                   else
                      direction = direction(4:4);
                   end
                elseif sum(isletter(distpoint(1:2)))==0;
                   distpoint = str2num(distpoint(1:2));
                   if sum(isletter(direction(3:5)))==3;
                      direction = direction (3:5);
                   elseif sum(isletter(direction(3:4)))==2;
                      direction = direction(3:4);
                   else
                      direction = direction(3:3);
                   end
                else
                   distpoint = str2num(distpoint(1:1));
                   if sum(isletter(direction(2:4)))==3;
                      direction = direction (2:4);
                   elseif sum(isletter(direction(2:3)))==2;
                      direction = direction(2:3);
                   else
                      direction = direction(2:2);
                   end
                end
             else
                if sum(isletter(distpoint(1:3)))==0;
                   distpoint = str2num(distpoint(1:3));
                   if sum(isletter(direction(4:length(direction))))==3;
                      direction = direction (4:length(direction));
                   elseif sum(isletter(direction(4:length(direction))))==2;
                      direction = direction(4:length(direction));
                   else
                      direction = direction(4:length(direction));
                   end
                elseif sum(isletter(distpoint(1:2)))==0;
                   distpoint = str2num(distpoint(1:2));
```

```
            if sum(isletter(direction(3:length(direction))))==3;
               direction = direction (3:length(direction));
            elseif sum(isletter(direction(3:length(direction))))==2;
               direction = direction(3:length(direction));
            else
               direction = direction(3:length(direction));
            end
         else
            distpoint = str2num(distpoint(1:1));
            if sum(isletter(direction(2:length(direction))))==3;
               direction = direction (2:length(direction));
            elseif sum(isletter(direction(2:length(direction))))==2;
               direction = direction(2:length(direction));
            else
               direction = direction(2:length(direction));
            end
         end
      end

%Matches Direction to correct numerical value deviation.

      if strcmp(direction,'N')==1
         directiondegrees = 0;
      elseif strcmp(direction,'S')==1
         directiondegrees = 180;
      elseif strcmp(direction,'E')==1
         directiondegrees = 90;
      elseif strcmp(direction,'W')==1
         directiondegrees = 270;
      elseif strcmp(direction,'NNE')==1
         directiondegrees = 22.5;
      elseif strcmp(direction,'NE')==1
         directiondegrees = 45;
      elseif strcmp(direction,'ENE')==1
         directiondegrees = 67.5;
      elseif strcmp(direction,'ESE')==1
         directiondegrees = 112.5;
      elseif strcmp(direction,'SE')==1
         directiondegrees = 135;
      elseif strcmp(direction,'SSE')==1
         directiondegrees = 157.5;
      elseif strcmp(direction,'SSW')==1
         directiondegrees = 202.5;
      elseif strcmp(direction,'SW')
         directiondegrees = 225;
      elseif strcmp(direction,'WSW')==1
         directiondegrees = 247.5;
      elseif strcmp(direction,'WNW')==1
         directiondegrees = 292.5;
      elseif strcmp(direction,'NW')==1
         directiondegrees = 315;
      elseif strcmp(direction,'NNW')==1
         directiondegrees = 337.5;
      end
      NAVpoint = char(WEATHER_OUTPUT(j).POLYGON5(k));
      if sum(isspace(NAVpoint))>0
         NAVpoint = NAVpoint(length(NAVpoint)-2:length(NAVpoint));
         breakout = breakout+0;
      else
         NAVpoint = char(WEATHER_OUTPUT(j).POLYGON5(k+1));
         WEATHER_OUTPUT(j).POLYGON5(k)={strcat(char(WEATHER_OUTPUT(j).POLYGON5(k)), ...
         char(WEATHER_OUTPUT(j).POLYGON5(k+1)))};
```

```
            WEATHER_OUTPUT(j).POLYGON5(k+1)=[];
            breakout = breakout+1;
        end

  %Matches NAVAID indicator to the longitude and latitude values.

        for i = 1:length(NAV_LIST)
            if length(NAV_LIST(i).ID) == length(NAVpoint)
                if NAV_LIST(i).ID == NAVpoint
                    NAVpointLAT= NAV_LIST(i).Latitude;
                    NAVpointLONG= NAV_LIST(i).Longitude;
                    break
                end
            end
        end
        [NEWLAT,NEWLONG] = reckon(NAVpointLAT,NAVpointLONG,nm2deg(distpoint),directiondegrees);
        WEATHER_OUTPUT(j).LAT5(k)= NEWLAT;
        WEATHER_OUTPUT(j).LONG5(k)= NEWLONG;

        if breakout >= 1
            if k <= length(WEATHER_OUTPUT(j).POLYGON5)-1
                k=k+1;
            else
                break;
            end
        end

      else
        for i = 1:length(NAV_LIST)
            if length(NAV_LIST(i).ID) == length(char(WEATHER_OUTPUT(j).POLYGON5(k)))
                if NAV_LIST(i).ID == char(WEATHER_OUTPUT(j).POLYGON5(k))
                    WEATHER_OUTPUT(j).LAT5(k)= NAV_LIST(i).Latitude;
                    WEATHER_OUTPUT(j).LONG5(k)= NAV_LIST(i).Longitude;
                    break
                end
            end
        end

        if breakout >= 1
            if k <= length(WEATHER_OUTPUT(j).POLYGON5)-1
                k=k+1;
            else
                break;
            end
        end
      end

    end
  end

end

save WEATHER_OUTPUT WEATHER_OUTPUT
```

APPENDIX F: Run Flight Generation (Plotting Program)

```
% Runflightgeneration

% Programmed by: Melinda Gates and Hojong Baik (07/2004)
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%This program plotted the AIRMETS and flight trajectory paths in 3-D.
%-------------------------------------------------------------------------

clf
figure('Name', 'WeatherOutput');
load WEATHER_OUTPUT
set(gcf, 'doublebuffer', 'on');

%Plot flight paths.

flightPathGeneration_SATS('BOS', 'CLE', 1)
hold on;
flightPathGeneration_SATS('BOS', 'IAD', 1)
flightPathGeneration_SATS('IAD', 'CLE', 1)

%Store data for Plotting.

sizeOfPoints = size(WEATHER_OUTPUT(1).LONG1, 2);
if (sizeOfPoints > 1)
   maxAlt = ones(1, sizeOfPoints) * str2num(char(WEATHER_OUTPUT(1).ALTITUDEMAX1));
   minAlt = ones(1, sizeOfPoints) * str2num(char(WEATHER_OUTPUT(1).ALTITUDEMIN1));
   long   = WEATHER_OUTPUT(1).LONG1;
   lat    = WEATHER_OUTPUT(1).LAT1;

   h1_top    = plot3(long, lat, maxAlt, 'b', 'linewidth', 2);
   h1_bottom = plot3(long, lat, minAlt, 'b', 'linewidth', 2);
   for j = 1:20
      h_stems(j) = plot3([0 0], [0 0], [0 0], 'b', 'linewidth', 2);
   end

   for j = 1:sizeOfPoints-1
      h_stems(j) = plot3([long(j) long(j)], [lat(j) lat(j)], [minAlt(j) maxAlt(j)], 'b', 'linewidth', 2);
   end
drawnow
end;
sizeOfPoints2 = size(WEATHER_OUTPUT(15).LONG2, 2);
if (sizeOfPoints2 > 1)
   maxAlt2 = ones(1, sizeOfPoints2) * str2num(char(WEATHER_OUTPUT(15).ALTITUDEMAX2));
   minAlt2 = ones(1, sizeOfPoints2) * str2num(char(WEATHER_OUTPUT(15).ALTITUDEMIN2));
   long2   = WEATHER_OUTPUT(15).LONG2;
   lat2    = WEATHER_OUTPUT(15).LAT2;

   h2_top    = plot3(long2, lat2, maxAlt2, 'm', 'linewidth', 2);
   h2_bottom = plot3(long2, lat2, minAlt2, 'm', 'linewidth', 2);

   for j = 1:20
      h_stems2(j) = plot3([0 0], [0 0], [0 0], 'm', 'linewidth', 2);
   end

   for j = 1:sizeOfPoints2-1
      h_stems2(j) = plot3([long2(j) long2(j)], [lat2(j) lat2(j)], [minAlt2(j) maxAlt2(j)], 'm', 'linewidth', 2);
   end
  drawnow
end;
```

```
sizeOfPoints3 = size(WEATHER_OUTPUT(564).LONG3, 2);
if (sizeOfPoints3 > 1)
   maxAlt3 = ones(1, sizeOfPoints3) * str2num(char(WEATHER_OUTPUT(564).ALTITUDEMAX3));
   minAlt3 = ones(1, sizeOfPoints3) * str2num(char(WEATHER_OUTPUT(564).ALTITUDEMIN3));
   long3   = WEATHER_OUTPUT(564).LONG3;
   lat3    = WEATHER_OUTPUT(564).LAT3;

   h3_top    = plot3(long3, lat3, maxAlt3, 'c', 'linewidth', 2);
   h3_bottom = plot3(long3, lat3, minAlt3, 'c', 'linewidth', 2);

   for j = 1:20
      h_stems3(j) = plot3([0 0], [0 0], [0 0], 'c', 'linewidth', 2);
   end

   for j = 1:sizeOfPoints3-1
      h_stems3(j) = plot3([long3(j) long3(j)], [lat3(j) lat3(j)], [minAlt3(j) maxAlt3(j)], 'c', 'linewidth', 2);
   end
   drawnow
end;

sizeOfPoints4 = size(WEATHER_OUTPUT(604).LONG4, 2);
if (sizeOfPoints4 > 1)
   maxAlt4 = ones(1, sizeOfPoints4) * str2num(char(WEATHER_OUTPUT(604).ALTITUDEMAX4));
   minAlt4 = ones(1, sizeOfPoints4) * str2num(char(WEATHER_OUTPUT(604).ALTITUDEMIN4));
   long4   = WEATHER_OUTPUT(604).LONG4;
   lat4    = WEATHER_OUTPUT(604).LAT4;

   h4_top    = plot3(long4, lat4, maxAlt4, 'k', 'linewidth', 2);
   h4_bottom = plot3(long4, lat4, minAlt4, 'k', 'linewidth', 2);

   for j = 1:20
      h_stems4(j) = plot3([0 0], [0 0], [0 0], 'k', 'linewidth', 2);
   end

   for j = 1:sizeOfPoints4-1
      h_stems4(j) = plot3([long4(j) long4(j)], [lat4(j) lat4(j)], [minAlt4(j) maxAlt4(j)], 'k', 'linewidth', 2);
   end
   drawnow
end;

sizeOfPoints5 = size(WEATHER_OUTPUT(3029).LONG5, 2);
if (sizeOfPoints5 > 1)
   maxAlt5 = ones(1, sizeOfPoints5) * str2num(char(WEATHER_OUTPUT(3029).ALTITUDEMAX5));
   minAlt5 = ones(1, sizeOfPoints5) * str2num(char(WEATHER_OUTPUT(3029).ALTITUDEMIN5));
   long5   = WEATHER_OUTPUT(3029).LONG5;
   lat5    = WEATHER_OUTPUT(3029).LAT5;

   h5_top    = plot3(long5, lat5, maxAlt5, 'y', 'linewidth', 2);
   h5_bottom = plot3(long5, lat5, minAlt5, 'y', 'linewidth', 2);

   for j = 1:20
      h_stems5(j) = plot3([0 0], [0 0], [0 0], 'y', 'linewidth', 2);
   end

   for j = 1:sizeOfPoints5-1
      h_stems5(j) = plot3([long5(j) long5(j)], [lat5(j) lat5(j)], [minAlt5(j) maxAlt5(j)], 'y', 'linewidth', 2);
   end
   drawnow
end;

for j = 1:sizeOfPoints-1
```

```
      set(h_stems(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'b', 'MarkerFaceColor',
'b','linewidth', 2);
   end;
   for j = 1:sizeOfPoints2-1
      set(h_stems2(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'm', 'MarkerFaceColor',
'm','linewidth', 2);
   end;
   for j = 1:sizeOfPoints3-1
      set(h_stems3(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'c', 'MarkerFaceColor',
'c','linewidth', 2);
   end;
   for j = 1:sizeOfPoints4-1
      set(h_stems4(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'k', 'MarkerFaceColor',
'k','linewidth', 2);
   end;
   for j = 1:sizeOfPoints5-1
      set(h_stems5(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'y', 'MarkerFaceColor',
'y','linewidth', 2);
   end;
   drawnow

for i =  2:length(WEATHER_OUTPUT)
   load usalo
   sizeOfPoints = size(WEATHER_OUTPUT(i).LONG1, 2);

   maxAlt = ones(1, sizeOfPoints) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMAX1));
   minAlt = ones(1, sizeOfPoints) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMIN1));
   long   = WEATHER_OUTPUT(i).LONG1;
   lat    = WEATHER_OUTPUT(i).LAT1;

   set(h1_top, 'Xdata', long, 'Ydata', lat,  'Zdata', maxAlt, 'linewidth', 2);
   set(h1_bottom, 'Xdata', long, 'Ydata', lat,  'Zdata', minAlt, 'linewidth', 2);
   for j = 1:sizeOfPoints-1
      set(h_stems(j), 'Xdata', [long(j) long(j)], 'Ydata', [lat(j) lat(j)], 'Zdata', [minAlt(j) maxAlt(j)],...
      'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b','linewidth', 2);
   end
   drawnow

   sizeOfPoints2 = size(WEATHER_OUTPUT(i).LONG2, 2);

   maxAlt2 = ones(1, sizeOfPoints2) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMAX2));
   minAlt2 = ones(1, sizeOfPoints2) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMIN2));
   long2   = WEATHER_OUTPUT(i).LONG2;
   lat2    = WEATHER_OUTPUT(i).LAT2;

   set(h2_top, 'Xdata', long2, 'Ydata', lat2,  'Zdata', maxAlt2, 'linewidth', 2);
   set(h2_bottom, 'Xdata', long2, 'Ydata', lat2,  'Zdata', minAlt2, 'linewidth', 2);
   for j = 1:sizeOfPoints2-1
      set(h_stems2(j), 'Xdata', [long2(j) long2(j)], 'Ydata', [lat2(j) lat2(j)], 'Zdata', [minAlt2(j) maxAlt2(j)],...
      'MarkerEdgeColor', 'm', 'MarkerFaceColor', 'm','linewidth', 2);
   end
   drawnow


   sizeOfPoints3 = size(WEATHER_OUTPUT(i).LONG3, 2);

   maxAlt3 = ones(1, sizeOfPoints3) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMAX3));
   minAlt3 = ones(1, sizeOfPoints3) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMIN3));
   long3   = WEATHER_OUTPUT(i).LONG3;
   lat3    = WEATHER_OUTPUT(i).LAT3;

   set(h3_top, 'Xdata', long3, 'Ydata', lat3,  'Zdata', maxAlt3, 'linewidth', 2);
```

```
   set(h3_bottom, 'Xdata', long3, 'Ydata', lat3,  'Zdata', minAlt3, 'linewidth', 2);
   for j = 1:sizeOfPoints3-1
      set(h_stems3(j), 'Xdata', [long3(j) long3(j)], 'Ydata', [lat3(j) lat3(j)], 'Zdata', [minAlt3(j) maxAlt3(j)],...
      'MarkerEdgeColor', 'c', 'MarkerFaceColor', 'c','linewidth', 2);
   end
   drawnow



   sizeOfPoints4 = size(WEATHER_OUTPUT(i).LONG4, 2);

   maxAlt4 = ones(1, sizeOfPoints4) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMAX4));
   minAlt4 = ones(1, sizeOfPoints4) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMIN4));
   long4   = WEATHER_OUTPUT(i).LONG4;
   lat4    = WEATHER_OUTPUT(i).LAT4;

   set(h4_top, 'Xdata', long4, 'Ydata', lat4,  'Zdata', maxAlt4, 'linewidth', 2);
   set(h4_bottom, 'Xdata', long4, 'Ydata', lat4,  'Zdata', minAlt4, 'linewidth', 2);
   for j = 1:sizeOfPoints4-1
      set(h_stems4(j), 'Xdata', [long4(j) long4(j)], 'Ydata', [lat4(j) lat4(j)], 'Zdata', [minAlt4(j) maxAlt4(j)],...
      'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k','linewidth', 2);
   end
   drawnow



   sizeOfPoints5 = size(WEATHER_OUTPUT(i).LONG5, 2);

   maxAlt5 = ones(1, sizeOfPoints5) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMAX5));
   minAlt5 = ones(1, sizeOfPoints5) * str2num(char(WEATHER_OUTPUT(i).ALTITUDEMIN5));
   long5   = WEATHER_OUTPUT(i).LONG5;
   lat5    = WEATHER_OUTPUT(i).LAT5;

   set(h5_top, 'Xdata', long5, 'Ydata', lat5,  'Zdata', maxAlt5, 'linewidth', 2);
   set(h5_bottom, 'Xdata', long5, 'Ydata', lat5,  'Zdata', minAlt5, 'linewidth', 2);
   for j = 1:sizeOfPoints5-1
      set(h_stems5(j), 'Xdata', [long5(j) long5(j)], 'Ydata', [lat5(j) lat5(j)], 'Zdata', [minAlt5(j) maxAlt5(j)],...
      'MarkerEdgeColor', 'y', 'MarkerFaceColor', 'y','linewidth', 2);
   end
   drawnow

   weathermovie(i) =  getframe;

   %Set Plotting Parameters.

   set(h1_top, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h1_bottom, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h2_top, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h2_bottom, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h3_top, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h3_bottom, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h4_top, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h4_bottom, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h5_top, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);
   set(h5_bottom, 'Xdata', 0, 'Ydata', 0,  'Zdata', 0, 'linewidth', 2);

   %Plot Data.

   for j = 1:sizeOfPoints-1
      set(h_stems(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'b', 'MarkerFaceColor',
'b','linewidth', 2);
   end;
```

```
   for j = 1:sizeOfPoints2-1
      set(h_stems2(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'm', 'MarkerFaceColor',
'm','linewidth', 2);
   end;
   for j = 1:sizeOfPoints3-1
      set(h_stems3(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'c', 'MarkerFaceColor',
'c','linewidth', 2);
   end;
   for j = 1:sizeOfPoints4-1
      set(h_stems4(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'k', 'MarkerFaceColor',
'k','linewidth', 2);
   end;
   for j = 1:sizeOfPoints5-1
      set(h_stems5(j), 'Xdata', [0 0], 'Ydata', [0 0], 'Zdata', [0 0], 'MarkerEdgeColor', 'y', 'MarkerFaceColor',
'y','linewidth', 2);
   end;
   drawnow


end;
```

APPENDIX G: BOSCLE Intersections 12,000 Feet

```
% BOSCLEIntersections12,000

% Programmed by: Melinda Gates and Hojong Baik (07/2004)
%-------------------------------------------------------------------------

clc
clear all;
IntersectionsBOSCLE2 = fopen('IntersectionsBOSCLE2.txt','w');

info = flightPathGeneration_SATS('bos', 'cle', 1);
load WPtestBOSCLE
load WEATHER_OUTPUT
load usalo
load BOSCLEalt

totFlightTrajectories = 0;
totIcingAirmets = 0;
IntersectLLA = 0;

totFlightTrajectories = size(totFlightTrajectories, 1);
totIcingAirmets = size(WEATHER_OUTPUT, 2);

for i = 1:totFlightTrajectories
   for j = 1:totIcingAirmets

% Check for Intersections on First Polygon
%-------------------------------------------------------------------------------------------------------------------------------
      if(~isempty(WEATHER_OUTPUT(j).LONG1))
         IntersectLLA=0;
         x1 = WEATHER_OUTPUT(j).LONG1;
         y1 = WEATHER_OUTPUT(j).LAT1;
         in = inpolygon(xvBOSCLE,yvBOSCLE,x1,y1);
         indices4PassingWaypoints1 = find(in == 1);
         if ~isempty(indices4PassingWaypoints1)
            maxalt1 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMAX1));
            minalt1 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMIN1));
            for i=1:30
               inwith1 = in(i);
               BOSCLEaltitude = BOSCLEalt(i);
               if inwith1==1
                  if BOSCLEaltitude <= maxalt1 & BOSCLEaltitude >= minalt1
                     IntersectLLA = IntersectLLA+1;
                     break;
                  end;
               end;
            end;
            if IntersectLLA > 0;
               fprintf(IntersectionsBOSCLE2,'%g\n',j);
            end;
         end;
      end;

% Check for Intersections on Second Polygon
%-------------------------------------------------------------------------------------------------------------------------------
```

```
    if(~isempty(WEATHER_OUTPUT(j).LONG2))
        IntersectLLA=0;
        x2 = WEATHER_OUTPUT(j).LONG2;
        y2 = WEATHER_OUTPUT(j).LAT2;
        in = inpolygon(xvBOSCLE,yvBOSCLE,x2,y2);
        indices4PassingWaypoints2 = find(in == 1);
        if ~isempty(indices4PassingWaypoints2)
           maxalt2 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMAX2));
           minalt2 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMIN2));
           for i=1:30
              inwith1 = in(i);
              BOSCLEaltitude = BOSCLEalt(i);
              if inwith1==1
                 if BOSCLEaltitude <= maxalt2 & BOSCLEaltitude >= minalt2
                    IntersectLLA = IntersectLLA+1;
                    break;
                 end;
              end;
           end;
           if IntersectLLA > 0;
              fprintf(IntersectionsBOSCLE2,'%g\n',j);
           end;
        end;
    end;

% Check for Intersections on Third Polygon
%-------------------------------------------------------------------------------------------------------------------------------
    if(~isempty(WEATHER_OUTPUT(j).LONG3))
        IntersectLLA=0;
        x3 = WEATHER_OUTPUT(j).LONG3;
        y3 = WEATHER_OUTPUT(j).LAT3;
        in = inpolygon(xvBOSCLE,yvBOSCLE,x3,y3);
        indices4PassingWaypoints3 = find(in == 1);
        if ~isempty(indices4PassingWaypoints3)
           maxalt3 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMAX3));
           minalt3 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMIN3));
           for i=1:30
              inwith1 = in(i);
              BOSCLEaltitude = BOSCLEalt(i);
              if inwith1==1
                 if BOSCLEaltitude <= maxalt3 & BOSCLEaltitude >= minalt3
                    IntersectLLA = IntersectLLA+1;
                    break;
                 end;
              end;
           end;
           if IntersectLLA > 0;
              fprintf(IntersectionsBOSCLE2,'%g\n',j);
           end;
        end;
    end;

% Check for Intersections on Fourth Polygon
%-------------------------------------------------------------------------------------------------------------------------------
    if(~isempty(WEATHER_OUTPUT(j).LONG4))
        IntersectLLA=0;
```

```
            x4 = WEATHER_OUTPUT(j).LONG4;
            y4 = WEATHER_OUTPUT(j).LAT4;
            in = inpolygon(xvBOSCLE,yvBOSCLE,x4,y4);
            indices4PassingWaypoints4 = find(in == 1);
            if ~isempty(indices4PassingWaypoints4)
               maxalt4 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMAX4));
               minalt4 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMIN4));
               for i=1:30
                  inwith1 = in(i);
                  BOSCLEaltitude = BOSCLEalt(i);
                  if inwith1==1
                     if BOSCLEaltitude <= maxalt4 & BOSCLEaltitude >= minalt4
                        IntersectLLA = IntersectLLA+1;
                        break;
                     end;
                  end;
               end;
               if IntersectLLA > 0;
                  fprintf(IntersectionsBOSCLE2,'%g\n',j);
               end;
            end;
         end;

% Check for Intersections on Fifth Polygon
%-------------------------------------------------------------------------------------------------------------------------------------------
         if(~isempty(WEATHER_OUTPUT(j).LONG5))
            IntersectLLA=0;
            x5 = WEATHER_OUTPUT(j).LONG5;
            y5 = WEATHER_OUTPUT(j).LAT5;

            in = inpolygon(xvBOSCLE,yvBOSCLE,x5,y5);
            indices4PassingWaypoints5 = find(in == 1);
            if ~isempty(indices4PassingWaypoints5)
               maxalt5 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMAX5));
               minalt5 =  str2num(char(WEATHER_OUTPUT(j).ALTITUDEMIN5));
               for i=1:30
                  inwith1 = in(i);
                  BOSCLEaltitude = BOSCLEalt(i);
                  if inwith1==1
                     if BOSCLEaltitude <= maxalt5 & BOSCLEaltitude >= minalt5
                        IntersectLLA = IntersectLLA+1;
                        break;
                     end;
                  end;
               end;
               if IntersectLLA > 0;
                  fprintf(IntersectionsBOSCLE2,'%g\n',j);
               end;
            end;
         end;

      end;
   end;

fclose(IntersectionsBOSCLE2);
```

APPENDIX H: BOSCLE Intersection Data Reduction-12,000 Feet

```
% BOSCLEIntersectionDataReduction12,000

% Programmed by: Melinda Gates (08/2004)
%--------------------------------------------------------------------------

clear all;

% Eliminate When Single AIRMET is Listed Multiple Times
%-----------------------------------------------------------------------------------------------------------------------------

BOSCLEIntersections = fopen('IntersectionsBOSCLE2.txt','rt');
NoRepeatIntersectionBOSCLE = fopen('NumberofnoRepeatIntersectionsBOSCLE.txt','w');


   line = fgetl(BOSCLEIntersections);
   line2 = fgetl(BOSCLEIntersections);
   if strmatch(line,line2)
      fprintf(NoRepeatIntersectionBOSCLE, '%s\n', line);
   else
      fprintf(NoRepeatIntersectionBOSCLE, '%s\n', line);
      fprintf(NoRepeatIntersectionBOSCLE, '%s\n', line2);
   end;
while (not(feof(BOSCLEIntersections)))
   line = line2;
   line2 = fgetl(BOSCLEIntersections);
   if isempty (strmatch(line,line2))
      fprintf(NoRepeatIntersectionBOSCLE, '%s\n', line2);
   end;
end;

fclose(BOSCLEIntersections);
fclose(NoRepeatIntersectionBOSCLE);

% Eliminate Redundant AIRMETs (Same Information)
%-----------------------------------------------------------------------------------------------------------------------------

NoRepeatIntersectionsBOSCLE = fopen('NumberofnoRepeatIntersectionsBOSCLE.txt', 'rt');
STandETBOSCLE = fopen('StarttimeandEndtimeNoRepeatBOSCLE.txt', 'w');
COMPAREnrAIRMETs = fopen('CompareNRAirmetsBOSCLE.txt', 'w');

load WEATHER_OUTPUT;

AIRMETnumber = fgetl(NoRepeatIntersectionsBOSCLE);
AIRMETstr2num = str2num(AIRMETnumber);
   fprintf(COMPAREnrAIRMETs, '%s', AIRMETnumber);
   fprintf(STandETBOSCLE, '%s\n', AIRMETnumber);
   fprintf(STandETBOSCLE, '%s\n', WEATHER_OUTPUT(AIRMETstr2num).STARTTIME);
   fprintf(STandETBOSCLE, '%s\n', WEATHER_OUTPUT(AIRMETstr2num).ENDTIME);

while (not(feof(NoRepeatIntersectionsBOSCLE)))
   AIRMETnumber = fgetl(NoRepeatIntersectionsBOSCLE);
   AIRMETstr2num = str2num(AIRMETnumber);
  if strmatch(WEATHER_OUTPUT(AIRMETstr2num).STARTTIME, WEATHER_OUTPUT(AIRMETstr2num-1).STARTTIME, 'exact') &…
  strmatch(WEATHER_OUTPUT(AIRMETstr2num).ENDTIME, WEATHER_OUTPUT(AIRMETstr2num-1).ENDTIME, 'exact')
      fprintf(STandETBOSCLE, '%s', ' ');
   else
      fprintf(COMPAREnrAIRMETs, '\n%s', AIRMETnumber);
      fprintf(STandETBOSCLE, '\n%s\n', AIRMETnumber);
      fprintf(STandETBOSCLE, '%s\n', WEATHER_OUTPUT(AIRMETstr2num).STARTTIME);
      fprintf(STandETBOSCLE, '%s\n', WEATHER_OUTPUT(AIRMETstr2num).ENDTIME);
```

```
   end;
end;

fclose(NoRepeatIntersectionsBOSCLE);
fclose(STandETBOSCLE);
fclose(COMPAREnrAIRMETs);

% Sum Number of Intersecting AIRMETs Each Month
%-------------------------------------------------------------------------------------------------------------------------

CountIntersectionsReadBOSCLE = fopen('StarttimeandEndtimeNoRepeatBOSCLE.txt', 'rt');
CountIntersectionsWriteBOSCLE = fopen('NumberofIntersectionsBOSCLE.txt','w');

NOV2001 = 0;
DEC2001 = 0;
JAN2002 = 0;
FEB2002 = 0;
MAR2002 = 0;
APR2002 = 0;
MAY2002 = 0;
NOV2002 = 0;
DEC2002 = 0;
JAN2003 = 0;
FEB2003 = 0;
MAR2003 = 0;
APR2003 = 0;
MAY2003 = 0;

while (not(feof(CountIntersectionsReadBOSCLE)))
   line = fgetl(CountIntersectionsReadBOSCLE);
   if str2num(line) <= 361
      NOV2001 = NOV2001 + 1;
   elseif str2num(line) <= 631
      DEC2001 = DEC2001 + 1;
   elseif str2num(line) <= 928
      JAN2002 = JAN2002 + 1;
   elseif str2num(line) <= 1172
      FEB2002 = FEB2002 + 1;
   elseif str2num(line) <= 1452
      MAR2002 = MAR2002 + 1;
   elseif str2num(line) <= 1722
      APR2002 = APR2002 + 1;
   elseif str2num(line) <= 2002
      MAY2002 = MAY2002 + 1;
   elseif str2num(line) <= 2250
      NOV2002 = NOV2002 + 1;
   elseif str2num(line) <= 2512
      DEC2002 = DEC2002 + 1;
   elseif str2num(line) <= 2746
      JAN2003 = JAN2003 + 1;
   elseif str2num(line) <= 2873
      FEB2003 = FEB2003 + 1;
   elseif str2num(line) <= 3007
      MAR2003 = MAR2003 + 1;
   elseif str2num(line) <= 3146
      APR2003 = APR2003 + 1;
   elseif str2num(line) <= 3280
      MAY2003 = MAY2003 + 1;
   end;
   line = fgetl(CountIntersectionsReadBOSCLE);
   line = fgetl(CountIntersectionsReadBOSCLE);
   line = fgetl(CountIntersectionsReadBOSCLE);
```

```
end;

fprintf(CountIntersectionsWriteBOSCLE,'%s\n','NOVEMBER 2001');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',NOV2001);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','DECEMBER 2001');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',DEC2001);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','JANUARY 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',JAN2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','FEBRUARY 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',FEB2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','MARCH 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',MAR2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','APRIL 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',APR2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','MAY 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',MAY2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','NOVEMBER 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',NOV2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','DECEMBER 2002');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',DEC2002);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','JANUARY 2003');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',JAN2003);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','FEBRUARY 2003');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',FEB2003);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','MARCH 2003');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',MAR2003);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','APRIL 2003');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',APR2003);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','MAY 2003');
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',MAY2003);
fprintf(CountIntersectionsWriteBOSCLE,'%s\n','TOTAL');
Total = NOV2001+ DEC2001+ JAN2002+ FEB2002+ MAR2002+ APR2002+ MAY2002+ NOV2002+ DEC2002+
JAN2003+ FEB2003+ MAR2003+ APR2003+ MAY2003;
fprintf(CountIntersectionsWriteBOSCLE,'%g\n',Total);

fclose(CountIntersectionsReadBOSCLE);
fclose(CountIntersectionsWriteBOSCLE);
```

APPENDIX I: HourPlacementBOSCLE-November2001-12,000Feet-2Weeks

```
% HourPlacement1BOSCLE2Weeks12,000

% Programmed by: Melinda Gates (08/2004)
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%This program places AIRMET in time periods so the dispatch reliability
%can be divided into morning, afternoon, and evening periods.
%--------------------------------------------------------------------------

clear all;

%Set all possible time periods.

StartEnd = fopen('DataWithoutAMDboscle.txt', 'rt');
TimePeriods(1) = fopen('TimePeriodsBOSCLE1.txt', 'w');
TimePeriods(2) = fopen('TimePeriodsBOSCLE2.txt', 'w');
TimePeriods(3) = fopen('TimePeriodsBOSCLE3.txt', 'w');
TimePeriods(4) = fopen('TimePeriodsBOSCLE4.txt', 'w');
TimePeriods(5) = fopen('TimePeriodsBOSCLE5.txt', 'w');
TimePeriods(6) = fopen('TimePeriodsBOSCLE6.txt', 'w');
TimePeriods(7) = fopen('TimePeriodsBOSCLE7.txt', 'w');
TimePeriods(8) = fopen('TimePeriodsBOSCLE8.txt', 'w');
TimePeriods(9) = fopen('TimePeriodsBOSCLE9.txt', 'w');
TimePeriods(10) = fopen('TimePeriodsBOSCLE10.txt', 'w');
TimePeriods(11) = fopen('TimePeriodsBOSCLE11.txt', 'w');
TimePeriods(12) = fopen('TimePeriodsBOSCLE12.txt', 'w');
TimePeriods(13) = fopen('TimePeriodsBOSCLE13.txt', 'w');
TimePeriods(14) = fopen('TimePeriodsBOSCLE14.txt', 'w');
TimePeriods(15) = fopen('TimePeriodsBOSCLE15.txt', 'w');
TimePeriods(16) = fopen('TimePeriodsBOSCLE16.txt', 'w');
TimePeriods(17) = fopen('TimePeriodsBOSCLE17.txt', 'w');
TimePeriods(18) = fopen('TimePeriodsBOSCLE18.txt', 'w');
TimePeriods(19) = fopen('TimePeriodsBOSCLE19.txt', 'w');
TimePeriods(20) = fopen('TimePeriodsBOSCLE20.txt', 'w');
TimePeriods(21) = fopen('TimePeriodsBOSCLE21.txt', 'w');
TimePeriods(22) = fopen('TimePeriodsBOSCLE22.txt', 'w');
TimePeriods(23) = fopen('TimePeriodsBOSCLE23.txt', 'w');
TimePeriods(24) = fopen('TimePeriodsBOSCLE24.txt', 'w');
TimePeriods(25) = fopen('TimePeriodsBOSCLE25.txt', 'w');
TimePeriods(26) = fopen('TimePeriodsBOSCLE26.txt', 'w');
TimePeriods(27) = fopen('TimePeriodsBOSCLE27.txt', 'w');
TimePeriods(28) = fopen('TimePeriodsBOSCLE28.txt', 'w');
TimePeriods(29) = fopen('TimePeriodsBOSCLE29.txt', 'w');
TimePeriods(30) = fopen('TimePeriodsBOSCLE30.txt', 'w');
TimePeriods(31) = fopen('TimePeriodsBOSCLE31.txt', 'w');

%Initialize all values.

for i=1:31
   Hour2300to0600(i)=0; Hour0000to0600(i)=0; Hour0000to0700(i)=0; Hour0100to0600(i)=0; Hour0100to0700(i)=0;
Hour0100to0800(i)=0; Hour0200to0600(i)=0;
   Hour0200to0700(i)=0; Hour0200to0800(i)=0; Hour0200to0900(i)=0; Hour0300to0600(i)=0; Hour0300to0700(i)=0;
Hour0300to0800(i)=0;
   Hour0300to0900(i)=0; Hour0300to1000(i)=0; Hour0400to0600(i)=0; Hour0400to0700(i)=0; Hour0400to0800(i)=0;
Hour0400to0900(i)=0;
   Hour0400to1000(i)=0; Hour0400to1100(i)=0; Hour0500to0600(i)=0; Hour0500to0700(i)=0; Hour0500to0800(i)=0;
Hour0500to0900(i)=0;
   Hour0500to1000(i)=0; Hour0500to1100(i)=0; Hour0500to1200(i)=0; Hour0600to0700(i)=0; Hour0600to0800(i)=0;
Hour0600to0900(i)=0;
```

```
   Hour0600to1000(i)=0; Hour0600to1100(i)=0; Hour0600to1200(i)=0; Hour0600to1300(i)=0; Hour0700to0800(i)=0;
Hour0700to0900(i)=0;
   Hour0700to1000(i)=0; Hour0700to1100(i)=0; Hour0700to1200(i)=0; Hour0700to1300(i)=0; Hour0700to1400(i)=0;
Hour0800to0900(i)=0;
   Hour0800to1000(i)=0; Hour0800to1100(i)=0; Hour0800to1200(i)=0; Hour0800to1300(i)=0; Hour0800to1400(i)=0;
Hour0800to1500(i)=0;
   Hour0900to1000(i)=0; Hour0900to1100(i)=0; Hour0900to1200(i)=0; Hour0900to1300(i)=0; Hour0900to1400(i)=0;
Hour0900to1500(i)=0;
   Hour0900to1600(i)=0; Hour1000to1100(i)=0; Hour1000to1200(i)=0; Hour1000to1300(i)=0; Hour1000to1400(i)=0;
Hour1000to1500(i)=0;
   Hour1000to1600(i)=0; Hour1000to1700(i)=0; Hour1100to1200(i)=0; Hour1100to1300(i)=0; Hour1100to1400(i)=0;
Hour1100to1500(i)=0;
   Hour1100to1600(i)=0; Hour1100to1700(i)=0; Hour1100to1800(i)=0; Hour1200to1300(i)=0; Hour1200to1400(i)=0;
Hour1200to1500(i)=0;
   Hour1200to1600(i)=0; Hour1200to1700(i)=0; Hour1200to1800(i)=0; Hour1200to1900(i)=0; Hour1300to1400(i)=0;
Hour1300to1500(i)=0;
   Hour1300to1600(i)=0; Hour1300to1700(i)=0; Hour1300to1800(i)=0; Hour1300to1900(i)=0; Hour1300to2000(i)=0;
Hour1400to1500(i)=0;
   Hour1400to1600(i)=0; Hour1400to1700(i)=0; Hour1400to1800(i)=0; Hour1400to1900(i)=0; Hour1400to2000(i)=0;
Hour1400to2100(i)=0;
   Hour1500to1600(i)=0; Hour1500to1700(i)=0; Hour1500to1800(i)=0; Hour1500to1900(i)=0; Hour1500to2000(i)=0;
Hour1500to2100(i)=0;
   Hour1500to2200(i)=0; Hour1600to1700(i)=0; Hour1600to1800(i)=0; Hour1600to1900(i)=0; Hour1600to2000(i)=0;
Hour1600to2100(i)=0;
   Hour1600to2200(i)=0; Hour1600to2300(i)=0; Hour1700to1800(i)=0; Hour1700to1900(i)=0; Hour1700to2000(i)=0;
Hour1700to2100(i)=0;
   Hour1700to2200(i)=0; Hour1700to2300(i)=0; Hour1700to0000(i)=0; Hour1800to1900(i)=0; Hour1800to2000(i)=0;
Hour1800to2100(i)=0;
   Hour1800to2200(i)=0; Hour1800to2300(i)=0; Hour1800to0000(i)=0; Hour1800to0100(i)=0; Hour1900to2000(i)=0;
Hour1900to2100(i)=0;
   Hour1900to2200(i)=0; Hour1900to2300(i)=0; Hour1900to0000(i)=0; Hour1900to0100(i)=0; Hour1900to0200(i)=0;
Hour2000to2100(i)=0;
   Hour2000to2200(i)=0; Hour2000to2300(i)=0; Hour2000to0000(i)=0; Hour2000to0100(i)=0; Hour2000to0200(i)=0;
Hour2000to0300(i)=0;
end;

while(not(feof(StartEnd)))
   line = fgetl(StartEnd);
   AirmetNum = str2num(line);
   line2 = fgetl(StartEnd);
   date = str2num(line2(1:2));
   numline = str2num(line2(3:4));
   line3 = fgetl(StartEnd);
   endtime = str2num(line3(3:4));

%For the first two weeks of November 2001 place AIRMETs into
%categies by the day they occur.

   if (AirmetNum >=1 & AirmetNum<= 361)
      if date ==1
         category = 1;
      elseif date ==2
         category = 2;
      elseif date ==3
         category = 3;
      elseif date ==4
         category = 4;
      elseif date ==5
         category = 5;
      elseif date ==6
         category = 6;
      elseif date ==7
```

```
        category = 7;
    elseif date ==8
        category = 8;
    elseif date ==9
        category = 9;
    elseif date ==10
        category = 10;
    elseif date ==11
        category = 11;
    elseif date ==12
        category = 12;
    elseif date ==13
        category = 13;
    elseif date ==14
        category = 14;
    elseif date ==15
        category = 15;
    elseif date ==16
        category = 16;
    elseif date ==17
        category = 17;
    elseif date ==18
        category = 18;
    elseif date ==19
        category = 19;
    elseif date ==20
        category = 20;
    elseif date ==21
        category = 21;
    elseif date ==22
        category = 22;
    elseif date ==23
        category = 23;
    elseif date ==24
        category = 24;
    elseif date ==25
        category = 25;
    elseif date ==26
        category = 26;
    elseif date ==27
        category = 27;
    elseif date ==28
        category = 28;
    elseif date ==29
        category = 29;
    elseif date ==30
        category = 30;
    elseif date ==31
        category = 31;
    end;

%Change time from ZULU time to Eastern Standard Time accounting for
%Daylight Savings Time.

  if (((AirmetNum >=1453 & AirmetNum<= 1722) | (AirmetNum >=3008& AirmetNum<=3146)) & (date>=7)) |...
 ((AirmetNum >=1723 & AirmetNum<= 2002) | (AirmetNum >=3147& AirmetNum<=3280))
  for timeperiod = category
     if numline==4 & endtime == 10
        Hour0000to0600(timeperiod) = Hour0000to0600(timeperiod) + 1;
     elseif numline==4 & endtime == 11
        Hour0000to0700(timeperiod) = Hour0000to0700(timeperiod) + 1;
     elseif numline==5 & endtime == 10
```

```
    Hour0100to0600(timeperiod) = Hour0100to0600(timeperiod) + 1;
elseif numline==5 & endtime == 11
    Hour0100to0700(timeperiod) = Hour0100to0700(timeperiod) + 1;
elseif numline==5 & endtime == 12
    Hour0100to0800(timeperiod) = Hour0100to0800(timeperiod) + 1;
elseif numline==6 & endtime == 10
    Hour0200to0600(timeperiod) = Hour0200to0600(timeperiod) + 1;
elseif numline==6 & endtime == 11
    Hour0200to0700(timeperiod) = Hour0200to0700(timeperiod) + 1;
elseif numline==6 & endtime == 12
    Hour0200to0800(timeperiod) = Hour0200to0800(timeperiod) + 1;
elseif numline==6 & endtime == 13
    Hour0200to0900(timeperiod) = Hour0200to0900(timeperiod) + 1;
elseif numline==7 & endtime == 10
    Hour0300to0600(timeperiod) = Hour0300to0600(timeperiod) + 1;
elseif numline==7 & endtime == 11
    Hour0300to0700(timeperiod) = Hour0300to0700(timeperiod) + 1;
elseif numline==7 & endtime == 12
    Hour0300to0800(timeperiod) = Hour0300to0800(timeperiod) + 1;
elseif numline==7 & endtime == 13
    Hour0300to0900(timeperiod) = Hour0300to0900(timeperiod) + 1;
elseif numline==7 & endtime == 14
    Hour0300to1000(timeperiod) = Hour0300to1000(timeperiod) + 1;
elseif numline==8 & endtime == 10
    Hour0400to0600(timeperiod) = Hour0400to0600(timeperiod) + 1;
elseif numline==8 & endtime == 11
    Hour0400to0700(timeperiod) = Hour0400to0700(timeperiod) + 1;
elseif numline==8 & endtime == 12
    Hour0400to0800(timeperiod) = Hour0400to0800(timeperiod) + 1;
elseif numline==8 & endtime == 13
    Hour0400to0900(timeperiod) = Hour0400to0900(timeperiod) + 1;
elseif numline==8 & endtime == 14
    Hour0400to1000(timeperiod) = Hour0400to1000(timeperiod) + 1;
elseif numline==8 & endtime == 15
    Hour0400to1100(timeperiod) = Hour0400to1100(timeperiod) + 1;
elseif numline==9 & endtime == 10
    Hour0500to0600(timeperiod) = Hour0500to0600(timeperiod) + 1;
elseif numline==9 & endtime == 11
    Hour0500to0700(timeperiod) = Hour0500to0700(timeperiod) + 1;
elseif numline==9 & endtime == 12
    Hour0500to0800(timeperiod) = Hour0500to0800(timeperiod) + 1;
elseif numline==9 & endtime == 13
    Hour0500to0900(timeperiod) = Hour0500to0900(timeperiod) + 1;
elseif numline==9 & endtime == 14
    Hour0500to1000(timeperiod) = Hour0500to1000(timeperiod) + 1;
elseif numline==9 & endtime ==15
    Hour0500to1100(timeperiod) = Hour0500to1100(timeperiod) + 1;
elseif numline==9 & endtime ==16
    Hour0500to1200(timeperiod) = Hour0500to1200(timeperiod) + 1;
elseif numline==10 & endtime == 11
    Hour0600to0700(timeperiod) = Hour0600to0700(timeperiod) + 1;
elseif numline==10 & endtime ==12
    Hour0600to0800(timeperiod) = Hour0600to0800(timeperiod) + 1;
elseif numline==10 & endtime ==13
    Hour0600to0900(timeperiod) = Hour0600to0900(timeperiod) + 1;
elseif numline==10 & endtime ==14
    Hour0600to1000(timeperiod) = Hour0600to1000(timeperiod) + 1;
elseif numline==10 & endtime ==15
    Hour0600to1100(timeperiod) = Hour0600to1100(timeperiod) + 1;
elseif numline==10 & endtime ==16
    Hour0600to1200(timeperiod) = Hour0600to1200(timeperiod) + 1;
elseif numline==10 & endtime ==17
```

```
    Hour0600to1300(timeperiod) = Hour0600to1300(timeperiod) + 1;
elseif numline==11 & endtime == 12
    Hour0700to0800(timeperiod) = Hour0700to0800(timeperiod) + 1;
elseif numline==11 & endtime ==13
    Hour0700to0900(timeperiod) = Hour0700to0900(timeperiod) + 1;
elseif numline==11 & endtime ==14
    Hour0700to1000(timeperiod) = Hour0700to1000(timeperiod) + 1;
elseif numline==11 & endtime ==15
    Hour0700to1100(timeperiod) = Hour0700to1100(timeperiod) + 1;
elseif numline==11 & endtime ==16
    Hour0700to1200(timeperiod) = Hour0700to1200(timeperiod) + 1;
elseif numline==11 & endtime ==17
    Hour0700to1300(timeperiod) = Hour0700to1300(timeperiod) + 1;
elseif numline==11 & endtime ==18
    Hour0700to1400(timeperiod) = Hour0700to1400(timeperiod) + 1;
elseif numline==12 & endtime == 13
    Hour0800to0900(timeperiod) = Hour0800to0900(timeperiod) + 1;
elseif numline==12 & endtime ==14
    Hour0800to1000(timeperiod) = Hour0800to1000(timeperiod) + 1;
elseif numline==12 & endtime ==15
    Hour0800to1100(timeperiod) = Hour0800to1100(timeperiod) + 1;
elseif numline==12 & endtime ==16
    Hour0800to1200(timeperiod) = Hour0800to1200(timeperiod) + 1;
elseif numline==12 & endtime ==17
    Hour0800to1300(timeperiod) = Hour0800to1300(timeperiod) + 1;
elseif numline==12 & endtime ==18
    Hour0800to1400(timeperiod) = Hour0800to1400(timeperiod) + 1;
elseif numline==12 & endtime ==19
    Hour0800to1500(timeperiod) = Hour0800to1500(timeperiod) + 1;
elseif numline==13 & endtime == 14
    Hour0900to1000(timeperiod) = Hour0900to1000(timeperiod) + 1;
elseif numline==13 & endtime ==15
    Hour0900to1100(timeperiod) = Hour0900to1100(timeperiod) + 1;
elseif numline==13 & endtime ==16
    Hour0900to1200(timeperiod) = Hour0900to1200(timeperiod) + 1;
elseif numline==13 & endtime ==17
    Hour0900to1300(timeperiod) = Hour0900to1300(timeperiod) + 1;
elseif numline==13 & endtime ==18
    Hour0900to1400(timeperiod) = Hour0900to1400(timeperiod) + 1;
elseif numline==13 & endtime ==19
    Hour0900to1500(timeperiod) = Hour0900to1500(timeperiod) + 1;
elseif numline==13 & endtime ==20
    Hour0900to1600(timeperiod) = Hour0900to1600(timeperiod) + 1;
elseif numline==14 & endtime == 15
    Hour1000to1100(timeperiod) = Hour1000to1100(timeperiod) + 1;
elseif numline==14 & endtime ==16
    Hour1000to1200(timeperiod) = Hour1000to1200(timeperiod) + 1;
elseif numline==14 & endtime ==17
    Hour1000to1300(timeperiod) = Hour1000to1300(timeperiod) + 1;
elseif numline==14 & endtime ==18
    Hour1000to1400(timeperiod) = Hour1000to1400(timeperiod) + 1;
elseif numline==14 & endtime ==19
    Hour1000to1500(timeperiod) = Hour1000to1500(timeperiod) + 1;
elseif numline==14 & endtime ==20
    Hour1000to1600(timeperiod) = Hour1000to1600(timeperiod) + 1;
elseif numline==14 & endtime ==21
    Hour1000to1700(timeperiod) = Hour1000to1700(timeperiod) + 1;
elseif numline==15 & endtime == 16
    Hour1100to1200(timeperiod) = Hour1100to1200(timeperiod) + 1;
elseif numline==15 & endtime ==17
    Hour1100to1300(timeperiod) = Hour1100to1300(timeperiod) + 1;
elseif numline==15 & endtime ==18
```

```
   Hour1100to1400(timeperiod) = Hour1100to1400(timeperiod) + 1;
elseif numline==15 & endtime ==19
   Hour1100to1500(timeperiod) = Hour1100to1500(timeperiod) + 1;
elseif numline==15 & endtime ==20
   Hour1100to1600(timeperiod) = Hour1100to1600(timeperiod) + 1;
elseif numline==15 & endtime ==21
   Hour1100to1700(timeperiod) = Hour1100to1700(timeperiod) + 1;
elseif numline==15 & endtime ==22
   Hour1100to1800(timeperiod) = Hour1100to1800(timeperiod) + 1;
elseif numline==16 & endtime == 17
   Hour1200to1300(timeperiod) = Hour1200to1300(timeperiod) + 1;
elseif numline==16 & endtime ==18
   Hour1200to1400(timeperiod) = Hour1200to1400(timeperiod) + 1;
elseif numline==16 & endtime ==19
   Hour1200to1500(timeperiod) = Hour1200to1500(timeperiod) + 1;
elseif numline==16 & endtime ==20
   Hour1200to1600(timeperiod) = Hour1200to1600(timeperiod) + 1;
elseif numline==16 & endtime ==21
   Hour1200to1700(timeperiod) = Hour1200to1700(timeperiod) + 1;
elseif numline==16 & endtime ==22
   Hour1200to1800(timeperiod) = Hour1200to1800(timeperiod) + 1;
elseif numline==16 & endtime ==23
   Hour1200to1900(timeperiod) = Hour1200to1900(timeperiod) + 1;
elseif numline==17 & endtime == 18
   Hour1300to1400(timeperiod) = Hour1300to1400(timeperiod) + 1;
elseif numline==17 & endtime ==19
   Hour1300to1500(timeperiod) = Hour1300to1500(timeperiod) + 1;
elseif numline==17 & endtime ==20
   Hour1300to1600(timeperiod) = Hour1300to1600(timeperiod) + 1;
elseif numline==17 & endtime ==21
   Hour1300to1700(timeperiod) = Hour1300to1700(timeperiod) + 1;
elseif numline==17 & endtime ==22
   Hour1300to1800(timeperiod) = Hour1300to1800(timeperiod) + 1;
elseif numline==17 & endtime ==23
   Hour1300to1900(timeperiod) = Hour1300to1900(timeperiod) + 1;
elseif numline==17 & endtime ==00
   Hour1300to2000(timeperiod) = Hour1300to2000(timeperiod) + 1;
elseif numline==18 & endtime == 19
   Hour1400to1500(timeperiod) = Hour1400to1500(timeperiod) + 1;
elseif numline==18 & endtime ==20
   Hour1400to1600(timeperiod) = Hour1400to1600(timeperiod) + 1;
elseif numline==18 & endtime ==21
   Hour1400to1700(timeperiod) = Hour1400to1700(timeperiod) + 1;
elseif numline==18 & endtime ==22
   Hour1400to1800(timeperiod) = Hour1400to1800(timeperiod) + 1;
elseif numline==18 & endtime ==23
   Hour1400to1900(timeperiod) = Hour1400to1900(timeperiod) + 1;
elseif numline==18 & endtime ==00
   Hour1400to2000(timeperiod) = Hour1400to2000(timeperiod) + 1;
elseif numline==18 & endtime ==01
   Hour1400to2100(timeperiod) = Hour1400to2100(timeperiod) + 1;
elseif numline==19 & endtime == 20
   Hour1500to1600(timeperiod) = Hour1500to1600(timeperiod) + 1;
elseif numline==19 & endtime ==21
   Hour1500to1700(timeperiod) = Hour1500to1700(timeperiod) + 1;
elseif numline==19 & endtime ==22
   Hour1500to1800(timeperiod) = Hour1500to1800(timeperiod) + 1;
elseif numline==19 & endtime ==23
   Hour1500to1900(timeperiod) = Hour1500to1900(timeperiod) + 1;
elseif numline==19 & endtime ==00
   Hour1500to2000(timeperiod) = Hour1500to2000(timeperiod) + 1;
elseif numline==19 & endtime ==01
```

```
   Hour1500to2100(timeperiod) = Hour1500to2100(timeperiod) + 1;
elseif numline==19 & endtime ==02
   Hour1500to2200(timeperiod) = Hour1500to2200(timeperiod) + 1;
elseif numline==20 & endtime == 21
   Hour1600to1700(timeperiod) = Hour1600to1700(timeperiod) + 1;
elseif numline==20 & endtime ==22
   Hour1600to1800(timeperiod) = Hour1600to1800(timeperiod) + 1;
elseif numline==20 & endtime ==23
   Hour1600to1900(timeperiod) = Hour1600to1900(timeperiod) + 1;
elseif numline==20 & endtime ==00
   Hour1600to2000(timeperiod) = Hour1600to2000(timeperiod) + 1;
elseif numline==20 & endtime ==01
   Hour1600to2100(timeperiod) = Hour1600to2100(timeperiod) + 1;
elseif numline==20 & endtime ==02
   Hour1600to2200(timeperiod) = Hour1600to2200(timeperiod) + 1;
elseif numline==20 & endtime ==03
   Hour1600to2300(timeperiod) = Hour1600to2300(timeperiod) + 1;
elseif numline==21 & endtime == 22
   Hour1700to1800(timeperiod) = Hour1700to1800(timeperiod) + 1;
elseif numline==21 & endtime ==23
   Hour1700to1900(timeperiod) = Hour1700to1900(timeperiod) + 1;
elseif numline==21 & endtime ==00
   Hour1700to2000(timeperiod) = Hour1700to2000(timeperiod) + 1;
elseif numline==21 & endtime ==01
   Hour1700to2100(timeperiod) = Hour1700to2100(timeperiod) + 1;
elseif numline==21 & endtime ==02
   Hour1700to2200(timeperiod) = Hour1700to2200(timeperiod) + 1;
elseif numline==21 & endtime ==03
   Hour1700to2300(timeperiod) = Hour1700to2300(timeperiod) + 1;
elseif numline==21 & endtime ==04
   Hour1700to0000(timeperiod) = Hour1700to0000(timeperiod) + 1;
elseif numline==22 & endtime == 23
   Hour1800to1900(timeperiod) = Hour1800to1900(timeperiod) + 1;
elseif numline==22 & endtime ==00
   Hour1800to2000(timeperiod) = Hour1800to2000(timeperiod) + 1;
elseif numline==22 & endtime ==01
   Hour1800to2100(timeperiod) = Hour1800to2100(timeperiod) + 1;
elseif numline==22 & endtime ==02
   Hour1800to2200(timeperiod) = Hour1800to2200(timeperiod) + 1;
elseif numline==22 & endtime ==03
   Hour1800to2300(timeperiod) = Hour1800to2300(timeperiod) + 1;
elseif numline==22 & endtime ==04
   Hour1800to0000(timeperiod) = Hour1800to0000(timeperiod) + 1;
elseif numline==22 & endtime ==05
   Hour1800to0100(timeperiod) = Hour1800to0100(timeperiod) + 1;
elseif numline==23 & endtime == 00
   Hour1900to2000(timeperiod) = Hour1900to2000(timeperiod) + 1;
elseif numline==23 & endtime ==01
   Hour1900to2100(timeperiod) = Hour1900to2100(timeperiod) + 1;
elseif numline==23 & endtime ==02
   Hour1900to2200(timeperiod) = Hour1900to2200(timeperiod) + 1;
elseif numline==23 & endtime ==03
   Hour1900to2300(timeperiod) = Hour1900to2300(timeperiod) + 1;
elseif numline==23 & endtime ==04
   Hour1900to0000(timeperiod) = Hour1900to0000(timeperiod) + 1;
elseif numline==23 & endtime ==05
   Hour1900to0100(timeperiod) = Hour1900to0100(timeperiod) + 1;
elseif numline==23 & endtime ==06
   Hour1900to0200(timeperiod) = Hour1900to0200(timeperiod) + 1;
elseif numline==00 & endtime == 01
   Hour2000to2100(timeperiod) = Hour2000to2100(timeperiod) + 1;
elseif numline==00 & endtime ==02
```

```
        Hour2000to2200(timeperiod) = Hour2000to2200(timeperiod) + 1;
      elseif numline==00 & endtime ==03
        Hour2000to2300(timeperiod) = Hour2000to2300(timeperiod) + 1;
      elseif numline==00 & endtime ==04
        Hour2000to0000(timeperiod) = Hour2000to0000(timeperiod) + 1;
      elseif numline==00 & endtime ==05
        Hour2000to0100(timeperiod) = Hour2000to0100(timeperiod) + 1;
      elseif numline==00 & endtime ==06
        Hour2000to0200(timeperiod) = Hour2000to0200(timeperiod) + 1;
      elseif numline==00 & endtime ==07
        Hour2000to0300(timeperiod) = Hour2000to02300(timeperiod) + 1;

      end;
    end;

%Change time from ZULU time to Eastern Standard Time

else
    for timeperiod = category
      if numline==4 & endtime == 11
        Hour2300to0600(timeperiod) = Hour2300to0600(timeperiod) + 1;
      elseif numline==5 & endtime == 11
        Hour0000to0600(timeperiod) = Hour0000to0600(timeperiod) + 1;
      elseif numline==5 & endtime == 12
        Hour0000to0700(timeperiod) = Hour0000to0700(timeperiod) + 1;
      elseif numline==6 & endtime == 11
        Hour0100to0600(timeperiod) = Hour0100to0600(timeperiod) + 1;
      elseif numline==6 & endtime == 12
        Hour0100to0700(timeperiod) = Hour0100to0700(timeperiod) + 1;
      elseif numline==6 & endtime == 13
        Hour0100to0800(timeperiod) = Hour0100to0800(timeperiod) + 1;
      elseif numline==7 & endtime == 11
        Hour0200to0600(timeperiod) = Hour0200to0600(timeperiod) + 1;
      elseif numline==7 & endtime == 12
        Hour0200to0700(timeperiod) = Hour0200to0700(timeperiod) + 1;
      elseif numline==7 & endtime == 13
        Hour0200to0800(timeperiod) = Hour0200to0800(timeperiod) + 1;
      elseif numline==7 & endtime == 14
        Hour0200to0900(timeperiod) = Hour0200to0900(timeperiod) + 1;
      elseif numline==8 & endtime == 11
        Hour0300to0600(timeperiod) = Hour0300to0600(timeperiod) + 1;
      elseif numline==8 & endtime == 12
        Hour0300to0700(timeperiod) = Hour0300to0700(timeperiod) + 1;
      elseif numline==8 & endtime == 13
        Hour0300to0800(timeperiod) = Hour0300to0800(timeperiod) + 1;
      elseif numline==8 & endtime == 14
        Hour0300to0900(timeperiod) = Hour0300to0900(timeperiod) + 1;
      elseif numline==8 & endtime == 15
        Hour0300to1000(timeperiod) = Hour0300to1000(timeperiod) + 1;
      elseif numline==9 & endtime == 11
        Hour0400to0600(timeperiod) = Hour0400to0600(timeperiod) + 1;
      elseif numline==9 & endtime == 12
        Hour0400to0700(timeperiod) = Hour0400to0700(timeperiod) + 1;
      elseif numline==9 & endtime == 13
        Hour0400to0800(timeperiod) = Hour0400to0800(timeperiod) + 1;
      elseif numline==9 & endtime == 14
        Hour0400to0900(timeperiod) = Hour0400to0900(timeperiod) + 1;
      elseif numline==9 & endtime ==15
        Hour0400to1000(timeperiod) = Hour0400to1000(timeperiod) + 1;
      elseif numline==9 & endtime ==16
        Hour0400to1100(timeperiod) = Hour0400to1100(timeperiod) + 1;
      elseif numline==10 & endtime == 11
```

```
    Hour0500to0600(timeperiod) = Hour0500to0600(timeperiod) + 1;
elseif numline==10 & endtime ==12
    Hour0500to0700(timeperiod) = Hour0500to0700(timeperiod) + 1;
elseif numline==10 & endtime ==13
    Hour0500to0800(timeperiod) = Hour0500to0800(timeperiod) + 1;
elseif numline==10 & endtime ==14
    Hour0500to0900(timeperiod) = Hour0500to0900(timeperiod) + 1;
elseif numline==10 & endtime ==15
    Hour0500to1000(timeperiod) = Hour0500to1000(timeperiod) + 1;
elseif numline==10 & endtime ==16
    Hour0500to1100(timeperiod) = Hour0500to1100(timeperiod) + 1;
elseif numline==10 & endtime ==17
    Hour0500to1200(timeperiod) = Hour0500to1200(timeperiod) + 1;
elseif numline==11 & endtime == 12
    Hour0600to0700(timeperiod) = Hour0600to0700(timeperiod) + 1;
elseif numline==11 & endtime ==13
    Hour0600to0800(timeperiod) = Hour0600to0800(timeperiod) + 1;
elseif numline==11 & endtime ==14
    Hour0600to0900(timeperiod) = Hour0600to0900(timeperiod) + 1;
elseif numline==11 & endtime ==15
    Hour0600to1000(timeperiod) = Hour0600to1000(timeperiod) + 1;
elseif numline==11 & endtime ==16
    Hour0600to1100(timeperiod) = Hour0600to1100(timeperiod) + 1;
elseif numline==11 & endtime ==17
    Hour0600to1200(timeperiod) = Hour0600to1200(timeperiod) + 1;
elseif numline==11 & endtime ==18
    Hour0600to1300(timeperiod) = Hour0600to1300(timeperiod) + 1;
elseif numline==12 & endtime == 13
    Hour0700to0800(timeperiod) = Hour0700to0800(timeperiod) + 1;
elseif numline==12 & endtime ==14
    Hour0700to0900(timeperiod) = Hour0700to0900(timeperiod) + 1;
elseif numline==12 & endtime ==15
    Hour0700to1000(timeperiod) = Hour0700to1000(timeperiod) + 1;
elseif numline==12 & endtime ==16
    Hour0700to1100(timeperiod) = Hour0700to1100(timeperiod) + 1;
elseif numline==12 & endtime ==17
    Hour0700to1200(timeperiod) = Hour0700to1200(timeperiod) + 1;
elseif numline==12 & endtime ==18
    Hour0700to1300(timeperiod) = Hour0700to1300(timeperiod) + 1;
elseif numline==12 & endtime ==19
    Hour0700to1400(timeperiod) = Hour0700to1400(timeperiod) + 1;
elseif numline==13 & endtime == 14
    Hour0800to0900(timeperiod) = Hour0800to0900(timeperiod) + 1;
elseif numline==13 & endtime ==15
    Hour0800to1000(timeperiod) = Hour0800to1000(timeperiod) + 1;
elseif numline==13 & endtime ==16
    Hour0800to1100(timeperiod) = Hour0800to1100(timeperiod) + 1;
elseif numline==13 & endtime ==17
    Hour0800to1200(timeperiod) = Hour0800to1200(timeperiod) + 1;
elseif numline==13 & endtime ==18
    Hour0800to1300(timeperiod) = Hour0800to1300(timeperiod) + 1;
elseif numline==13 & endtime ==19
    Hour0800to1400(timeperiod) = Hour0800to1400(timeperiod) + 1;
elseif numline==13 & endtime ==20
    Hour0800to1500(timeperiod) = Hour0800to1500(timeperiod) + 1;
elseif numline==14 & endtime == 15
    Hour0900to1000(timeperiod) = Hour0900to1000(timeperiod) + 1;
elseif numline==14 & endtime ==16
    Hour0900to1100(timeperiod) = Hour0900to1100(timeperiod) + 1;
elseif numline==14 & endtime ==17
    Hour0900to1200(timeperiod) = Hour0900to1200(timeperiod) + 1;
elseif numline==14 & endtime ==18
```

```
    Hour0900to1300(timeperiod) = Hour0900to1300(timeperiod) + 1;
elseif numline==14 & endtime ==19
    Hour0900to1400(timeperiod) = Hour0900to1400(timeperiod) + 1;
elseif numline==14 & endtime ==20
    Hour0900to1500(timeperiod) = Hour0900to1500(timeperiod) + 1;
elseif numline==14 & endtime ==21
    Hour0900to1600(timeperiod) = Hour0900to1600(timeperiod) + 1;
elseif numline==15 & endtime == 16
    Hour1000to1100(timeperiod) = Hour1000to1100(timeperiod) + 1;
elseif numline==15 & endtime ==17
    Hour1000to1200(timeperiod) = Hour1000to1200(timeperiod) + 1;
elseif numline==15 & endtime ==18
    Hour1000to1300(timeperiod) = Hour1000to1300(timeperiod) + 1;
elseif numline==15 & endtime ==19
    Hour1000to1400(timeperiod) = Hour1000to1400(timeperiod) + 1;
elseif numline==15 & endtime ==20
    Hour1000to1500(timeperiod) = Hour1000to1500(timeperiod) + 1;
elseif numline==15 & endtime ==21
    Hour1000to1600(timeperiod) = Hour1000to1600(timeperiod) + 1;
elseif numline==15 & endtime ==22
    Hour1000to1700(timeperiod) = Hour1000to1700(timeperiod) + 1;
elseif numline==16 & endtime == 17
    Hour1100to1200(timeperiod) = Hour1100to1200(timeperiod) + 1;
elseif numline==16 & endtime ==18
    Hour1100to1300(timeperiod) = Hour1100to1300(timeperiod) + 1;
elseif numline==16 & endtime ==19
    Hour1100to1400(timeperiod) = Hour1100to1400(timeperiod) + 1;
elseif numline==16 & endtime ==20
    Hour1100to1500(timeperiod) = Hour1100to1500(timeperiod) + 1;
elseif numline==16 & endtime ==21
    Hour1100to1600(timeperiod) = Hour1100to1600(timeperiod) + 1;
elseif numline==16 & endtime ==22
    Hour1100to1700(timeperiod) = Hour1100to1700(timeperiod) + 1;
elseif numline==16 & endtime ==23
    Hour1100to1800(timeperiod) = Hour1100to1800(timeperiod) + 1;
elseif numline==17 & endtime == 18
    Hour1200to1300(timeperiod) = Hour1200to1300(timeperiod) + 1;
elseif numline==17 & endtime ==19
    Hour1200to1400(timeperiod) = Hour1200to1400(timeperiod) + 1;
elseif numline==17 & endtime ==20
    Hour1200to1500(timeperiod) = Hour1200to1500(timeperiod) + 1;
elseif numline==17 & endtime ==21
    Hour1200to1600(timeperiod) = Hour1200to1600(timeperiod) + 1;
elseif numline==17 & endtime ==22
    Hour1200to1700(timeperiod) = Hour1200to1700(timeperiod) + 1;
elseif numline==17 & endtime ==23
    Hour1200to1800(timeperiod) = Hour1200to1800(timeperiod) + 1;
elseif numline==17 & endtime ==00
    Hour1200to1900(timeperiod) = Hour1200to1900(timeperiod) + 1;
elseif numline==18 & endtime == 19
    Hour1300to1400(timeperiod) = Hour1300to1400(timeperiod) + 1;
elseif numline==18 & endtime ==20
    Hour1300to1500(timeperiod) = Hour1300to1500(timeperiod) + 1;
elseif numline==18 & endtime ==21
    Hour1300to1600(timeperiod) = Hour1300to1600(timeperiod) + 1;
elseif numline==18 & endtime ==22
    Hour1300to1700(timeperiod) = Hour1300to1700(timeperiod) + 1;
elseif numline==18 & endtime ==23
    Hour1300to1800(timeperiod) = Hour1300to1800(timeperiod) + 1;
elseif numline==18 & endtime ==00
    Hour1300to1900(timeperiod) = Hour1300to19000(timeperiod) + 1;
elseif numline==18 & endtime ==01
```

```
    Hour1300to2000(timeperiod) = Hour1300to2000(timeperiod) + 1;
elseif numline==19 & endtime == 20
    Hour1400to1500(timeperiod) = Hour1400to1500(timeperiod) + 1;
elseif numline==19 & endtime ==21
    Hour1400to1600(timeperiod) = Hour1400to1600(timeperiod) + 1;
elseif numline==19 & endtime ==22
    Hour1400to1700(timeperiod) = Hour1400to1700(timeperiod) + 1;
elseif numline==19 & endtime ==23
    Hour1400to1800(timeperiod) = Hour1400to1800(timeperiod) + 1;
elseif numline==19 & endtime ==00
    Hour1400to1900(timeperiod) = Hour1400to1900(timeperiod) + 1;
elseif numline==19 & endtime ==01
    Hour1400to2000(timeperiod) = Hour1400to2000(timeperiod) + 1;
elseif numline==19 & endtime ==02
    Hour1400to2100(timeperiod) = Hour1400to2100(timeperiod) + 1;
elseif numline==20 & endtime == 21
    Hour1500to1600(timeperiod) = Hour1500to1600(timeperiod) + 1;
elseif numline==20 & endtime ==22
    Hour1500to1700(timeperiod) = Hour1500to1700(timeperiod) + 1;
elseif numline==20 & endtime ==23
    Hour1500to1800(timeperiod) = Hour1500to1800(timeperiod) + 1;
elseif numline==20 & endtime ==00
    Hour1500to1900(timeperiod) = Hour1500to1900(timeperiod) + 1;
elseif numline==20 & endtime ==01
    Hour1500to2000(timeperiod) = Hour1500to2000(timeperiod) + 1;
elseif numline==20 & endtime ==02
    Hour1500to2100(timeperiod) = Hour1500to2100(timeperiod) + 1;
elseif numline==20 & endtime ==03
    Hour1500to2200(timeperiod) = Hour1500to2200(timeperiod) + 1;
elseif numline==21 & endtime == 22
    Hour1600to1700(timeperiod) = Hour1600to1700(timeperiod) + 1;
elseif numline==21 & endtime ==23
    Hour1600to1800(timeperiod) = Hour1600to1800(timeperiod) + 1;
elseif numline==21 & endtime ==00
    Hour1600to1900(timeperiod) = Hour1600to1900(timeperiod) + 1;
elseif numline==21 & endtime ==01
    Hour1600to2000(timeperiod) = Hour1600to2000(timeperiod) + 1;
elseif numline==21 & endtime ==02
    Hour1600to2100(timeperiod) = Hour1600to2100(timeperiod) + 1;
elseif numline==21 & endtime ==03
    Hour1600to2200(timeperiod) = Hour1600to2200(timeperiod) + 1;
elseif numline==21 & endtime ==04
    Hour1600to2300(timeperiod) = Hour1600to2300(timeperiod) + 1;
elseif numline==22 & endtime == 23
    Hour1700to1800(timeperiod) = Hour1700to1800(timeperiod) + 1;
elseif numline==22 & endtime ==00
    Hour1700to1900(timeperiod) = Hour1700to1900(timeperiod) + 1;
elseif numline==22 & endtime ==01
    Hour1700to2000(timeperiod) = Hour1700to2000(timeperiod) + 1;
elseif numline==22 & endtime ==02
    Hour1700to2100(timeperiod) = Hour1700to2100(timeperiod) + 1;
elseif numline==22 & endtime ==03
    Hour1700to2200(timeperiod) = Hour1700to2200(timeperiod) + 1;
elseif numline==22 & endtime ==04
    Hour1700to2300(timeperiod) = Hour1700to2300(timeperiod) + 1;
elseif numline==22 & endtime ==05
    Hour1700to0000(timeperiod) = Hour1700to0000(timeperiod) + 1;
elseif numline==23 & endtime == 00
    Hour1800to1900(timeperiod) = Hour1800to1900(timeperiod) + 1;
elseif numline==23 & endtime ==01
    Hour1800to2000(timeperiod) = Hour1800to2000(timeperiod) + 1;
elseif numline==23 & endtime ==02
```

```
        Hour1800to2100(timeperiod) = Hour1800to2100(timeperiod) + 1;
      elseif numline==23 & endtime ==03
        Hour1800to2200(timeperiod) = Hour1800to2200(timeperiod) + 1;
      elseif numline==23 & endtime ==04
        Hour1800to2300(timeperiod) = Hour1800to2300(timeperiod) + 1;
      elseif numline==23 & endtime ==05
        Hour1800to0000(timeperiod) = Hour1800to0000(timeperiod) + 1;
      elseif numline==23 & endtime ==06
        Hour1800to0100(timeperiod) = Hour1800to0100(timeperiod) + 1;
      elseif numline==00 & endtime == 01
        Hour1900to2000(timeperiod) = Hour1900to2000(timeperiod) + 1;
      elseif numline==00 & endtime ==02
        Hour1900to2100(timeperiod) = Hour1900to2100(timeperiod) + 1;
      elseif numline==00 & endtime ==03
        Hour1900to2200(timeperiod) = Hour1900to2200(timeperiod) + 1;
      elseif numline==00 & endtime ==04
        Hour1900to2300(timeperiod) = Hour1900to2300(timeperiod) + 1;
      elseif numline==00 & endtime ==05
        Hour1900to0000(timeperiod) = Hour1900to0000(timeperiod) + 1;
      elseif numline==00 & endtime ==06
        Hour1900to0100(timeperiod) = Hour1900to0100(timeperiod) + 1;
      elseif numline==00 & endtime ==07
        Hour1900to0200(timeperiod) = Hour1900to0200(timeperiod) + 1;
      elseif numline==01 & endtime ==02
        Hour2000to2100(timeperiod) = Hour2000to2100(timeperiod) + 1;
      elseif numline==01 & endtime ==03
        Hour2000to2200(timeperiod) = Hour2000to2200(timeperiod) + 1;
      elseif numline==01 & endtime ==04
        Hour2000to2300(timeperiod) = Hour2000to2300(timeperiod) + 1;
      elseif numline==01 & endtime ==05
        Hour2000to0000(timeperiod) = Hour2000to0000(timeperiod) + 1;
      elseif numline==01 & endtime ==06
        Hour2000to0100(timeperiod) = Hour2000to0100(timeperiod) + 1;
      elseif numline==01 & endtime ==07
        Hour2000to0200(timeperiod) = Hour2000to0200(timeperiod) + 1;
      elseif numline==01 & endtime ==08
        Hour2000to0300(timeperiod) = Hour2000to0300(timeperiod) + 1;
      end;
   end;
end;
      end;
   line = fgetl(StartEnd);
end;

%Print results to a separate file for each day.

for i=1:31
if Hour0000to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0000to0600'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0000to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0000to0700'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0100to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0100to0600'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0100to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0100to0700'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0100to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0100to0800'); fprintf(TimePeriods(i), '%g\n', '1');
end;
```

```
if Hour0200to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0200to0600'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0200to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0200to0700'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0200to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0200to0800'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0200to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0200to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0300to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0300to0600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0300to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0300to0700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0300to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0300to0800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0300to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0300to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0300to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0300to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to0600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to0700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to0800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0400to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0400to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to0600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to0600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to0700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to0800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0500to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to1100'); fprintf(TimePeriods(i), '%s\n', '1');
```

```
end;
if Hour0500to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0500to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to0700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to0700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to0800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to0900'); fprintf(TimePeriods(i), '%g\n', '1');
end;
if Hour0600to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0600to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0600to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to0800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to0800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0700to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0700to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to0900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to0900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1400(i)>=1
```

```
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0800to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0800to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour0900to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour0900to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1000to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1000to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
```

```
if Hour1100to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1100to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1100to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1200to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1200to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1400(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1400'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1300to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1300to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to1500(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to1500'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to1900'); fprintf(TimePeriods(i), '%s\n', '1');
```

```
end;
if Hour1400to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1400to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1400to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to1600(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to1600'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1500to2200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1500to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to1700(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to1700'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to2200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1600to2300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1600to2300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to1800(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to1800'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to2200(i)>=1
```

```
fprintf(TimePeriods(i), '%s\n', 'Hour1700to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to2300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to2300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1700to0000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1700to0000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to1900(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to1900'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to2200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to2300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to2300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to0000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to0000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1800to0100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1800to0100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to2000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to2000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to2200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to2300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to2300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to0000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to0000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to0100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to0100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour1900to0200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour1900to0200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to2100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to2100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to2200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to2200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to2300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to2300'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to0000(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to0000'); fprintf(TimePeriods(i), '%s\n', '1');
end;
```

```
if Hour2000to0100(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to0100'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to0200(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to0200'); fprintf(TimePeriods(i), '%s\n', '1');
end;
if Hour2000to0300(i)>=1
fprintf(TimePeriods(i), '%s\n', 'Hour2000to0300'); fprintf(TimePeriods(i), '%s\n', '1');
end;

fprintf(TimePeriods(i), '%s\n', 'Total');
Total = Hour0000to0600(i) + Hour0000to0700(i) + Hour0100to0600(i) + Hour0100to0700(i) + Hour0100to0800(i) +
Hour0200to0600(i) + ...
   Hour0200to0700(i) + Hour0200to0800(i) + Hour0200to0900(i) + Hour0300to0600(i) + Hour0300to0700(i) +
Hour0300to0800(i) + ...
   Hour0300to0900(i) + Hour0300to1000(i) + Hour0400to0600(i) + Hour0400to0700(i) + Hour0400to0800(i) +
Hour0400to0900(i) + ...
   Hour0400to1000(i) + Hour0400to1100(i) + Hour0500to0600(i) + Hour0500to0700(i) + Hour0500to0800(i) +
Hour0500to0900(i) + ...
   Hour0500to1000(i) + Hour0500to1100(i) + Hour0500to1200(i) + Hour0600to0700(i) + Hour0600to0800(i) +
Hour0600to0900(i) + ...
   Hour0600to1000(i) + Hour0600to1100(i) + Hour0600to1200(i) + Hour0600to1300(i) + Hour0700to0800(i) +
Hour0700to0900(i) + ...
   Hour0700to1000(i) + Hour0700to1100(i) + Hour0700to1200(i) + Hour0700to1300(i) + Hour0700to1400(i) +
Hour0800to0900(i) + ...
   Hour0800to1000(i) + Hour0800to1100(i) + Hour0800to1200(i) + Hour0800to1300(i) + Hour0800to1400(i) +
Hour0800to1500(i) + ...
   Hour0900to1000(i) + Hour0900to1100(i) + Hour0900to1200(i) + Hour0900to1300(i) + Hour0900to1400(i) +
Hour0900to1500(i) + ...
   Hour0900to1600(i) + Hour1000to1100(i) + Hour1000to1200(i) + Hour1000to1300(i) + Hour1000to1400(i) +
Hour1000to1500(i) + ...
   Hour1000to1600(i) + Hour1000to1700(i) + Hour1100to1200(i) + Hour1100to1300(i) + Hour1100to1400(i) +
Hour1100to1500(i) + ...
   Hour1100to1600(i) + Hour1100to1700(i) + Hour1100to1800(i) + Hour1200to1300(i) + Hour1200to1400(i) +
Hour1200to1500(i) + ...
   Hour1200to1600(i) + Hour1200to1700(i) + Hour1200to1800(i) + Hour1200to1900(i) + Hour1300to1400(i) +
Hour1300to1500(i) + ...
   Hour1300to1600(i) + Hour1300to1700(i) + Hour1300to1800(i) + Hour1300to1900(i) + Hour1300to2000(i) +
Hour1400to1500(i) + ...
   Hour1400to1600(i) + Hour1400to1700(i) + Hour1400to1800(i) + Hour1400to1900(i) + Hour1400to2000(i) +
Hour1400to2100(i) + ...
   Hour1500to1600(i) + Hour1500to1700(i) + Hour1500to1800(i) + Hour1500to1900(i) + Hour1500to2000(i) +
Hour1500to2100(i) + ...
   Hour1500to2200(i) + Hour1600to1700(i) + Hour1600to1800(i) + Hour1600to1900(i) + Hour1600to2000(i) +
Hour1600to2100(i) + ...
   Hour1600to2200(i) + Hour1600to2300(i) + Hour1700to1800(i) + Hour1700to1900(i) + Hour1700to2000(i) +
Hour1700to2100(i) + ...
   Hour1700to2200(i) + Hour1700to2300(i) + Hour1700to0000(i) + Hour1800to1900(i) + Hour1800to2000(i) +
Hour1800to2100(i) + ...
   Hour1800to2200(i) + Hour1800to2300(i) + Hour1800to0000(i) + Hour1800to0100(i) + Hour1900to2000(i) +
Hour1900to2100(i) + ...
   Hour1900to2200(i) + Hour1900to2300(i) + Hour1900to0000(i) + Hour1900to0100(i) + Hour1900to0200(i) +
Hour2000to2100(i) + ...
   Hour2000to2200(i) + Hour2000to2300(i) + Hour2000to0000(i) + Hour2000to0100(i) + Hour2000to0200(i) +
Hour2000to0300(i);
fprintf(TimePeriods(i), '%g\n', Total);
end;

fclose(StartEnd);
fclose(TimePeriods(1));
fclose(TimePeriods(2));
fclose(TimePeriods(3));
```

```
fclose(TimePeriods(4));
fclose(TimePeriods(5));
fclose(TimePeriods(6));
fclose(TimePeriods(7));
fclose(TimePeriods(8));
fclose(TimePeriods(9));
fclose(TimePeriods(10));
fclose(TimePeriods(11));
fclose(TimePeriods(12));
fclose(TimePeriods(13));
fclose(TimePeriods(14));
fclose(TimePeriods(15));
fclose(TimePeriods(16));
fclose(TimePeriods(17));
fclose(TimePeriods(18));
fclose(TimePeriods(19));
fclose(TimePeriods(20));
fclose(TimePeriods(21));
fclose(TimePeriods(22));
fclose(TimePeriods(23));
fclose(TimePeriods(24));
fclose(TimePeriods(25));
fclose(TimePeriods(26));
fclose(TimePeriods(27));
fclose(TimePeriods(28));
fclose(TimePeriods(29));
fclose(TimePeriods(30));
fclose(TimePeriods(31));
```

APPENDIX J: Flights Over Maximum Altitude of AIRMET- BOSCLE

```
% FlightsOverMaximumAltitudeofAIRMETBOSCLE

% Programmed by: Melinda Gates (08/2004)
%--------------------------------------------------------------------------

clear all;
load BOSCLEalt;

% Find Flights Flying Over AIRMET
%-------------------------------------------------------------------------------------------------------------------

IntersectionsBOSCLE = fopen('NonIntersectingAirmetsBOSCLE.txt', 'rt');
FlightOverMaxAlt = fopen('FlightOverAltitudeBOSCLE.txt', 'w');

while (not(feof(IntersectionsBOSCLE)))
   line = fgetl(IntersectionsBOSCLE);
   line2 = str2num(line);
for i = 1:30
   if BOSCLEalt(i) > str2num(char(WEATHER_OUTPUT(line2).ALTITUDEMAX1))
      fprintf(FlightOverMaxAlt, '%g\n', line2);
   elseif BOSCLEalt(i) > str2num(char(WEATHER_OUTPUT(line2).ALTITUDEMAX2))
      fprintf(FlightOverMaxAlt, '%g\n', line2);
   elseif BOSCLEalt(i) > str2num(char(WEATHER_OUTPUT(line2).ALTITUDEMAX3))
      fprintf(FlightOverMaxAlt, '%g\n', line2);
   elseif BOSCLEalt(i) > str2num(char(WEATHER_OUTPUT(line2).ALTITUDEMAX4))
      fprintf(FlightOverMaxAlt, '%g\n', line2);
   elseif BOSCLEalt(i) > str2num(char(WEATHER_OUTPUT(line2).ALTITUDEMAX5))
      fprintf(FlightOverMaxAlt, '%g\n', line2);
   end
end
end

fclose(IntersectionsBOSCLE);
fclose(FlightOverMaxAlt);

% Eliminate Redundancies
%-------------------------------------------------------------------------------------------------------------------

FlightOverBOSCLE = fopen('FlightOverAltitudeBOSCLE.txt', 'rt');
FlightOverBOSCLENoRepeat = fopen('FlightOverAltitudeBOSCLENR.txt', 'w');

 line = fgetl(FlightOverBOSCLE);
   line2 = fgetl(FlightOverBOSCLE);
   if strmatch(line,line2)
      fprintf(FlightOverBOSCLENoRepeat, '%s\n', line);
   else
      fprintf(FlightOverBOSCLENoRepeat, '%s\n', line);
      fprintf(FlightOverBOSCLENoRepeat, '%s\n', line2);
   end;
while (not(feof(FlightOverBOSCLE)))
   line = line2;
   line2 = fgetl(FlightOverBOSCLE);
   if isempty (strmatch(line,line2))
      fprintf(FlightOverBOSCLENoRepeat, '%s\n', line2);
   end;
end;

fclose(FlightOverBOSCLE);
fclose(FlightOverBOSCLENoRepeat);
```

APPENDIX K: Reliability Measure Spreadsheets

BOS to CLE Unreliability Measures for Monthly Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	9	9	9	12	9	9	9	9	9	13	11	11	11	11	141
November B	24	24	24	26	23	23	23	23	23	24	23	23	23	23	329
Hour Total	33	33	33	38	32	32	32	32	32	37	34	34	34	34	470
TOTAL		137				160					173				470
Fraction		0.570833333				0.533333333					0.576666667				
December A	16	16	16	22	20	20	20	20	20	21	18	19	19	19	266
December B	22	22	22	24	23	23	23	23	23	25	22	22	22	22	318
Hour Total	38	38	38	46	43	43	43	43	43	46	40	41	41	41	584
TOTAL		160				215					209				584
Fraction		0.64516129				0.693548387					0.674193548				
January A	21	22	22	23	21	21	21	21	21	22	20	20	20	19	294
January B	24	24	24	25	24	24	24	24	24	26	23	23	23	23	335
Hour Total	45	46	46	48	45	45	45	45	45	48	43	43	43	42	629
TOTAL		185				225					219				629
Fraction		0.745967742				0.725806452					0.706451613				
February A	18	18	18	21	16	16	16	16	16	18	15	15	15	15	233
February B	19	19	19	21	16	16	16	16	16	20	18	18	18	17	249
Hour Total	37	37	37	42	32	32	32	32	32	38	33	33	33	32	482
TOTAL		153				160					169				482
Fraction		0.683035714				0.571428571					0.603571429				
March A	21	23	23	26	21	21	21	21	21	24	21	21	21	20	305
March B	16	16	16	18	18	18	18	18	18	23	21	21	21	21	263
Hour Total	37	39	39	44	39	39	39	39	39	47	42	42	42	41	568
TOTAL		159				195					214				568
Fraction		0.641129032				0.629032258					0.690322581				
April A	17	17	17	18	15	15	15	15	15	17	17	17	17	17	229
April B	21	21	21	21	18	18	18	18	19	22	19	19	19	19	273
Hour Total	38	38	38	39	33	33	33	33	34	39	36	36	36	36	502
TOTAL		153				166					183				502
Fraction		0.6375				0.553333333					0.61				
May A	10	11	11	15	10	11	11	11	11	14	11	11	11	11	159
May B	17	17	17	17	14	14	14	14	14	17	14	14	14	14	211
Hour Total	27	28	28	32	24	25	25	25	25	31	25	25	25	25	370
TOTAL		115				124					131				370
Fraction		0.463709677				0.4					0.422580645				

BOS to CLE Unreliability Measures for Monthly Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	5	5	5	7	5	5	5	5	5	9	9	9	9	9	92
November B	22	22	22	24	23	23	23	23	23	24	23	23	23	23	321
Hour Total	27	27	27	31	28	28	28	28	28	33	32	32	32	32	413
TOTAL	112				140					161					413
Fraction	0.466666667				0.466666667					0.536666667					
December A	15	15	15	20	18	18	18	18	18	20	16	17	17	17	242
December B	22	22	22	24	21	21	21	21	21	23	20	20	20	20	298
Hour Total	37	37	37	44	39	39	39	39	39	43	36	37	37	37	540
TOTAL	155				195					190					540
Fraction	0.625				0.629032258					0.612903226					
January A	20	21	21	23	20	20	20	20	20	20	18	18	18	18	277
January B	24	24	24	25	24	24	24	24	24	26	23	23	23	23	335
Hour Total	44	45	45	48	44	44	44	44	44	46	41	41	41	41	612
TOTAL	182				220					210					612
Fraction	0.733870968				0.709677419					0.677419355					
February A	17	17	17	19	15	15	15	15	15	17	14	14	14	14	218
February B	19	19	19	21	16	16	16	16	16	20	18	18	18	17	249
Hour Total	36	36	36	40	31	31	31	31	31	37	32	32	32	31	467
TOTAL	148				155					164					467
Fraction	0.660714286				0.553571429					0.585714286					
March A	19	21	21	24	19	19	19	19	19	24	21	21	21	20	287
March B	16	16	16	18	17	17	17	17	17	22	21	21	21	21	257
Hour Total	35	37	37	42	36	36	36	36	36	46	42	42	42	41	544
TOTAL	151				180					213					544
Fraction	0.608870968				0.580645161					0.687096774					
April A	15	15	15	17	15	15	15	15	15	16	15	15	15	15	213
April B	17	17	17	17	14	14	14	14	14	18	15	15	15	15	216
Hour Total	32	32	32	34	29	29	29	29	29	34	30	30	30	30	429
TOTAL	130				145					154					429
Fraction	0.541666667				0.483333333					0.513333333					
May A	9	10	10	12	6	7	7	7	7	8	7	7	7	7	111
May B	8	8	8	11	8	8	8	8	9	11	8	8	8	8	119
Hour Total	17	18	18	23	14	15	15	15	16	19	15	15	15	15	230
TOTAL	76				75					79					230
Fraction	0.306451613				0.241935484					0.25483871					

BOS to CLE Unreliability Measures for Monthly Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	4	4	4	5	4	4	4	4	4	7	7	7	7	7	72
November B	22	22	22	24	23	23	23	23	23	24	23	23	23	23	321
Hour Total	26	26	26	29	27	27	27	27	27	31	30	30	30	30	393
TOTAL	107				135					151					393
Fraction	0.445833333				0.45					0.503333333					
December A	15	15	15	19	17	17	17	17	17	18	14	15	15	15	226
December B	19	19	19	23	20	20	20	20	20	23	20	20	20	20	283
Hour Total	34	34	34	42	37	37	37	37	37	41	34	35	35	35	509
TOTAL	144				185					180					509
Fraction	0.580645161				0.596774194					0.580645161					
January A	18	19	19	23	20	20	20	20	20	20	18	18	18	18	271
January B	23	23	23	25	24	24	24	24	24	26	23	23	23	23	332
Hour Total	41	42	42	48	44	44	44	44	44	46	41	41	41	41	603
TOTAL	173				220					210					603
Fraction	0.697580645				0.709677419					0.677419355					
February A	17	17	17	19	15	15	15	15	15	17	14	14	14	14	218
February B	16	16	16	19	15	15	15	15	15	19	16	16	16	16	225
Hour Total	33	33	33	38	30	30	30	30	30	36	30	30	30	30	443
TOTAL	137				150					156					443
Fraction	0.611607143				0.535714286					0.557142857					
March A	18	20	20	24	19	19	19	19	19	23	20	20	20	19	279
March B	12	12	12	15	14	14	14	14	14	20	19	19	19	19	217
Hour Total	30	32	32	39	33	33	33	33	33	43	39	39	39	38	496
TOTAL	133				165					198					496
Fraction	0.536290323				0.532258065					0.638709677					
April A	14	14	14	15	12	12	12	12	12	13	13	13	13	13	182
April B	17	17	17	16	13	13	13	13	13	16	12	12	12	12	196
Hour Total	31	31	31	31	25	25	25	25	25	29	25	25	25	25	378
TOTAL	124				125					129					378
Fraction	0.516666667				0.416666667					0.43					
May A	8	9	9	11	5	6	6	6	6	7	7	7	7	7	101
May B	4	4	4	5	4	4	4	4	5	7	6	6	6	6	69
Hour Total	12	13	13	16	9	10	10	10	11	14	13	13	13	13	170
TOTAL	54				50					66					170
Fraction	0.217741935				0.161290323					0.212903226					

BOS to IAD Unreliability Measures for Monthly Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	7	7	7	10	7	7	7	7	7	8	6	6	6	6	98
November B	16	17	17	21	18	18	18	18	18	21	18	18	18	18	254
Hour Total	23	24	24	31	25	25	25	25	25	29	24	24	24	24	352
TOTAL	102				125					125					352
Fraction	0.425				0.416666667					0.416666667					
December A	12	12	12	15	13	13	13	13	13	18	15	16	16	16	197
December B	19	19	19	19	15	15	15	15	15	19	15	15	15	15	230
Hour Total	31	31	31	34	28	28	28	28	28	37	30	31	31	31	427
TOTAL	127				140					160					427
Fraction	0.512096774				0.451612903					0.516129032					
January A	19	20	20	22	18	18	18	18	18	22	20	20	20	19	272
January B	23	23	23	24	22	22	22	22	22	27	19	19	19	19	306
Hour Total	42	43	43	46	40	40	40	40	40	49	39	39	39	38	578
TOTAL	174				200					204					578
Fraction	0.701612903				0.64516129					0.658064516					
February A	10	10	10	13	11	11	11	11	11	14	11	11	11	11	156
February B	16	16	16	21	21	21	21	21	21	23	21	21	21	20	280
Hour Total	26	26	26	34	32	32	32	32	32	37	32	32	32	31	436
TOTAL	112				160					164					436
Fraction	0.5				0.571428571					0.585714286					
March A	16	16	16	17	15	15	15	15	15	19	17	17	17	16	226
March B	10	10	10	14	13	13	13	13	13	15	12	12	13	13	174
Hour Total	26	26	26	31	28	28	28	28	28	34	29	29	30	29	400
TOTAL	109				140					151					400
Fraction	0.439516129				0.451612903					0.487096774					
April A	12	12	12	16	11	11	11	11	12	18	17	17	17	17	194
April B	17	17	17	21	19	19	19	19	19	21	18	18	18	18	194
Hour Total	29	29	29	37	30	30	30	30	31	39	35	35	35	35	388
TOTAL	124				151					179					454
Fraction	0.516666667				0.503333333					0.596666667					
May A	9	10	10	11	9	9	9	9	9	11	10	11	11	11	139
May B	16	17	17	20	17	17	17	17	17	19	14	14	14	14	230
Hour Total	25	27	27	31	26	26	26	26	26	30	24	25	25	25	369
TOTAL	110				130					129					369
Fraction	0.443548387				0.419354839					0.416129032					

BOS to IAD Unreliability Measures for Monthly Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	7	7	7	10	6	6	6	6	6	7	6	6	6	6	92
November B	16	17	17	21	17	17	17	17	17	19	16	16	16	16	239
Hour Total	23	24	24	31	23	23	23	23	23	26	22	22	22	22	331
TOTAL	102				115					114					331
Fraction	0.425				0.383333333					0.38					
December A	12	12	12	15	13	13	13	13	13	18	15	16	16	16	197
December B	18	18	18	19	14	14	14	14	14	19	15	15	15	15	222
Hour Total	30	30	30	34	27	27	27	27	27	37	30	31	31	31	419
TOTAL	124				135					160					419
Fraction	0.5				0.435483871					0.516129032					
January A	19	20	20	22	18	18	18	18	18	22	19	19	19	18	268
January B	23	23	23	24	22	22	22	22	22	27	19	19	19	19	306
Hour Total	42	43	43	46	40	40	40	40	40	49	38	38	38	37	574
TOTAL	174				200					200					574
Fraction	0.701612903				0.64516129					0.64516129					
February A	10	10	10	13	11	11	11	11	11	14	11	11	11	11	156
February B	16	16	16	21	21	21	21	21	21	23	21	21	21	20	280
Hour Total	26	26	26	34	32	32	32	32	32	37	32	32	32	31	436
TOTAL	112				160					164					436
Fraction	0.5				0.571428571					0.585714286					
March A	16	16	16	17	15	15	15	15	15	19	17	17	17	16	226
March B	10	10	10	14	13	13	13	13	13	15	12	12	13	13	174
Hour Total	26	26	26	31	28	28	28	28	28	34	29	29	30	29	400
TOTAL	109				140					151					400
Fraction	0.439516129				0.451612903					0.487096774					
April A	11	11	11	14	10	10	10	10	11	16	15	15	15	15	174
April B	17	17	17	20	18	18	18	18	18	21	18	18	18	18	174
Hour Total	28	28	28	34	28	28	28	28	29	37	33	33	33	33	348
TOTAL	118				141					169					428
Fraction	0.491666667				0.47					0.563333333					
May A	8	9	9	11	8	8	8	8	8	9	7	8	8	8	117
May B	15	16	16	19	15	15	15	15	15	17	13	13	13	13	210
Hour Total	23	25	25	30	23	23	23	23	23	26	20	21	21	21	327
TOTAL	103				115					109					327
Fraction	0.415322581				0.370967742					0.351612903					

BOS to IAD Unreliability Measures for Monthly Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	5	5	5	6	4	4	4	4	4	6	6	6	6	6	71
November B	12	12	12	17	16	16	16	16	16	19	16	16	16	16	216
Hour Total	17	17	17	23	20	20	20	20	20	25	22	22	22	22	287
TOTAL	74				100					113					287
Fraction	0.308333333				0.333333333					0.376666667					
December A	11	11	11	14	11	11	11	11	11	15	12	14	14	14	171
December B	18	18	18	19	13	13	13	13	13	16	13	13	13	13	206
Hour Total	29	29	29	33	24	24	24	24	24	31	25	27	27	27	377
TOTAL	120				120					137					377
Fraction	0.483870968				0.387096774					0.441935484					
January A	19	20	20	21	17	17	17	17	17	19	16	16	16	16	248
January B	21	21	21	23	21	21	21	21	21	26	19	19	19	19	293
Hour Total	40	41	41	44	38	38	38	38	38	45	35	35	35	35	541
TOTAL	166				190					185					541
Fraction	0.669354839				0.612903226					0.596774194					
February A	9	9	9	12	10	10	10	10	10	13	10	10	10	10	142
February B	16	16	16	21	20	20	20	20	20	22	20	20	20	19	270
Hour Total	25	25	25	33	30	30	30	30	30	35	30	30	30	29	412
TOTAL	108				150					154					412
Fraction	0.482142857				0.535714286					0.55					
March A	15	15	15	16	14	14	14	14	14	19	17	17	17	16	217
March B	10	10	10	13	10	10	10	10	10	13	11	11	12	12	152
Hour Total	25	25	25	29	24	24	24	24	24	32	28	28	29	28	369
TOTAL	104				120					145					369
Fraction	0.419354839				0.387096774					0.467741935					
April A	10	10	10	13	10	10	10	10	11	15	15	15	15	15	169
April B	16	16	16	18	15	15	15	15	15	17	14	14	14	14	169
Hour Total	26	26	26	31	25	25	25	25	26	32	29	29	29	29	338
TOTAL	109				126					148					383
Fraction	0.454166667				0.42					0.493333333					
May A	7	8	8	9	5	5	5	5	5	6	5	6	6	6	86
May B	7	7	7	10	7	7	7	7	8	8	5	5	5	5	95
Hour Total	14	15	15	19	12	12	12	12	13	14	10	11	11	11	181
TOTAL	63				61					57					181
Fraction	0.254032258				0.196774194					0.183870968					

BOS to IAD Unreliability Measures for Monthly Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	4	4	4	5	4	4	4	4	4	4	4	4	4	4	57
November B	12	12	12	17	15	15	16	16	16	19	15	15	15	15	210
Hour Total	16	16	16	22	19	19	20	20	20	23	19	19	19	19	267
TOTAL	70				98					99					267
Fraction	0.291666667				0.326666667					0.33					
December A	10	10	10	12	8	8	8	8	8	12	10	11	11	11	137
December B	16	16	16	18	12	12	12	12	12	15	12	12	12	12	189
Hour Total	26	26	26	30	20	20	20	20	20	27	22	23	23	23	326
TOTAL	108				100					118					326
Fraction	0.435483871				0.322580645					0.380645161					
January A	17	18	18	21	17	17	17	17	17	18	15	15	15	15	237
January B	20	20	20	23	21	21	21	21	21	26	19	19	19	19	290
Hour Total	37	38	38	44	38	38	38	38	38	44	34	34	34	34	527
TOTAL	157				190					180					527
Fraction	0.633064516				0.612903226					0.580645161					
February A	9	9	9	12	10	10	10	10	10	13	10	10	10	10	142
February B	14	14	14	19	18	18	18	18	18	19	17	17	17	17	238
Hour Total	23	23	23	31	28	28	28	28	28	32	27	27	27	27	380
TOTAL	100				140					140					380
Fraction	0.446428571				0.5					0.5					
March A	14	14	14	16	14	14	14	14	14	18	14	14	14	13	201
March B	9	9	9	11	8	8	8	8	8	11	10	10	11	11	131
Hour Total	23	23	23	27	22	22	22	22	22	29	24	24	25	24	332
TOTAL	96				110					126					332
Fraction	0.387096774				0.35483871					0.406451613					
April A	10	10	10	11	8	8	8	8	9	13	13	13	13	13	147
April B	15	15	15	15	12	12	12	12	12	15	11	11	11	11	147
Hour Total	25	25	25	26	20	20	20	20	21	28	24	24	24	24	294
TOTAL	101				101					124					326
Fraction	0.420833333				0.336666667					0.413333333					
May A	6	7	7	7	3	3	3	3	3	4	4	5	5	5	65
May B	3	3	3	4	3	3	3	3	4	4	4	4	4	4	49
Hour Total	9	10	10	11	6	6	6	6	7	8	8	9	9	9	114
TOTAL	40				31					43					114
Fraction	0.161290323				0.1					0.138709677					

CLE to IAD Unreliability Measures for Monthly Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	6	6	6	7	6	6	6	6	6	7	6	6	6	6	86
November B	24	24	24	25	20	20	20	20	20	24	23	23	23	23	313
Hour Total	30	30	30	32	26	26	26	26	26	31	29	29	29	29	399
TOTAL	122				130					147					399
Fraction	0.508333333				0.433333333					0.49					
December A	17	17	17	23	21	21	21	21	21	22	17	18	18	18	272
December B	21	21	21	24	23	23	23	23	23	24	21	21	21	21	310
Hour Total	38	38	38	47	44	44	44	44	44	46	38	39	39	39	582
TOTAL	161				220					201					582
Fraction	0.649193548				0.709677419					0.648387097					
January A	19	19	19	23	19	19	19	19	19	21	19	19	19	19	272
January B	21	21	21	22	20	20	20	20	20	27	24	24	24	24	308
Hour Total	40	40	40	45	39	39	39	39	39	48	43	43	43	43	580
TOTAL	165				195					220					580
Fraction	0.665322581				0.629032258					0.709677419					
February A	14	14	14	17	15	15	15	15	15	16	12	12	12	12	198
February B	19	19	19	22	21	21	22	22	22	24	23	23	23	22	302
Hour Total	33	33	33	39	36	36	37	37	37	40	35	35	35	34	500
TOTAL	138				183					179					500
Fraction	0.616071429				0.653571429					0.639285714					
March A	19	20	20	22	18	18	18	18	18	21	20	20	20	19	271
March B	15	15	15	18	18	18	18	18	18	22	21	21	21	21	259
Hour Total	34	35	35	40	36	36	36	36	36	43	41	41	41	40	530
TOTAL	144				180					206					530
Fraction	0.580645161				0.580645161					0.664516129					
April A	16	16	16	19	14	14	15	15	15	17	14	15	15	15	216
April B	14	14	14	15	14	14	14	14	15	16	13	13	13	13	216
Hour Total	30	30	30	34	28	28	29	29	30	33	27	28	28	28	432
TOTAL	124				144					144					412
Fraction	0.516666667				0.48					0.48					
May A	8	8	8	13	11	11	12	12	12	15	11	12	12	12	157
May B	16	16	16	17	12	12	12	12	13	17	10	10	11	11	185
Hour Total	24	24	24	30	23	23	24	24	25	32	21	22	23	23	342
TOTAL	102				119					121					342
Fraction	0.411290323				0.383870968					0.390322581					

CLE to IAD Unreliability Measures for Monthly Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	5	5	5	7	5	5	5	5	5	7	6	6	6	6	78
November B	24	24	24	25	19	19	19	19	19	23	22	22	22	22	303
Hour Total	29	29	29	32	24	24	24	24	24	30	28	28	28	28	381
TOTAL	119				120					142					381
Fraction	0.495833333				0.4					0.473333333					
December A	16	16	16	23	20	20	20	20	20	21	16	17	17	17	259
December B	20	20	20	24	22	22	22	22	22	24	21	21	21	21	302
Hour Total	36	36	36	47	42	42	42	42	42	45	37	38	38	38	561
TOTAL	155				210					196					561
Fraction	0.625				0.677419355					0.632258065					
January A	19	19	19	22	18	18	18	18	18	21	18	18	18	18	262
January B	21	21	21	22	20	20	20	20	20	27	24	24	24	24	308
Hour Total	40	40	40	44	38	38	38	38	38	48	42	42	42	42	570
TOTAL	164				190					216					570
Fraction	0.661290323				0.612903226					0.696774194					
February A	13	13	13	17	15	15	15	15	15	16	12	12	12	12	195
February B	19	19	19	22	21	21	22	22	22	24	23	23	23	22	302
Hour Total	32	32	32	39	36	36	37	37	37	40	35	35	35	34	497
TOTAL	135				183					179					497
Fraction	0.602678571				0.653571429					0.639285714					
March A	19	20	20	22	18	18	18	18	18	21	20	20	20	19	271
March B	15	15	15	18	18	18	18	18	18	22	21	21	21	21	259
Hour Total	34	35	35	40	36	36	36	36	36	43	41	41	41	40	530
TOTAL	144				180					206					530
Fraction	0.580645161				0.580645161					0.664516129					
April A	15	15	15	16	11	11	12	12	12	14	13	13	13	13	185
April B	14	14	14	15	14	14	14	14	15	16	13	13	13	13	185
Hour Total	29	29	29	31	25	25	26	26	27	30	26	26	26	26	370
TOTAL	118				129					134					381
Fraction	0.491666667				0.43					0.446666667					
May A	7	7	7	9	7	7	8	8	8	11	8	9	9	9	114
May B	14	14	14	15	10	10	10	10	11	15	10	10	10	10	163
Hour Total	21	21	21	24	17	17	18	18	19	26	18	19	19	19	277
TOTAL	87				89					101					277
Fraction	0.350806452				0.287096774					0.325806452					

CLE to IAD Unreliability Measures for Monthly Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	1	1	1	2	2	2	2	2	2	4	4	4	4	4	35
November B	20	20	20	23	18	18	18	18	18	23	22	22	22	22	284
Hour Total	21	21	21	25	20	20	20	20	20	27	26	26	26	26	319
TOTAL	88				100					131					319
Fraction	0.366666667				0.333333333					0.436666667					
December A	15	15	15	21	18	18	18	18	18	19	13	14	14	14	230
December B	20	20	20	24	20	20	20	20	20	21	18	18	18	18	277
Hour Total	35	35	35	45	38	38	38	38	38	40	31	32	32	32	507
TOTAL	150				190					167					507
Fraction	0.60483871				0.612903226					0.538709677					
January A	18	18	18	21	16	16	16	16	16	19	17	17	17	17	242
January B	20	20	20	22	20	20	20	20	20	27	24	24	24	24	305
Hour Total	38	38	38	43	36	36	36	36	36	46	41	41	41	41	547
TOTAL	157				180					210					547
Fraction	0.633064516				0.580645161					0.677419355					
February A	12	12	12	15	14	14	14	14	14	15	11	11	11	11	180
February B	19	19	19	22	21	21	22	22	22	24	23	23	23	22	302
Hour Total	31	31	31	37	35	35	36	36	36	39	34	34	34	33	482
TOTAL	130				178					174					482
Fraction	0.580357143				0.635714286					0.621428571					
March A	17	18	18	20	16	16	16	16	16	21	20	20	20	19	253
March B	15	15	15	17	17	17	17	17	17	22	21	21	21	21	253
Hour Total	32	33	33	37	33	33	33	33	33	43	41	41	41	40	506
TOTAL	135				165					206					506
Fraction	0.544354839				0.532258065					0.664516129					
April A	13	13	13	16	11	11	12	12	12	14	12	12	12	12	175
April B	10	10	10	11	10	10	10	10	10	11	9	9	9	9	175
Hour Total	23	23	23	27	21	21	22	22	22	25	21	21	21	21	350
TOTAL	96				108					109					313
Fraction	0.4				0.36					0.363333333					
May A	6	6	6	8	4	4	5	5	5	7	6	7	7	7	83
May B	5	5	5	8	6	6	6	6	7	10	7	7	7	7	92
Hour Total	11	11	11	16	10	10	11	11	12	17	13	14	14	14	175
TOTAL	49				54					72					175
Fraction	0.197580645				0.174193548					0.232258065					

CLE to IAD Unreliability Measures for Monthly Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
November A	1	1	1	1	1	1	1	1	1	4	4	4	4	4	29
November B	20	20	20	23	18	18	18	18	18	23	22	22	22	22	284
Hour Total	21	21	21	24	19	19	19	19	19	27	26	26	26	26	313
TOTAL	87				95					131					313
Fraction	0.3625				0.316666667					0.436666667					
December A	14	14	14	17	15	15	15	15	15	16	10	11	11	11	193
December B	15	15	15	21	19	19	19	19	19	21	17	17	17	17	250
Hour Total	29	29	29	38	34	34	34	34	34	37	27	28	28	28	443
TOTAL	125				170					148					443
Fraction	0.504032258				0.548387097					0.477419355					
January A	16	16	16	21	16	16	16	16	16	18	16	16	16	16	231
January B	19	19	19	22	20	20	20	20	20	27	24	24	24	24	302
Hour Total	35	35	35	43	36	36	36	36	36	45	40	40	40	40	533
TOTAL	148				180					205					533
Fraction	0.596774194				0.580645161					0.661290323					
February A	12	12	12	15	14	14	14	14	14	15	11	11	11	11	180
February B	18	18	18	21	19	19	20	20	20	22	21	21	21	21	279
Hour Total	30	30	30	36	33	33	34	34	34	37	32	32	32	32	459
TOTAL	126				168					165					459
Fraction	0.5625				0.6					0.589285714					
March A	15	16	16	19	14	14	14	14	14	19	18	18	18	17	226
March B	11	11	11	14	14	14	14	14	14	20	19	19	19	19	213
Hour Total	26	27	27	33	28	28	28	28	28	39	37	37	37	36	439
TOTAL	113				140					186					439
Fraction	0.455645161				0.451612903					0.6					
April A	11	11	11	13	8	8	9	9	9	11	10	10	10	10	140
April B	9	9	9	9	6	6	6	6	6	7	4	4	4	4	140
Hour Total	20	20	20	22	14	14	15	15	15	18	14	14	14	14	280
TOTAL	82				73					74					229
Fraction	0.341666667				0.243333333					0.246666667					
May A	6	6	6	7	2	2	3	3	3	5	5	6	6	6	66
May B	2	2	2	2	2	2	2	2	3	5	3	3	3	3	36
Hour Total	8	8	8	9	4	4	5	5	6	10	8	9	9	9	102
TOTAL	33				24					45					102
Fraction	0.133064516				0.077419355					0.14516129					

BOS to CLE Unreliability Measures for 3-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	7	7	7	8	6	6	6	6	6	8	7	7	7	7	95
Nov 1-21 b	16	16	16	18	17	17	17	17	17	18	17	17	17	17	237
Hour Total	23	23	23	26	23	23	23	23	23	26	24	24	24	24	332
TOTAL		95				115					122				332
Fraction		0.56547619				0.547619048					0.580952381				
Nov22-Dec12 a	5	5	5	11	9	9	9	9	9	11	9	9	9	9	118
Nov22-Dec12 b	15	15	15	15	15	15	15	15	15	16	14	14	14	14	207
Hour Total	20	20	20	26	24	24	24	24	24	27	23	23	23	23	325
TOTAL		86				120					119				325
Fraction		0.511904762				0.571428571					0.566666667				
Dec13-Jan3 a	16	16	16	18	17	17	17	17	17	18	15	16	16	16	232
Dec13-Jan3 b	18	18	18	21	20	20	20	20	20	20	19	19	19	19	271
Hour Total	34	34	34	39	37	37	37	37	37	38	34	35	35	35	503
TOTAL		141				185					177				503
Fraction		0.839285714				0.880952381					0.842857143				
Jan 4-24 a	17	18	18	19	18	18	18	18	18	18	16	16	16	15	243
Jan 4-24 b	14	14	14	15	15	15	15	15	15	16	13	13	13	13	200
Hour Total	31	32	32	34	33	33	33	33	33	34	29	29	29	28	443
TOTAL		129				165					149				443
Fraction		0.767857143				0.785714286					0.70952381				
Jan25-Feb14 a	11	11	11	14	11	11	11	11	11	12	9	9	9	9	150
Jan25-Feb14 b	16	16	16	17	16	16	16	16	16	18	16	16	16	15	226
Hour Total	27	27	27	31	27	27	27	27	27	30	25	25	25	24	376
TOTAL		112				135					129				376
Fraction		0.666666667				0.642857143					0.614285714				
Feb15-Mar7 a	16	17	17	18	13	13	13	13	13	15	13	13	13	13	200
Feb15-Mar7 b	15	15	15	16	11	11	11	11	11	16	16	16	16	16	196
Hour Total	31	32	32	34	24	24	24	24	24	31	29	29	29	29	396
TOTAL		129				120					147				396
Fraction		0.767857143				0.571428571					0.7				
Mar 8-28 a	13	14	14	16	14	14	14	14	14	16	15	15	15	14	202
Mar 8-28 b	8	8	8	10	10	10	10	10	10	13	11	11	11	11	141
Hour Total	21	22	22	26	24	24	24	24	24	29	26	26	26	25	343
TOTAL		91				120					132				343
Fraction		0.541666667				0.571428571					0.628571429				
Mar29-Apr18 a	14	14	14	16	11	11	11	11	11	12	12	12	12	12	173
Mar29-Apr18 b	15	15	15	15	14	14	14	14	15	17	16	16	16	16	212
Hour Total	29	29	29	31	25	25	25	25	26	29	28	28	28	28	385
TOTAL		118				126					141				385
Fraction		0.702380952				0.6					0.671428571				
Apr19-May9 a	11	12	12	14	12	12	12	12	12	14	12	12	12	12	171
Apr19-May9 b	13	13	13	13	10	10	10	10	10	12	9	9	9	9	150
Hour Total	24	25	25	27	22	22	22	22	22	26	21	21	21	21	321
TOTAL		101				110					110				321
Fraction		0.601190476				0.523809524					0.523809524				
May 10-31 a	8	8	8	12	9	10	10	10	10	12	12	12	12	12	145
May 10-31 b	14	14	14	14	13	13	13	13	13	15	12	12	12	12	184
Hour Total	22	22	22	26	22	23	23	23	23	27	24	24	24	24	329
TOTAL		92				114					123				329
Fraction		0.522727273				0.518181818					0.559090909				

BOS to CLE Unreliability Measures for 3-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	7	7	7	8	6	6	6	6	6	8	7	7	7	7	95
Nov 1-21 b	16	16	16	18	15	15	15	15	15	16	15	15	15	15	217
Hour Total	23	23	23	26	21	21	21	21	21	24	22	22	22	22	312
TOTAL	95				105					112					312
Fraction	0.56547619				0.5					0.533333333					
Nov22-Dec12 a	4	4	4	9	7	7	7	7	7	10	9	9	9	9	102
Nov22-Dec12 b	15	15	15	15	15	15	15	15	15	16	14	14	14	14	207
Hour Total	19	19	19	24	22	22	22	22	22	26	23	23	23	23	309
TOTAL	81				110					118					309
Fraction	0.482142857				0.523809524					0.561904762					
Dec13-Jan3 a	15	15	15	18	17	17	17	17	17	17	14	15	15	15	224
Dec13-Jan3 b	18	18	18	20	19	19	19	19	19	20	19	19	19	19	265
Hour Total	33	33	33	38	36	36	36	36	36	37	33	34	34	34	489
TOTAL	137				180					172					489
Fraction	0.81547619				0.857142857					0.819047619					
Jan 4-24 a	17	18	18	19	18	18	18	18	18	18	16	16	16	15	243
Jan 4-24 b	14	14	14	15	15	15	15	15	15	16	13	13	13	13	200
Hour Total	31	32	32	34	33	33	33	33	33	34	29	29	29	28	443
TOTAL	129				165					149					443
Fraction	0.767857143				0.785714286					0.70952381					
Jan25-Feb14 a	11	11	11	13	10	10	10	10	10	12	9	9	9	9	144
Jan25-Feb14 b	16	16	16	17	16	16	16	16	16	18	16	16	16	15	226
Hour Total	27	27	27	30	26	26	26	26	26	30	25	25	25	24	370
TOTAL	111				130					129					370
Fraction	0.660714286				0.619047619					0.614285714					
Feb15-Mar7 a	16	17	17	18	13	13	13	13	13	15	13	13	13	13	200
Feb15-Mar7 b	15	15	15	16	11	11	11	11	11	16	16	16	16	16	196
Hour Total	31	32	32	34	24	24	24	24	24	31	29	29	29	29	396
TOTAL	129				120					147					396
Fraction	0.767857143				0.571428571					0.7					
Mar 8-28 a	13	14	14	16	14	14	14	14	14	16	15	15	15	14	202
Mar 8-28 b	8	8	8	10	10	10	10	10	10	13	11	11	11	11	141
Hour Total	21	22	22	26	24	24	24	24	24	29	26	26	26	25	343
TOTAL	91				120					132					343
Fraction	0.541666667				0.571428571					0.628571429					
Mar29-Apr18 a	12	12	12	13	9	9	9	9	9	10	10	10	10	10	144
Mar29-Apr18 b	15	15	15	15	14	14	14	14	15	17	16	16	16	16	212
Hour Total	27	27	27	28	23	23	23	23	24	27	26	26	26	26	356
TOTAL	109				116					131					356
Fraction	0.648809524				0.552380952					0.623809524					
Apr19-May9 a	10	11	11	14	12	12	12	12	12	14	11	11	11	11	164
Apr19-May9 b	13	13	13	13	10	10	10	10	10	12	9	9	9	9	150
Hour Total	23	24	24	27	22	22	22	22	22	26	20	20	20	20	314
TOTAL	98				110					106					314
Fraction	0.583333333				0.523809524					0.504761905					
May 10-31 a	7	7	7	9	6	7	7	7	7	9	9	9	9	9	109
May 10-31 b	13	13	13	12	11	11	11	11	11	13	11	11	11	11	163
Hour Total	20	20	20	21	17	18	18	18	18	22	20	20	20	20	272
TOTAL	81				89					102					272
Fraction	0.460227273				0.404545455					0.463636364					

BOS to CLE Unreliability Measures for 3-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	4	4	4	6	5	5	5	5	5	6	6	6	6	6	73
Nov 1-21 b	14	14	14	16	15	15	15	15	15	16	15	15	15	15	209
Hour Total	18	18	18	22	20	20	20	20	20	22	21	21	21	21	282
TOTAL	76				100					106					282
Fraction	0.452380952				0.476190476					0.504761905					
Nov22-Dec12 a	3	3	3	5	3	3	3	3	3	8	8	8	8	8	69
Nov22-Dec12 b	15	15	15	15	15	15	15	15	15	16	14	14	14	14	207
Hour Total	18	18	18	20	18	18	18	18	18	24	22	22	22	22	276
TOTAL	74				90					112					276
Fraction	0.44047619				0.428571429					0.533333333					
Dec13-Jan3 a	14	14	14	17	16	16	16	16	16	16	12	13	13	13	206
Dec13-Jan3 b	18	18	18	20	17	17	17	17	17	18	17	17	17	17	245
Hour Total	32	32	32	37	33	33	33	33	33	34	29	30	30	30	451
TOTAL	133				165					153					451
Fraction	0.791666667				0.785714286					0.728571429					
Jan 4-24 a	16	17	17	19	17	17	17	17	17	17	15	15	15	15	231
Jan 4-24 b	14	14	14	15	15	15	15	15	15	16	13	13	13	13	200
Hour Total	30	31	31	34	32	32	32	32	32	33	28	28	28	28	431
TOTAL	126				160					145					431
Fraction	0.75				0.761904762					0.69047619					
Jan25-Feb14 a	11	11	11	13	10	10	10	10	10	11	8	8	8	8	139
Jan25-Feb14 b	16	16	16	17	16	16	16	16	16	18	16	16	16	15	226
Hour Total	27	27	27	30	26	26	26	26	26	29	24	24	24	23	365
TOTAL	111				130					124					365
Fraction	0.660714286				0.619047619					0.59047619					
Feb15-Mar7 a	15	16	16	16	12	12	12	12	12	14	12	12	12	12	185
Feb15-Mar7 b	15	15	15	16	11	11	11	11	11	16	16	16	16	16	196
Hour Total	30	31	31	32	23	23	23	23	23	30	28	28	28	28	381
TOTAL	124				115					142					381
Fraction	0.738095238				0.547619048					0.676190476					
Mar 8-28 a	11	12	12	14	12	12	12	12	12	16	15	15	15	14	184
Mar 8-28 b	8	8	8	10	9	9	9	9	9	12	11	11	11	11	135
Hour Total	19	20	20	24	21	21	21	21	21	28	26	26	26	25	319
TOTAL	83				105					131					319
Fraction	0.494047619				0.5					0.623809524					
Mar29-Apr18 a	11	11	11	12	9	9	9	9	9	9	9	9	9	9	135
Mar29-Apr18 b	15	15	15	14	12	12	12	12	12	14	13	13	13	13	185
Hour Total	26	26	26	26	21	21	21	21	21	23	22	22	22	22	320
TOTAL	104				105					111					320
Fraction	0.619047619				0.5					0.528571429					
Apr19-May9 a	9	10	10	12	9	9	9	9	9	10	8	8	8	8	128
Apr19-May9 b	6	6	6	8	6	6	6	6	6	8	5	5	5	5	84
Hour Total	15	16	16	20	15	15	15	15	15	18	13	13	13	13	212
TOTAL	67				75					70					212
Fraction	0.398809524				0.357142857					0.333333333					
May 10-31 a	6	6	6	8	5	6	6	6	6	7	7	7	7	7	90
May 10-31 b	7	7	7	9	7	7	7	7	8	10	8	8	8	8	108
Hour Total	13	13	13	17	12	13	13	13	14	17	15	15	15	15	198
TOTAL	56				65					77					198
Fraction	0.318181818				0.295454545					0.35					

BOS to CLE Unreliability Measures for 3-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	4	4	4	5	4	4	4	4	4	6	6	6	6	6	67
Nov 1-21 b	14	14	14	16	15	15	15	15	15	16	15	15	15	15	209
Hour Total	18	18	18	21	19	19	19	19	19	22	21	21	21	21	276
TOTAL	75				95					106					276
Fraction	0.446428571				0.452380952					0.504761905					
Nov22-Dec12 a	2	2	2	3	2	2	2	2	2	4	4	4	4	4	39
Nov22-Dec12 b	15	15	15	15	15	15	15	15	15	16	14	14	14	14	207
Hour Total	17	17	17	18	17	17	17	17	17	20	18	18	18	18	246
TOTAL	69				85					92					246
Fraction	0.410714286				0.404761905					0.438095238					
Dec13-Jan3 a	14	14	14	17	16	16	16	16	16	16	12	13	13	13	206
Dec13-Jan3 b	14	14	14	19	16	16	16	16	16	18	17	17	17	17	227
Hour Total	28	28	28	36	32	32	32	32	32	34	29	30	30	30	433
TOTAL	120				160					153					433
Fraction	0.714285714				0.761904762					0.728571429					
Jan 4-24 a	14	15	15	19	17	17	17	17	17	17	15	15	15	15	225
Jan 4-24 b	14	14	14	15	15	15	15	15	15	16	13	13	13	13	200
Hour Total	28	29	29	34	32	32	32	32	32	33	28	28	28	28	425
TOTAL	120				160					145					425
Fraction	0.714285714				0.761904762					0.69047619					
Jan25-Feb14 a	11	11	11	13	10	10	10	10	10	11	8	8	8	8	139
Jan25-Feb14 b	14	14	14	17	16	16	16	16	16	18	15	15	15	15	217
Hour Total	25	25	25	30	26	26	26	26	26	29	23	23	23	23	356
TOTAL	105				130					121					356
Fraction	0.625				0.619047619					0.576190476					
Feb15-Mar7 a	14	15	15	16	12	12	12	12	12	14	12	12	12	12	182
Feb15-Mar7 b	14	14	14	14	10	10	10	10	10	15	15	15	15	15	181
Hour Total	28	29	29	30	22	22	22	22	22	29	27	27	27	27	363
TOTAL	116				110					137					363
Fraction	0.69047619				0.523809524					0.652380952					
Mar 8-28 a	11	12	12	14	12	12	12	12	12	15	14	14	14	13	179
Mar 8-28 b	4	4	4	7	6	6	6	6	6	10	9	9	9	9	95
Hour Total	15	16	16	21	18	18	18	18	18	25	23	23	23	22	274
TOTAL	68				90					116					274
Fraction	0.404761905				0.428571429					0.552380952					
Mar29-Apr18 a	10	10	10	11	7	7	7	7	7	7	7	7	7	7	111
Mar29-Apr18 b	15	15	15	14	12	12	12	12	12	14	11	11	11	11	177
Hour Total	25	25	25	25	19	19	19	19	19	21	18	18	18	18	288
TOTAL	100				95					93					288
Fraction	0.595238095				0.452380952					0.442857143					
Apr19-May9 a	8	9	9	10	7	7	7	7	7	8	8	8	8	8	111
Apr19-May9 b	6	6	6	7	5	5	5	5	5	6	4	4	4	4	72
Hour Total	14	15	15	17	12	12	12	12	12	14	12	12	12	12	183
TOTAL	61				60					62					183
Fraction	0.363095238				0.285714286					0.295238095					
May 10-31 a	6	6	6	8	5	6	6	6	6	7	7	7	7	7	90
May 10-31 b	3	3	3	3	3	3	3	3	4	6	6	6	6	6	58
Hour Total	9	9	9	11	8	9	9	9	10	13	13	13	13	13	148
TOTAL	38				45					65					148
Fraction	0.215909091				0.204545455					0.295454545					

BOS to IAD Unreliability Measures for 3-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	5	5	5	6	4	4	4	4	4	4	4	4	4	4	61
Nov 1-21 b	11	11	11	14	12	12	12	12	12	14	13	13	13	13	173
Hour Total	16	16	16	20	16	16	16	16	16	18	17	17	17	17	234
TOTAL	68				80					86					234
Fraction	0.404761905				0.380952381					0.40952381					
Nov22-Dec12 a	6	6	6	9	8	8	8	8	8	10	7	8	8	8	108
Nov22-Dec12 b	12	13	13	14	12	12	12	12	12	14	10	10	10	10	166
Hour Total	18	19	19	23	20	20	20	20	20	24	17	18	18	18	274
TOTAL	79				100					95					274
Fraction	0.470238095				0.476190476					0.452380952					
Dec13-Jan3 a	9	9	9	11	9	9	9	9	9	13	10	10	10	10	136
Dec13-Jan3 b	15	15	15	15	11	11	11	11	11	15	13	13	13	13	182
Hour Total	24	24	24	26	20	20	20	20	20	28	23	23	23	23	318
TOTAL	98				100					120					318
Fraction	0.583333333				0.476190476					0.571428571					
Jan 4-24 a	15	16	16	18	15	15	15	15	15	18	17	17	17	16	225
Jan 4-24 b	13	13	13	14	13	13	13	13	13	17	11	11	11	11	179
Hour Total	28	29	29	32	28	28	28	28	28	35	28	28	28	27	404
TOTAL	118				140					146					404
Fraction	0.702380952				0.666666667					0.695238095					
Jan25-Feb14 a	6	6	6	8	7	7	7	7	7	9	8	8	8	8	102
Jan25-Feb14 b	13	13	13	17	17	17	17	17	17	18	14	14	14	13	214
Hour Total	19	19	19	25	24	24	24	24	24	27	22	22	22	21	316
TOTAL	82				120					114					316
Fraction	0.488095238				0.571428571					0.542857143					
Feb15-Mar7 a	10	10	10	11	9	9	9	9	9	12	9	9	9	9	134
Feb15-Mar7 b	15	15	15	16	15	15	15	15	15	16	16	16	16	16	216
Hour Total	25	25	25	27	24	24	24	24	24	28	25	25	25	25	350
TOTAL	102				120					128					350
Fraction	0.607142857				0.571428571					0.60952381					
Mar 8-28 a	10	10	10	11	10	10	10	10	10	13	13	13	13	12	155
Mar 8-28 b	3	3	3	7	7	7	7	7	7	8	6	6	7	7	85
Hour Total	13	13	13	18	17	17	17	17	17	21	19	19	20	19	240
TOTAL	57				85					98					240
Fraction	0.339285714				0.404761905					0.466666667					
Mar29-Apr18 a	9	9	9	12	7	7	7	7	8	11	9	9	9	9	122
Mar29-Apr18 b	13	13	13	14	13	13	13	13	13	16	14	14	14	14	190
Hour Total	22	22	22	26	20	20	20	20	21	27	23	23	23	23	312
TOTAL	92				101					119					312
Fraction	0.547619048				0.480952381					0.566666667					
Apr19-May9 a	9	10	10	12	10	10	10	10	10	14	13	13	13	13	157
Apr19-May9 b	8	8	8	13	11	11	11	11	11	12	8	8	8	8	136
Hour Total	17	18	18	25	21	21	21	21	21	26	21	21	21	21	293
TOTAL	78				105					110					293
Fraction	0.464285714				0.5					0.523809524					
May 10-31 a	6	6	6	6	5	5	5	5	5	6	6	7	7	7	82
May 10-31 b	14	15	15	16	14	14	14	14	14	15	12	12	12	12	193
Hour Total	20	21	21	22	19	19	19	19	19	21	18	19	19	19	275
TOTAL	84				95					96					275
Fraction	0.477272727				0.431818182					0.436363636					

BOS to IAD Unreliability Measures for 3-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	5	5	5	6	4	4	4	4	4	4	4	4	4	4	61
Nov 1-21 b	11	11	11	14	11	11	11	11	11	12	11	11	11	11	158
Hour Total	16	16	16	20	15	15	15	15	15	16	15	15	15	15	219
TOTAL	68				75					76					219
Fraction	0.404761905				0.357142857					0.361904762					
Nov22-Dec12 a	6	6	6	9	7	7	7	7	7	9	7	8	8	8	102
Nov22-Dec12 b	11	12	12	14	12	12	12	12	12	14	10	10	10	10	163
Hour Total	17	18	18	23	19	19	19	19	19	23	17	18	18	18	265
TOTAL	76				95					94					265
Fraction	0.452380952				0.452380952					0.447619048					
Dec13-Jan3 a	9	9	9	11	9	9	9	9	9	13	10	10	10	10	136
Dec13-Jan3 b	15	15	15	15	10	10	10	10	10	15	13	13	13	13	177
Hour Total	24	24	24	26	19	19	19	19	19	28	23	23	23	23	313
TOTAL	98				95					120					313
Fraction	0.583333333				0.452380952					0.571428571					
Jan 4-24 a	15	16	16	18	15	15	15	15	15	18	16	16	16	15	221
Jan 4-24 b	13	13	13	14	13	13	13	13	13	17	11	11	11	11	179
Hour Total	28	29	29	32	28	28	28	28	28	35	27	27	27	26	400
TOTAL	118				140					142					400
Fraction	0.702380952				0.666666667					0.676190476					
Jan25-Feb14 a	6	6	6	8	7	7	7	7	7	9	8	8	8	8	102
Jan25-Feb14 b	13	13	13	17	17	17	17	17	17	18	14	14	14	13	214
Hour Total	19	19	19	25	24	24	24	24	24	27	22	22	22	21	316
TOTAL	82				120					114					316
Fraction	0.488095238				0.571428571					0.542857143					
Feb15-Mar7 a	10	10	10	11	9	9	9	9	9	12	9	9	9	9	134
Feb15-Mar7 b	15	15	15	16	15	15	15	15	15	16	16	16	16	16	216
Hour Total	25	25	25	27	24	24	24	24	24	28	25	25	25	25	350
TOTAL	102				120					128					350
Fraction	0.607142857				0.571428571					0.60952381					
Mar 8-28 a	10	10	10	11	10	10	10	10	10	13	13	13	13	12	155
Mar 8-28 b	3	3	3	7	7	7	7	7	7	8	6	6	7	7	85
Hour Total	13	13	13	18	17	17	17	17	17	21	19	19	20	19	240
TOTAL	57				85					98					240
Fraction	0.339285714				0.404761905					0.466666667					
Mar29-Apr18 a	8	8	8	10	6	6	6	6	7	9	7	7	7	7	102
Mar29-Apr18 b	13	13	13	13	12	12	12	12	12	16	14	14	14	14	184
Hour Total	21	21	21	23	18	18	18	18	19	25	21	21	21	21	286
TOTAL	86				91					109					286
Fraction	0.511904762				0.433333333					0.519047619					
Apr19-May9 a	8	9	9	12	10	10	10	10	10	13	11	11	11	11	145
Apr19-May9 b	8	8	8	13	11	11	11	11	11	12	8	8	8	8	136
Hour Total	16	17	17	25	21	21	21	21	21	25	19	19	19	19	281
TOTAL	75				105					101					281
Fraction	0.446428571				0.5					0.480952381					
May 10-31 a	6	6	6	6	4	4	4	4	4	5	5	6	6	6	72
May 10-31 b	13	14	14	15	12	12	12	12	12	13	11	11	11	11	173
Hour Total	19	20	20	21	16	16	16	16	16	18	16	17	17	17	245
TOTAL	80				80					85					245
Fraction	0.454545455				0.363636364					0.386363636					

BOS to IAD Unreliability Measures for 3-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	4	4	4	5	4	4	4	4	4	4	4	4	4	4	57
Nov 1-21 b	7	7	7	11	10	10	10	10	10	12	11	11	11	11	138
Hour Total	11	11	11	16	14	14	14	14	14	16	15	15	15	15	195
TOTAL			49				70					76			195
Fraction			0.291666667				0.333333333					0.361904762			
Nov22-Dec12 a	5	5	5	6	5	5	5	5	5	8	7	8	8	8	85
Nov22-Dec12 b	11	11	11	13	12	12	12	12	12	13	9	9	9	9	155
Hour Total	16	16	16	19	17	17	17	17	17	21	16	17	17	17	240
TOTAL			67				85					88			240
Fraction			0.398809524				0.404761905					0.419047619			
Dec13-Jan3 a	8	8	8	10	7	7	7	7	7	10	7	8	8	8	110
Dec13-Jan3 b	15	15	15	15	9	9	9	9	9	13	12	12	12	12	166
Hour Total	23	23	23	25	16	16	16	16	16	23	19	20	20	20	276
TOTAL			94				80					102			276
Fraction			0.55952381				0.380952381					0.485714286			
Jan 4-24 a	15	16	16	17	14	14	14	14	14	16	14	14	14	14	206
Jan 4-24 b	13	13	13	14	13	13	13	13	13	17	11	11	11	11	179
Hour Total	28	29	29	31	27	27	27	27	27	33	25	25	25	25	385
TOTAL			117				135					133			385
Fraction			0.696428571				0.642857143					0.633333333			
Jan25-Feb14 a	6	6	6	8	7	7	7	7	7	8	7	7	7	7	97
Jan25-Feb14 b	11	11	11	16	16	16	16	16	16	17	14	14	14	13	201
Hour Total	17	17	17	24	23	23	23	23	23	25	21	21	21	20	298
TOTAL			75				115					108			298
Fraction			0.446428571				0.547619048					0.514285714			
Feb15-Mar7 a	9	9	9	10	8	8	8	8	8	11	8	8	8	8	120
Feb15-Mar7 b	15	15	15	16	13	13	13	13	13	14	14	14	14	14	196
Hour Total	24	24	24	26	21	21	21	21	21	25	22	22	22	22	316
TOTAL			98				105					113			316
Fraction			0.583333333				0.5					0.538095238			
Mar 8-28 a	9	9	9	10	9	9	9	9	9	13	13	13	13	12	146
Mar 8-28 b	3	3	3	6	5	5	5	5	5	7	6	6	7	7	73
Hour Total	12	12	12	16	14	14	14	14	14	20	19	19	20	19	219
TOTAL			52				70					97			219
Fraction			0.30952381				0.333333333					0.461904762			
Mar29-Apr18 a	7	7	7	9	6	6	6	6	7	9	8	8	8	8	102
Mar29-Apr18 b	13	13	13	13	11	11	11	11	11	14	12	12	12	12	169
Hour Total	20	20	20	22	17	17	17	17	18	23	20	20	20	20	271
TOTAL			82				86					103			271
Fraction			0.488095238				0.40952381					0.49047619			
Apr19-May9 a	8	9	9	11	8	8	8	8	8	10	9	9	9	9	123
Apr19-May9 b	6	6	6	9	7	7	7	7	7	7	4	4	4	4	85
Hour Total	14	15	15	20	15	15	15	15	15	17	13	13	13	13	208
TOTAL			64				75					69			208
Fraction			0.380952381				0.357142857					0.328571429			
May 10-31 a	5	5	5	5	3	3	3	3	3	4	4	5	5	5	58
May 10-31 b	6	6	6	8	6	6	6	6	7	7	5	5	5	5	84
Hour Total	11	11	11	13	9	9	9	9	10	11	9	10	10	10	142
TOTAL			46				46					50			142
Fraction			0.261363636				0.209090909					0.227272727			

BOS to IAD Unreliability Measures for 3-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	4	4	4	5	4	4	4	4	4	4	4	4	4	4	57
Nov 1-21 b	7	7	7	11	10	10	10	10	10	12	11	11	11	11	138
Hour Total	11	11	11	16	14	14	14	14	14	16	15	15	15	15	195
TOTAL		49				70					76				195
Fraction		0.291666667				0.333333333					0.361904762				
Nov22-Dec12 a	3	3	3	3	2	2	2	2	2	3	3	3	3	3	37
Nov22-Dec12 b	11	11	11	13	10	10	11	11	11	13	8	8	8	8	144
Hour Total	14	14	14	16	12	12	13	13	13	16	11	11	11	11	181
TOTAL		58				63					60				181
Fraction		0.345238095				0.3					0.285714286				
Dec13-Jan3 a	8	8	8	10	7	7	7	7	7	10	7	8	8	8	110
Dec13-Jan3 b	12	12	12	14	9	9	9	9	9	12	11	11	11	11	151
Hour Total	20	20	20	24	16	16	16	16	16	22	18	19	19	19	261
TOTAL		84				80					97				261
Fraction		0.5				0.380952381					0.461904762				
Jan 4-24 a	13	14	14	17	14	14	14	14	14	15	13	13	13	13	195
Jan 4-24 b	13	13	13	14	13	13	13	13	13	17	11	11	11	11	179
Hour Total	26	27	27	31	27	27	27	27	27	32	24	24	24	24	374
TOTAL		111				135					128				374
Fraction		0.660714286				0.642857143					0.60952381				
Jan25-Feb14 a	6	6	6	8	7	7	7	7	7	8	7	7	7	7	97
Jan25-Feb14 b	10	10	10	15	15	15	15	15	15	15	12	12	12	12	183
Hour Total	16	16	16	23	22	22	22	22	22	23	19	19	19	19	280
TOTAL		71				110					99				280
Fraction		0.422619048				0.523809524					0.471428571				
Feb15-Mar7 a	8	8	8	10	8	8	8	8	8	11	7	7	7	7	113
Feb15-Mar7 b	14	14	14	15	12	12	12	12	12	13	13	13	13	13	182
Hour Total	22	22	22	25	20	20	20	20	20	24	20	20	20	20	295
TOTAL		91				100					104				295
Fraction		0.541666667				0.476190476					0.495238095				
Mar 8-28 a	9	9	9	10	9	9	9	9	9	12	11	11	11	10	137
Mar 8-28 b	2	2	2	4	3	3	3	3	3	5	5	5	6	6	52
Hour Total	11	11	11	14	12	12	12	12	12	17	16	16	17	16	189
TOTAL		47				60					82				189
Fraction		0.279761905				0.285714286					0.39047619				
Mar29-Apr18 a	7	7	7	7	4	4	4	4	5	7	6	6	6	6	80
Mar29-Apr18 b	13	13	13	13	11	11	11	11	11	14	10	10	10	10	161
Hour Total	20	20	20	20	15	15	15	15	16	21	16	16	16	16	241
TOTAL		80				76					85				241
Fraction		0.476190476				0.361904762					0.404761905				
Apr19-May9 a	7	8	8	9	6	6	6	6	6	8	8	8	8	8	102
Apr19-May9 b	5	5	5	5	3	3	3	3	3	4	3	3	3	3	51
Hour Total	12	13	13	14	9	9	9	9	9	12	11	11	11	11	153
TOTAL		52				45					56				153
Fraction		0.30952381				0.214285714					0.266666667				
May 10-31 a	5	5	5	5	3	3	3	3	3	4	4	5	5	5	58
May 10-31 b	2	2	2	3	3	3	3	3	4	4	4	4	4	4	45
Hour Total	7	7	7	8	6	6	6	6	7	8	8	9	9	9	103
TOTAL		29				31					43				103
Fraction		0.164772727				0.140909091					0.195454545				

CLE to IAD Unreliability Measures for 3-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	3	3	3	4	3	3	3	3	3	4	4	4	4	4	48
Nov 1-21 b	16	16	16	17	13	13	13	13	13	16	15	15	15	15	206
Hour Total	19	19	19	21	16	16	16	16	16	20	19	19	19	19	254
TOTAL		78				80					96				254
Fraction		0.464285714				0.380952381					0.457142857				
Nov22-Dec12 a	6	6	6	10	9	9	9	9	9	9	8	8	8	8	114
Nov22-Dec12 b	14	14	14	15	14	14	14	14	14	16	16	16	16	16	207
Hour Total	20	20	20	25	23	23	23	23	23	25	24	24	24	24	321
TOTAL		85				115					121				321
Fraction		0.505952381				0.547619048					0.576190476				
Dec13-Jan3 a	15	15	15	18	17	17	17	17	17	18	12	13	13	13	217
Dec13-Jan3 b	18	18	18	20	19	19	19	19	19	19	16	16	16	16	252
Hour Total	33	33	33	38	36	36	36	36	36	37	28	29	29	29	469
TOTAL		137				180					152				469
Fraction		0.81547619				0.857142857					0.723809524				
Jan 4-24 a	15	15	15	17	14	14	14	14	14	16	15	15	15	15	208
Jan 4-24 b	12	12	12	13	11	11	11	11	11	17	14	14	14	14	177
Hour Total	27	27	27	30	25	25	25	25	25	33	29	29	29	29	385
TOTAL		111				125					149				385
Fraction		0.660714286				0.595238095					0.70952381				
Jan25-Feb14 a	9	9	9	12	10	10	10	10	10	10	7	7	7	7	127
Jan25-Feb14 b	14	14	14	16	15	15	16	16	16	19	18	18	18	17	226
Hour Total	23	23	23	28	25	25	26	26	26	29	25	25	25	24	353
TOTAL		97				128					128				353
Fraction		0.577380952				0.60952381					0.60952381				
Feb15-Mar7 a	11	12	12	13	11	11	11	11	11	13	12	12	12	12	164
Feb15-Mar7 b	16	16	16	17	17	17	17	17	17	19	19	19	19	19	245
Hour Total	27	28	28	30	28	28	28	28	28	32	31	31	31	31	409
TOTAL		113				140					156				409
Fraction		0.672619048				0.666666667					0.742857143				
Mar 8-28 a	14	14	14	16	13	13	13	13	13	15	14	14	14	13	193
Mar 8-28 b	7	7	7	10	10	10	10	10	10	12	11	11	11	11	137
Hour Total	21	21	21	26	23	23	23	23	23	27	25	25	25	24	330
TOTAL		89				115					126				330
Fraction		0.529761905				0.547619048					0.6				
Mar29-Apr18 a	12	12	12	14	11	11	11	11	11	12	10	11	11	11	160
Mar29-Apr18 b	10	10	10	11	11	11	11	11	12	13	12	12	12	12	158
Hour Total	22	22	22	25	22	22	22	22	23	25	22	23	23	23	318
TOTAL		91				111					116				318
Fraction		0.541666667				0.528571429					0.552380952				
Apr19-May9 a	8	8	8	11	9	9	10	10	10	12	9	9	9	9	131
Apr19-May9 b	9	9	9	10	8	8	8	8	8	10	7	7	7	7	115
Hour Total	17	17	17	21	17	17	18	18	18	22	16	16	16	16	246
TOTAL		72				88					86				246
Fraction		0.428571429				0.419047619					0.40952381				
May 10-31 a	6	6	6	9	7	7	8	8	8	10	8	9	9	9	110
May 10-31 b	14	14	14	14	10	10	10	10	11	13	7	7	8	8	150
Hour Total	20	20	20	23	17	17	18	18	19	23	15	16	17	17	260
TOTAL		83				89					88				260
Fraction		0.471590909				0.404545455					0.4				

CLE to IAD Unreliability Measures for 3-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	3	3	3	4	3	3	3	3	3	4	4	4	4	4	48
Nov 1-21 b	16	16	16	17	12	12	12	12	12	15	14	14	14	14	196
Hour Total	19	19	19	21	15	15	15	15	15	19	18	18	18	18	244
TOTAL	78				75					91					244
Fraction	0.464285714				0.357142857					0.433333333					
Nov22-Dec12 a	5	5	5	10	8	8	8	8	8	9	8	8	8	8	106
Nov22-Dec12 b	13	13	13	15	14	14	14	14	14	16	16	16	16	16	204
Hour Total	18	18	18	25	22	22	22	22	22	25	24	24	24	24	310
TOTAL	79				110					121					310
Fraction	0.470238095				0.523809524					0.576190476					
Dec13-Jan3 a	14	14	14	18	16	16	16	16	16	17	11	12	12	12	204
Dec13-Jan3 b	18	18	18	20	18	18	18	18	18	19	16	16	16	16	247
Hour Total	32	32	32	38	34	34	34	34	34	36	27	28	28	28	451
TOTAL	134				170					147					451
Fraction	0.797619048				0.80952381					0.7					
Jan 4-24 a	15	15	15	17	14	14	14	14	14	16	14	14	14	14	204
Jan 4-24 b	12	12	12	13	11	11	11	11	11	17	14	14	14	14	177
Hour Total	27	27	27	30	25	25	25	25	25	33	28	28	28	28	381
TOTAL	111				125					145					381
Fraction	0.660714286				0.595238095					0.69047619					
Jan25-Feb14 a	8	8	8	11	9	9	9	9	9	10	7	7	7	7	118
Jan25-Feb14 b	14	14	14	16	15	15	16	16	16	19	18	18	18	17	226
Hour Total	22	22	22	27	24	24	25	25	25	29	25	25	25	24	344
TOTAL	93				123					128					344
Fraction	0.553571429				0.585714286					0.60952381					
Feb15-Mar7 a	11	12	12	13	11	11	11	11	11	13	12	12	12	12	164
Feb15-Mar7 b	16	16	16	17	17	17	17	17	17	19	19	19	19	19	245
Hour Total	27	28	28	30	28	28	28	28	28	32	31	31	31	31	409
TOTAL	113				140					156					409
Fraction	0.672619048				0.666666667					0.742857143					
Mar 8-28 a	14	14	14	16	13	13	13	13	13	15	14	14	14	13	193
Mar 8-28 b	7	7	7	10	10	10	10	10	10	12	11	11	11	11	137
Hour Total	21	21	21	26	23	23	23	23	23	27	25	25	25	24	330
TOTAL	89				115					126					330
Fraction	0.529761905				0.547619048					0.6					
Mar29-Apr18 a	11	11	11	11	8	8	8	8	8	9	9	9	9	9	129
Mar29-Apr18 b	10	10	10	11	11	11	11	11	12	13	12	12	12	12	158
Hour Total	21	21	21	22	19	19	19	19	20	22	21	21	21	21	287
TOTAL	85				96					106					287
Fraction	0.505952381				0.457142857					0.504761905					
Apr19-May9 a	8	8	8	10	8	8	9	9	9	10	7	7	7	7	115
Apr19-May9 b	9	9	9	10	8	8	8	8	8	10	7	7	7	7	115
Hour Total	17	17	17	20	16	16	17	17	17	20	14	14	14	14	230
TOTAL	71				83					76					230
Fraction	0.422619048				0.395238095					0.361904762					
May 10-31 a	5	5	5	6	4	4	5	5	5	8	7	8	8	8	83
May 10-31 b	12	12	12	12	8	8	8	8	9	11	7	7	7	7	128
Hour Total	17	17	17	18	12	12	13	13	14	19	14	15	15	15	211
TOTAL	69				64					78					211
Fraction	0.392045455				0.290909091					0.354545455					

CLE to IAD Unreliability Measures for 3-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	1	1	1	2	2	2	2	2	2	3	3	3	3	3	30
Nov 1-21 b	12	12	12	15	12	12	12	12	12	15	14	14	14	14	182
Hour Total	13	13	13	17	14	14	14	14	14	18	17	17	17	17	212
TOTAL	56				70					86					212
Fraction	0.333333333				0.333333333					0.40952381					
Nov22-Dec12 a	3	3	3	6	5	5	5	5	5	7	7	7	7	7	75
Nov22-Dec12 b	13	13	13	15	13	13	13	13	13	15	15	15	15	15	194
Hour Total	16	16	16	21	18	18	18	18	18	22	22	22	22	22	269
TOTAL	69				90					110					269
Fraction	0.410714286				0.428571429					0.523809524					
Dec13-Jan3 a	13	13	13	17	15	15	15	15	15	15	8	9	9	9	181
Dec13-Jan3 b	18	18	18	20	16	16	16	16	16	17	14	14	14	14	227
Hour Total	31	31	31	37	31	31	31	31	31	32	22	23	23	23	408
TOTAL	130				155					123					408
Fraction	0.773809524				0.738095238					0.585714286					
Jan 4-24 a	14	14	14	16	12	12	12	12	12	15	14	14	14	14	189
Jan 4-24 b	11	11	11	13	11	11	11	11	11	17	14	14	14	14	174
Hour Total	25	25	25	29	23	23	23	23	23	32	28	28	28	28	363
TOTAL	104				115					144					363
Fraction	0.619047619				0.547619048					0.685714286					
Jan25-Feb14 a	8	8	8	11	9	9	9	9	9	9	6	6	6	6	113
Jan25-Feb14 b	14	14	14	16	15	15	16	16	16	19	18	18	18	17	226
Hour Total	22	22	22	27	24	24	25	25	25	28	24	24	24	23	339
TOTAL	93				123					123					339
Fraction	0.553571429				0.585714286					0.585714286					
Feb15-Mar7 a	10	11	11	11	10	10	10	10	10	12	11	11	11	11	149
Feb15-Mar7 b	16	16	16	17	17	17	17	17	17	19	19	19	19	19	245
Hour Total	26	27	27	28	27	27	27	27	27	31	30	30	30	30	394
TOTAL	108				135					151					394
Fraction	0.642857143				0.642857143					0.719047619					
Mar 8-28 a	12	12	12	14	11	11	11	11	11	15	14	14	14	13	175
Mar 8-28 b	7	7	7	9	9	9	9	9	9	12	11	11	11	11	131
Hour Total	19	19	19	23	20	20	20	20	20	27	25	25	25	24	306
TOTAL	80				100					126					306
Fraction	0.476190476				0.476190476					0.6					
Mar29-Apr18 a	11	11	11	11	8	8	8	8	8	9	9	9	9	9	129
Mar29-Apr18 b	9	9	9	9	9	9	9	9	9	9	8	8	8	8	122
Hour Total	20	20	20	20	17	17	17	17	17	18	17	17	17	17	251
TOTAL	80				85					86					251
Fraction	0.476190476				0.404761905					0.40952381					
Apr19-May9 a	6	6	6	10	6	6	7	7	7	8	6	6	6	6	93
Apr19-May9 b	4	4	4	6	5	5	5	5	5	7	5	5	5	5	70
Hour Total	10	10	10	16	11	11	12	12	12	15	11	11	11	11	163
TOTAL	46				58					59					163
Fraction	0.273809524				0.276190476					0.280952381					
May 10-31 a	4	4	4	5	3	3	4	4	4	6	5	6	6	6	64
May 10-31 b	5	5	5	7	5	5	5	5	6	8	6	6	6	6	80
Hour Total	9	9	9	12	8	8	9	9	10	14	11	12	12	12	144
TOTAL	39				44					61					144
Fraction	0.221590909				0.2					0.277272727					

CLE to IAD Unreliability Measures for 3-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1-21 a	1	1	1	1	1	1	1	1	1	3	3	3	3	3	24
Nov 1-21 b	12	12	12	15	12	12	12	12	12	15	14	14	14	14	182
Hour Total	13	13	13	16	13	13	13	13	13	18	17	17	17	17	206
TOTAL	55				65					86					206
Fraction	0.327380952				0.30952381					0.40952381					
Nov22-Dec12 a	2	2	2	2	2	2	2	2	2	4	4	4	4	4	38
Nov22-Dec12 b	13	13	13	15	13	13	13	13	13	15	14	14	14	14	190
Hour Total	15	15	15	17	15	15	15	15	15	19	18	18	18	18	228
TOTAL	62				75					91					228
Fraction	0.369047619				0.357142857					0.433333333					
Dec13-Jan3 a	13	13	13	17	15	15	15	15	15	15	8	9	9	9	181
Dec13-Jan3 b	12	12	12	17	15	15	15	15	15	17	14	14	14	14	201
Hour Total	25	25	25	34	30	30	30	30	30	32	22	23	23	23	382
TOTAL	109				150					123					382
Fraction	0.648809524				0.714285714					0.585714286					
Jan 4-24 a	12	12	12	16	12	12	12	12	12	14	13	13	13	13	178
Jan 4-24 b	11	11	11	13	11	11	11	11	11	17	14	14	14	14	174
Hour Total	23	23	23	29	23	23	23	23	23	31	27	27	27	27	352
TOTAL	98				115					139					352
Fraction	0.583333333				0.547619048					0.661904762					
Jan25-Feb14 a	8	8	8	11	9	9	9	9	9	9	6	6	6	6	113
Jan25-Feb14 b	13	13	13	16	14	14	15	15	15	18	17	17	17	17	214
Hour Total	21	21	21	27	23	23	24	24	24	27	23	23	23	23	327
TOTAL	90				118					119					327
Fraction	0.535714286				0.561904762					0.566666667					
Feb15-Mar7 a	9	10	10	11	10	10	10	10	10	12	11	11	11	11	146
Feb15-Mar7 b	16	16	16	16	16	16	16	16	16	18	18	18	18	18	234
Hour Total	25	26	26	27	26	26	26	26	26	30	29	29	29	29	380
TOTAL	104				130					146					380
Fraction	0.619047619				0.619047619					0.695238095					
Mar 8-28 a	11	11	11	13	9	9	9	9	9	13	12	12	12	11	151
Mar 8-28 b	3	3	3	6	6	6	6	6	6	10	9	9	9	9	91
Hour Total	14	14	14	19	15	15	15	15	15	23	21	21	21	20	242
TOTAL	61				75					106					242
Fraction	0.363095238				0.357142857					0.504761905					
Mar29-Apr18 a	10	10	10	10	6	6	6	6	6	7	7	7	7	7	105
Mar29-Apr18 b	9	9	9	9	8	8	8	8	8	8	5	5	5	5	104
Hour Total	19	19	19	19	14	14	14	14	14	15	12	12	12	12	209
TOTAL	76				70					63					209
Fraction	0.452380952				0.333333333					0.3					
Apr19-May9 a	5	5	5	7	4	4	5	5	5	6	5	5	5	5	71
Apr19-May9 b	3	3	3	3	1	1	1	1	1	2	2	2	2	2	27
Hour Total	8	8	8	10	5	5	6	6	6	8	7	7	7	7	98
TOTAL	34				28					36					98
Fraction	0.202380952				0.133333333					0.171428571					
May 10-31 a	4	4	4	5	2	2	3	3	3	5	5	6	6	6	58
May 10-31 b	2	2	2	2	2	2	2	2	3	5	3	3	3	3	36
Hour Total	6	6	6	7	4	4	5	5	6	10	8	9	9	9	94
TOTAL	25				24					45					94
Fraction	0.142045455				0.109090909					0.204545455					

BOS to CLE Unreliability Measures for 2-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	5	5	5	6	4	4	4	4	4	5	4	4	4	4	62
Nov 2a	12	12	12	13	12	12	12	12	12	13	12	12	12	12	170
Hour Total	17	17	17	19	16	16	16	16	16	18	16	16	16	16	232
TOTAL		70				80					82				232
Fraction		0.583333333				0.533333333					0.546666667				
Nov 1b	5	5	5	7	6	6	6	6	6	8	7	7	7	7	88
Nov 2b	12	12	12	13	13	13	13	13	13	13	13	13	13	13	179
Hour Total	17	17	17	20	19	19	19	19	19	21	20	20	20	20	267
TOTAL		71				95					101				267
Fraction		0.591666667				0.633333333					0.673333333				
Dec 1a	4	4	4	8	6	6	6	6	6	7	5	5	5	5	77
Dec 2a	9	9	9	10	10	10	10	10	10	11	9	9	9	9	134
Hour Total	13	13	13	18	16	16	16	16	16	18	14	14	14	14	211
TOTAL		57				80					74				211
Fraction		0.475				0.533333333					0.493333333				
Dec 1b	13	13	13	15	15	15	15	15	15	16	14	15	15	15	204
Dec 2b	13	13	13	15	14	14	14	14	14	14	13	13	13	13	190
Hour Total	26	26	26	30	29	29	29	29	29	30	27	28	28	28	394
TOTAL		108				145					141				394
Fraction		0.84375				0.90625					0.88125				
Jan 1a	10	11	11	12	12	12	12	12	12	12	11	11	11	10	159
Jan 2a	13	13	13	13	13	13	13	13	13	13	13	13	13	13	182
Hour Total	23	24	24	25	25	25	25	25	25	25	24	24	24	23	341
TOTAL		96				125					120				341
Fraction		0.8				0.833333333					0.8				
Jan 1b	11	11	11	12	10	10	10	10	10	10	9	9	9	9	141
Jan 2b	11	11	11	12	11	11	11	11	11	13	10	10	10	10	153
Hour Total	22	22	22	24	21	21	21	21	21	23	19	19	19	19	294
TOTAL		90				105					99				294
Fraction		0.703125				0.65625					0.61875				
Feb 1a	8	8	8	10	8	8	8	8	8	10	7	7	7	7	112
Feb 2a	10	10	10	11	10	10	10	10	10	11	9	9	9	8	137
Hour Total	18	18	18	21	18	18	18	18	18	21	16	16	16	15	249
TOTAL		75				90					84				249
Fraction		0.625				0.6					0.56				
Feb 1b	10	10	10	11	8	8	8	8	8	8	8	8	8	8	121
Feb 2b	9	9	9	10	6	6	6	6	6	9	9	9	9	9	112
Hour Total	19	19	19	21	14	14	14	14	14	17	17	17	17	17	233
TOTAL		78				70					85				233
Fraction		0.75				0.538461538					0.653846154				
Mar 1a	8	10	10	12	10	10	10	10	10	12	9	9	9	8	137
Mar 2a	9	9	9	11	11	11	11	11	11	14	12	12	12	12	155
Hour Total	17	19	19	23	21	21	21	21	21	26	21	21	21	20	292
TOTAL		78				105					109				292
Fraction		0.65				0.7					0.726666667				
Mar 1b	13	13	13	14	11	11	11	11	11	12	12	12	12	12	168
Mar 2b	7	7	7	7	7	7	7	7	7	9	9	9	9	9	108
Hour Total	20	20	20	21	18	18	18	18	18	21	21	21	21	21	276
TOTAL		81				90					105				276
Fraction		0.6328125				0.5625					0.65625				
Apr 1a	12	12	12	13	9	9	9	9	9	10	10	10	10	10	144
Apr 2a	11	11	11	10	9	9	9	9	10	11	10	10	10	10	140
Hour Total	23	23	23	23	18	18	18	18	19	21	20	20	20	20	284
TOTAL		92				91					101				284
Fraction		0.766666667				0.606666667					0.673333333				
Apr 1b	7	7	7	8	8	8	8	8	8	9	9	9	9	9	114
Apr 2b	10	10	10	11	9	9	9	9	9	11	9	9	9	9	133
Hour Total	17	17	17	19	17	17	17	17	17	20	18	18	18	18	247
TOTAL		70				85					92				247
Fraction		0.583333333				0.566666667					0.613333333				
May 1a	8	9	9	10	8	8	8	8	8	10	8	8	8	8	118
May 2a	8	8	8	8	7	7	7	7	7	9	8	8	8	8	108
Hour Total	16	17	17	18	15	15	15	15	15	19	16	16	16	16	226
TOTAL		68				75					83				226
Fraction		0.566666667				0.5					0.553333333				
May 1b	4	4	4	8	5	6	6	6	6	7	7	7	7	7	84
May 2b	10	10	10	10	9	9	9	9	9	10	7	7	7	7	123
Hour Total	14	14	14	18	14	15	15	15	15	17	14	14	14	14	207
TOTAL		60				74					73				207
Fraction		0.46875				0.4625					0.45625				

BOS to CLE Unreliability Measures for 2-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	5	5	5	6	4	4	4	4	4	5	4	4	4	4	62
Nov 2a	12	12	12	13	10	10	10	10	10	11	10	10	10	10	150
Hour Total	17	17	17	19	14	14	14	14	14	16	14	14	14	14	212
TOTAL		70				70					72				212
Fraction		0.583333333				0.466666667					0.48				
Nov 1b	4	4	4	6	5	5	5	5	5	8	7	7	7	7	79
Nov 2b	12	12	12	13	13	13	13	13	13	13	13	13	13	13	179
Hour Total	16	16	16	19	18	18	18	18	18	21	20	20	20	20	258
TOTAL		67				90					101				258
Fraction		0.558333333				0.6					0.673333333				
Dec 1a	4	4	4	7	5	5	5	5	5	6	5	5	5	5	70
Dec 2a	9	9	9	10	10	10	10	10	10	11	9	9	9	9	134
Hour Total	13	13	13	17	15	15	15	15	15	17	14	14	14	14	204
TOTAL		56				75					73				204
Fraction		0.466666667				0.5					0.486666667				
Dec 1b	12	12	12	15	15	15	15	15	15	15	13	14	14	14	196
Dec 2b	13	13	13	14	13	13	13	13	13	14	13	13	13	13	184
Hour Total	25	25	25	29	28	28	28	28	28	29	26	27	27	27	380
TOTAL		104				140					136				380
Fraction		0.8125				0.875					0.85				
Jan 1a	10	11	11	12	12	12	12	12	12	12	11	11	11	10	159
Jan 2a	13	13	13	13	13	13	13	13	13	13	13	13	13	13	182
Hour Total	23	24	24	25	25	25	25	25	25	25	24	24	24	23	341
TOTAL		96				125					120				341
Fraction		0.8				0.833333333					0.8				
Jan 1b	11	11	11	11	9	9	9	9	9	10	9	9	9	9	135
Jan 2b	11	11	11	12	11	11	11	11	11	13	10	10	10	10	153
Hour Total	22	22	22	23	20	20	20	20	20	23	19	19	19	19	288
TOTAL		89				100					99				288
Fraction		0.6953125				0.625					0.61875				
Feb 1a	8	8	8	10	8	8	8	8	8	10	7	7	7	7	112
Feb 2a	10	10	10	11	10	10	10	10	10	11	9	9	9	8	137
Hour Total	18	18	18	21	18	18	18	18	18	21	16	16	16	15	249
TOTAL		75				90					84				249
Fraction		0.625				0.6					0.56				
Feb 1b	10	10	10	11	8	8	8	8	8	8	8	8	8	8	121
Feb 2b	9	9	9	10	6	6	6	6	6	9	9	9	9	9	112
Hour Total	19	19	19	21	14	14	14	14	14	17	17	17	17	17	233
TOTAL		78				70					85				233
Fraction		0.75				0.538461538					0.653846154				
Mar 1a	8	10	10	12	10	10	10	10	10	12	9	9	9	8	137
Mar 2a	9	9	9	11	11	11	11	11	11	14	12	12	12	12	155
Hour Total	17	19	19	23	21	21	21	21	21	26	21	21	21	20	292
TOTAL		78				105					109				292
Fraction		0.65				0.7					0.726666667				
Mar 1b	13	13	13	14	11	11	11	11	11	12	12	12	12	12	168
Mar 2b	7	7	7	7	7	7	7	7	7	9	9	9	9	9	108
Hour Total	20	20	20	21	18	18	18	18	18	21	21	21	21	21	276
TOTAL		81				90					105				276
Fraction		0.6328125				0.5625					0.65625				
Apr 1a	10	10	10	10	7	7	7	7	7	8	8	8	8	8	115
Apr 2a	11	11	11	10	9	9	9	9	10	11	10	10	10	10	140
Hour Total	21	21	21	20	16	16	16	16	17	19	18	18	18	18	255
TOTAL		83				81					91				255
Fraction		0.691666667				0.54					0.606666667				
Apr 1b	7	7	7	8	8	8	8	8	8	9	9	9	9	9	114
Apr 2b	10	10	10	11	9	9	9	9	9	11	9	9	9	9	133
Hour Total	17	17	17	19	17	17	17	17	17	20	18	18	18	18	247
TOTAL		70				85					92				247
Fraction		0.583333333				0.566666667					0.613333333				
May 1a	7	8	8	10	7	7	7	7	7	9	6	6	6	6	101
May 2a	8	8	8	8	7	7	7	7	7	9	8	8	8	8	108
Hour Total	15	16	16	18	14	14	14	14	14	18	14	14	14	14	209
TOTAL		65				70					74				209
Fraction		0.541666667				0.466666667					0.493333333				
May 1b	3	3	3	5	3	4	4	4	4	5	5	5	5	5	58
May 2b	9	9	9	9	7	7	7	7	7	8	6	6	6	6	103
Hour Total	12	12	12	14	10	11	11	11	11	13	11	11	11	11	161
TOTAL		50				54					57				161
Fraction		0.390625				0.3375					0.35625				

BOS to CLE Unreliability Measures for 2-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	3	3	3	4	3	3	3	3	3	4	4	4	4	4	48
Nov 2a	10	10	10	11	10	10	10	10	10	11	10	10	10	10	142
Hour Total	13	13	13	15	13	13	13	13	13	15	14	14	14	14	190
TOTAL		54				65					71				190
Fraction		0.45				0.433333333					0.473333333				
Nov 1b	2	2	2	3	2	2	2	2	2	5	5	5	5	5	44
Nov 2b	12	12	12	13	13	13	13	13	13	13	13	13	13	13	179
Hour Total	14	14	14	16	15	15	15	15	15	18	18	18	18	18	223
TOTAL		58				75					90				223
Fraction		0.483333333				0.5					0.6				
Dec 1a	4	4	4	6	4	4	4	4	4	6	5	5	5	5	64
Dec 2a	9	9	9	10	10	10	10	10	10	11	9	9	9	9	134
Hour Total	13	13	13	16	14	14	14	14	14	17	14	14	14	14	198
TOTAL		55				70					73				198
Fraction		0.458333333				0.466666667					0.486666667				
Dec 1b	11	11	11	14	14	14	14	14	14	14	11	12	12	12	178
Dec 2b	13	13	13	14	11	11	11	11	11	12	11	11	11	11	164
Hour Total	24	24	24	28	25	25	25	25	25	26	22	23	23	23	342
TOTAL		100				125					117				342
Fraction		0.78125				0.78125					0.73125				
Jan 1a	9	10	10	12	11	11	11	11	11	11	10	10	10	10	147
Jan 2a	13	13	13	13	13	13	13	13	13	13	13	13	13	13	182
Hour Total	22	23	23	25	24	24	24	24	24	24	23	23	23	23	329
TOTAL		93				120					116				329
Fraction		0.775				0.8					0.773333333				
Jan 1b	11	11	11	11	9	9	9	9	9	9	8	8	8	8	130
Jan 2b	11	11	11	12	11	11	11	11	11	13	10	10	10	10	153
Hour Total	22	22	22	23	20	20	20	20	20	22	18	18	18	18	283
TOTAL		89				100					94				283
Fraction		0.6953125				0.625					0.5875				
Feb 1a	8	8	8	10	8	8	8	8	8	10	7	7	7	7	112
Feb 2a	10	10	10	11	10	10	10	10	10	11	9	9	9	8	137
Hour Total	18	18	18	21	18	18	18	18	18	21	16	16	16	15	249
TOTAL		75				90					84				249
Fraction		0.625				0.6					0.56				
Feb 1b	9	9	9	9	7	7	7	7	7	7	7	7	7	7	106
Feb 2b	9	9	9	10	6	6	6	6	6	9	9	9	9	9	112
Hour Total	18	18	18	19	13	13	13	13	13	16	16	16	16	16	218
TOTAL		73				65					80				218
Fraction		0.701923077				0.5					0.615384615				
Mar 1a	8	10	10	11	9	9	9	9	9	12	9	9	9	8	131
Mar 2a	9	9	9	11	10	10	10	10	10	13	12	12	12	12	149
Hour Total	17	19	19	22	19	19	19	19	19	25	21	21	21	20	280
TOTAL		77				95					108				280
Fraction		0.641666667				0.633333333					0.72				
Mar 1b	11	11	11	13	10	10	10	10	10	12	12	12	12	12	156
Mar 2b	7	7	7	7	7	7	7	7	7	9	9	9	9	9	108
Hour Total	18	18	18	20	17	17	17	17	17	21	21	21	21	21	264
TOTAL		74				85					105				264
Fraction		0.578125				0.53125					0.65625				
Apr 1a	9	9	9	9	7	7	7	7	7	7	7	7	7	7	106
Apr 2a	11	11	11	10	8	8	8	8	8	9	8	8	8	8	124
Hour Total	20	20	20	19	15	15	15	15	15	16	15	15	15	15	230
TOTAL		79				75					76				230
Fraction		0.658333333				0.5					0.506666667				
Apr 1b	6	6	6	8	8	8	8	8	8	9	8	8	8	8	107
Apr 2b	6	6	6	7	6	6	6	6	6	9	7	7	7	7	92
Hour Total	12	12	12	15	14	14	14	14	14	18	15	15	15	15	199
TOTAL		51				70					78				199
Fraction		0.425				0.466666667					0.52				
May 1a	7	8	8	8	3	3	3	3	3	4	3	3	3	3	62
May 2a	4	4	4	6	5	5	5	5	5	6	5	5	5	5	69
Hour Total	11	12	12	14	8	8	8	8	8	10	8	8	8	8	131
TOTAL		49				40					42				131
Fraction		0.408333333				0.266666667					0.28				
May 1b	2	2	2	4	3	4	4	4	4	4	4	4	4	4	49
May 2b	4	4	4	5	3	3	3	3	4	5	3	3	3	3	50
Hour Total	6	6	6	9	6	7	7	7	8	9	7	7	7	7	99
TOTAL		27				35					37				99
Fraction		0.2109375				0.21875					0.23125				

BOS to CLE Unreliability Measures for 2-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	3	3	3	4	3	3	3	3	3	4	4	4	4	4	48
Nov 2a	10	10	10	11	10	10	10	10	10	11	10	10	10	10	142
Hour Total	13	13	13	15	13	13	13	13	13	15	14	14	14	14	190
TOTAL		54				65					71				190
Fraction		0.45				0.433333333					0.473333333				
Nov 1b	1	1	1	1	1	1	1	1	1	3	3	3	3	3	24
Nov 2b	12	12	12	13	13	13	13	13	13	13	13	13	13	13	179
Hour Total	13	13	13	14	14	14	14	14	14	16	16	16	16	16	203
TOTAL		53				70					80				203
Fraction		0.441666667				0.466666667					0.533333333				
Dec 1a	4	4	4	5	3	3	3	3	3	4	3	3	3	3	48
Dec 2a	9	9	9	10	10	10	10	10	10	11	9	9	9	9	134
Hour Total	13	13	13	15	13	13	13	13	13	15	12	12	12	12	182
TOTAL		54				65					63				182
Fraction		0.45				0.433333333					0.42				
Dec 1b	11	11	11	14	14	14	14	14	14	14	11	12	12	12	178
Dec 2b	10	10	10	13	10	10	10	10	10	12	11	11	11	11	149
Hour Total	21	21	21	27	24	24	24	24	24	26	22	23	23	23	327
TOTAL		90				120					117				327
Fraction		0.703125				0.75					0.73125				
Jan 1a	9	10	10	12	11	11	11	11	11	11	10	10	10	10	147
Jan 2a	12	12	12	13	13	13	13	13	13	13	13	13	13	13	179
Hour Total	21	22	22	25	24	24	24	24	24	24	23	23	23	23	326
TOTAL		90				120					116				326
Fraction		0.75				0.8					0.773333333				
Jan 1b	9	9	9	11	9	9	9	9	9	9	8	8	8	8	124
Jan 2b	11	11	11	12	11	11	11	11	11	13	10	10	10	10	153
Hour Total	20	20	20	23	20	20	20	20	20	22	18	18	18	18	277
TOTAL		83				100					94				277
Fraction		0.6484375				0.625					0.5875				
Feb 1a	8	8	8	10	8	8	8	8	8	10	7	7	7	7	112
Feb 2a	8	8	8	11	10	10	10	10	10	11	8	8	8	8	128
Hour Total	16	16	16	21	18	18	18	18	18	21	15	15	15	15	240
TOTAL		69				90					81				240
Fraction		0.575				0.6					0.54				
Feb 1b	9	9	9	9	7	7	7	7	7	7	7	7	7	7	106
Feb 2b	8	8	8	8	5	5	5	5	5	8	8	8	8	8	97
Hour Total	17	17	17	17	12	12	12	12	12	15	15	15	15	15	203
TOTAL		68				60					75				203
Fraction		0.653846154				0.461538462					0.576923077				
Mar 1a	7	9	9	11	9	9	9	9	9	11	8	8	8	7	123
Mar 2a	8	8	8	10	9	9	9	9	9	13	12	12	12	12	140
Hour Total	15	17	17	21	18	18	18	18	18	24	20	20	20	19	263
TOTAL		70				90					103				263
Fraction		0.583333333				0.6					0.686666667				
Mar 1b	11	11	11	13	10	10	10	10	10	12	12	12	12	12	156
Mar 2b	4	4	4	5	5	5	5	5	5	7	7	7	7	7	77
Hour Total	15	15	15	18	15	15	15	15	15	19	19	19	19	19	233
TOTAL		63				75					95				233
Fraction		0.4921875				0.46875					0.59375				
Apr 1a	8	8	8	8	5	5	5	5	5	5	5	5	5	5	82
Apr 2a	11	11	11	10	8	8	8	8	8	9	7	7	7	7	120
Hour Total	19	19	19	18	13	13	13	13	13	14	12	12	12	12	202
TOTAL		75				65					62				202
Fraction		0.625				0.433333333					0.413333333				
Apr 1b	6	6	6	7	7	7	7	7	7	8	8	8	8	8	100
Apr 2b	6	6	6	6	5	5	5	5	5	7	5	5	5	5	76
Hour Total	12	12	12	13	12	12	12	12	12	15	13	13	13	13	176
TOTAL		49				60					67				176
Fraction		0.408333333				0.4					0.446666667				
May 1a	6	7	7	7	2	2	2	2	2	3	3	3	3	3	52
May 2a	4	4	4	5	4	4	4	4	4	5	4	4	4	4	58
Hour Total	10	11	11	12	6	6	6	6	6	8	7	7	7	7	110
TOTAL		44				30					36				110
Fraction		0.366666667				0.2					0.24				
May 1b	2	2	2	4	3	4	4	4	4	4	4	4	4	4	49
May 2b	0	0	0	0	0	0	0	0	1	2	2	2	2	2	11
Hour Total	2	2	2	4	3	4	4	4	5	6	6	6	6	6	60
TOTAL		10				20					30				60
Fraction		0.078125				0.125					0.1875				

BOS to IAD Unreliability Measures for 2-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	4	4	4	5	3	3	3	3	3	3	3	3	3	3	47
Nov 2a	7	7	7	9	7	7	7	7	7	9	8	8	8	8	106
Hour Total	11	11	11	14	10	10	10	10	10	12	11	11	11	11	153
TOTAL		47				50					56				153
Fraction		0.391666667				0.333333333					0.373333333				
Nov 1b	3	3	3	5	4	4	4	4	4	5	3	3	3	3	51
Nov 2b	9	10	10	12	11	11	11	11	11	12	10	10	10	10	148
Hour Total	12	13	13	17	15	15	15	15	15	17	13	13	13	13	199
TOTAL		55				75					69				199
Fraction		0.458333333				0.5					0.46				
Dec 1a	6	6	6	7	6	6	6	6	6	7	6	7	7	7	89
Dec 2a	9	9	9	9	8	8	8	8	8	10	8	8	8	8	118
Hour Total	15	15	15	16	14	14	14	14	14	17	14	15	15	15	207
TOTAL		61				70					76				207
Fraction		0.508333333				0.466666667					0.506666667				
Dec 1b	6	6	6	8	7	7	7	7	7	11	9	9	9	9	108
Dec 2b	10	10	10	10	7	7	7	7	7	9	7	7	7	7	112
Hour Total	16	16	16	18	14	14	14	14	14	20	16	16	16	16	220
TOTAL		66				70					84				220
Fraction		0.515625				0.4375					0.525				
Jan 1a	8	9	9	11	10	10	10	10	10	12	10	10	10	9	138
Jan 2a	12	12	12	13	11	11	11	11	11	14	11	11	11	11	162
Hour Total	20	21	21	24	21	21	21	21	21	26	21	21	21	20	300
TOTAL		86				105					109				300
Fraction		0.716666667				0.7					0.726666667				
Jan 1b	11	11	11	11	8	8	8	8	8	10	10	10	10	10	134
Jan 2b	11	11	11	11	11	11	11	11	11	13	8	8	8	8	144
Hour Total	22	22	22	22	19	19	19	19	19	23	18	18	18	18	278
TOTAL		88				95					95				278
Fraction		0.6875				0.59375					0.59375				
Feb 1a	3	3	3	5	5	5	5	5	5	6	5	5	5	5	65
Feb 2a	7	7	7	11	11	11	11	11	11	12	10	10	10	9	138
Hour Total	10	10	10	16	16	16	16	16	16	18	15	15	15	14	203
TOTAL		46				80					77				203
Fraction		0.383333333				0.533333333					0.513333333				
Feb 1b	7	7	7	8	6	6	6	6	6	8	6	6	6	6	91
Feb 2b	9	9	9	10	10	10	10	10	10	11	11	11	11	11	142
Hour Total	16	16	16	18	16	16	16	16	16	19	17	17	17	17	233
TOTAL		66				80					87				233
Fraction		0.634615385				0.615384615					0.669230769				
Mar 1a	5	5	5	6	6	6	6	6	6	8	7	7	7	6	86
Mar 2a	7	7	7	9	8	8	8	8	8	8	6	6	7	7	104
Hour Total	12	12	12	15	14	14	14	14	14	16	13	13	14	13	190
TOTAL		51				70					69				190
Fraction		0.425				0.466666667					0.46				
Mar 1b	11	11	11	11	9	9	9	9	9	11	10	10	10	10	140
Mar 2b	3	3	3	5	5	5	5	5	5	7	6	6	6	6	70
Hour Total	14	14	14	16	14	14	14	14	14	18	16	16	16	16	210
TOTAL		58				70					82				210
Fraction		0.453125				0.4375					0.5125				
Apr 1a	6	6	6	8	4	4	4	4	5	8	8	8	8	8	87
Apr 2a	10	10	10	10	9	9	9	9	9	10	9	9	9	9	131
Hour Total	16	16	16	18	13	13	13	13	14	18	17	17	17	17	218
TOTAL		66				66					86				218
Fraction		0.55				0.44					0.573333333				
Apr 1b	6	6	6	8	7	7	7	7	7	10	9	9	9	9	107
Apr 2b	7	7	7	11	10	10	10	10	10	11	9	9	9	9	129
Hour Total	13	13	13	19	17	17	17	17	17	21	18	18	18	18	236
TOTAL		58				85					93				236
Fraction		0.483333333				0.566666667					0.62				
May 1a	7	8	8	9	8	8	8	8	8	10	9	9	9	9	118
May 2a	6	6	6	8	7	7	7	7	7	8	6	6	6	6	93
Hour Total	13	14	14	17	15	15	15	15	15	18	15	15	15	15	211
TOTAL		58				75					78				211
Fraction		0.483333333				0.5					0.52				
May 1b	2	2	2	2	1	1	1	1	1	1	1	2	2	2	21
May 2b	11	12	12	12	10	10	10	10	10	11	8	8	8	8	140
Hour Total	13	14	14	14	11	11	11	11	11	12	9	10	10	10	161
TOTAL		55				55					51				161
Fraction		0.4296875				0.34375					0.31875				

BOS to IAD Unreliability Measures for 2-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	4	4	4	5	3	3	3	3	3	3	3	3	3	3	47
Nov 2a	7	7	7	9	6	6	6	6	6	7	6	6	6	6	91
Hour Total	11	11	11	14	9	9	9	9	9	10	9	9	9	9	138
TOTAL		47				45					46				138
Fraction		0.391666667				0.3					0.306666667				
Nov 1b	3	3	3	5	3	3	3	3	3	4	3	3	3	3	45
Nov 2b	9	10	10	12	11	11	11	11	11	12	10	10	10	10	148
Hour Total	12	13	13	17	14	14	14	14	14	16	13	13	13	13	193
TOTAL		55				70					68				193
Fraction		0.458333333				0.466666667					0.453333333				
Dec 1a	6	6	6	7	6	6	6	6	6	7	6	7	7	7	89
Dec 2a	8	8	8	9	8	8	8	8	8	10	8	8	8	8	115
Hour Total	14	14	14	16	14	14	14	14	14	17	14	15	15	15	204
TOTAL		58				70					76				204
Fraction		0.483333333				0.466666667					0.506666667				
Dec 1b	6	6	6	8	7	7	7	7	7	11	9	9	9	9	108
Dec 2b	10	10	10	10	6	6	6	6	6	9	7	7	7	7	107
Hour Total	16	16	16	18	13	13	13	13	13	20	16	16	16	16	215
TOTAL		66				65					84				215
Fraction		0.515625				0.40625					0.525				
Jan 1a	8	9	9	11	10	10	10	10	10	12	10	10	10	9	138
Jan 2a	12	12	12	13	11	11	11	11	11	14	11	11	11	11	162
Hour Total	20	21	21	24	21	21	21	21	21	26	21	21	21	20	300
TOTAL		86				105					109				300
Fraction		0.716666667				0.7					0.726666667				
Jan 1b	11	11	11	11	8	8	8	8	8	10	9	9	9	9	130
Jan 2b	11	11	11	11	11	11	11	11	11	13	8	8	8	8	144
Hour Total	22	22	22	22	19	19	19	19	19	23	17	17	17	17	274
TOTAL		88				95					91				274
Fraction		0.6875				0.59375					0.56875				
Feb 1a	3	3	3	5	5	5	5	5	5	6	5	5	5	5	65
Feb 2a	7	7	7	11	11	11	11	11	11	12	10	10	10	9	138
Hour Total	10	10	10	16	16	16	16	16	16	18	15	15	15	14	203
TOTAL		46				80					77				203
Fraction		0.383333333				0.533333333					0.513333333				
Feb 1b	7	7	7	8	6	6	6	6	6	8	6	6	6	6	91
Feb 2b	9	9	9	10	10	10	10	10	10	11	11	11	11	11	142
Hour Total	16	16	16	18	16	16	16	16	16	19	17	17	17	17	233
TOTAL		66				80					87				233
Fraction		0.634615385				0.615384615					0.669230769				
Mar 1a	5	5	5	6	6	6	6	6	6	8	7	7	7	6	86
Mar 2a	7	7	7	9	8	8	8	8	8	8	6	6	7	7	104
Hour Total	12	12	12	15	14	14	14	14	14	16	13	13	14	13	190
TOTAL		51				70					69				190
Fraction		0.425				0.466666667					0.46				
Mar 1b	11	11	11	11	9	9	9	9	9	11	10	10	10	10	140
Mar 2b	3	3	3	5	5	5	5	5	5	7	6	6	6	6	70
Hour Total	14	14	14	16	14	14	14	14	14	18	16	16	16	16	210
TOTAL		58				70					82				210
Fraction		0.453125				0.4375					0.5125				
Apr 1a	5	5	5	7	4	4	4	4	5	7	6	6	6	6	74
Apr 2a	10	10	10	10	9	9	9	9	9	10	9	9	9	9	131
Hour Total	15	15	15	17	13	13	13	13	14	17	15	15	15	15	205
TOTAL		62				66					77				205
Fraction		0.516666667				0.44					0.513333333				
Apr 1b	6	6	6	7	6	6	6	6	6	9	9	9	9	9	100
Apr 2b	7	7	7	10	9	9	9	9	9	11	9	9	9	9	123
Hour Total	13	13	13	17	15	15	15	15	15	20	18	18	18	18	223
TOTAL		56				75					92				223
Fraction		0.466666667				0.5					0.613333333				
May 1a	6	7	7	9	7	7	7	7	7	8	6	6	6	6	96
May 2a	5	5	5	8	7	7	7	7	7	8	6	6	6	6	90
Hour Total	11	12	12	17	14	14	14	14	14	16	12	12	12	12	186
TOTAL		52				70					64				186
Fraction		0.433333333				0.466666667					0.426666667				
May 1b	2	2	2	2	1	1	1	1	1	1	1	2	2	2	21
May 2b	10	11	11	11	8	8	8	8	8	9	7	7	7	7	120
Hour Total	12	13	13	13	9	9	9	9	9	10	8	9	9	9	141
TOTAL		51				45					45				141
Fraction		0.3984375				0.28125					0.28125				

BOS to IAD Unreliability Measures for 2-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	3	3	3	4	3	3	3	3	3	3	3	3	3	3	43
Nov 2a	5	5	5	7	6	6	6	6	6	7	6	6	6	6	83
Hour Total	8	8	8	11	9	9	9	9	9	10	9	9	9	9	126
TOTAL	35				45					46					126
Fraction	0.291666667				0.3					0.306666667					
Nov 1b	2	2	2	2	1	1	1	1	1	3	3	3	3	3	28
Nov 2b	7	7	7	10	10	10	10	10	10	12	10	10	10	10	133
Hour Total	9	9	9	12	11	11	11	11	11	15	13	13	13	13	161
TOTAL	39				55					67					161
Fraction	0.325				0.366666667					0.446666667					
Dec 1a	6	6	6	7	6	6	6	6	6	7	5	6	6	6	85
Dec 2a	8	8	8	9	8	8	8	8	8	9	7	7	7	7	110
Hour Total	14	14	14	16	14	14	14	14	14	16	12	13	13	13	195
TOTAL	58				70					67					195
Fraction	0.483333333				0.466666667					0.446666667					
Dec 1b	5	5	5	7	5	5	5	5	5	8	7	8	8	8	86
Dec 2b	10	10	10	10	5	5	5	5	5	7	6	6	6	6	96
Hour Total	15	15	15	17	10	10	10	10	10	15	13	14	14	14	182
TOTAL	62				50					70					182
Fraction	0.484375				0.3125					0.4375					
Jan 1a	8	9	9	10	9	9	9	9	9	11	9	9	9	9	128
Jan 2a	12	12	12	13	11	11	11	11	11	14	11	11	11	11	162
Hour Total	20	21	21	23	20	20	20	20	20	25	20	20	20	20	290
TOTAL	85				100					105					290
Fraction	0.708333333				0.666666667					0.7					
Jan 1b	11	11	11	11	8	8	8	8	8	8	7	7	7	7	120
Jan 2b	9	9	9	10	10	10	10	10	10	12	8	8	8	8	131
Hour Total	20	20	20	21	18	18	18	18	18	20	15	15	15	15	251
TOTAL	81				90					80					251
Fraction	0.6328125				0.5625					0.5					
Feb 1a	3	3	3	5	5	5	5	5	5	6	5	5	5	5	65
Feb 2a	7	7	7	11	11	11	11	11	11	12	10	10	10	9	138
Hour Total	10	10	10	16	16	16	16	16	16	18	15	15	15	14	203
TOTAL	46				80					77					203
Fraction	0.383333333				0.533333333					0.513333333					
Feb 1b	6	6	6	7	5	5	5	5	5	7	5	5	5	5	77
Feb 2b	9	9	9	10	9	9	9	9	9	10	10	10	10	10	132
Hour Total	15	15	15	17	14	14	14	14	14	17	15	15	15	15	209
TOTAL	62				70					77					209
Fraction	0.596153846				0.538461538					0.592307692					
Mar 1a	5	5	5	6	6	6	6	6	6	8	7	7	7	6	86
Mar 2a	7	7	7	9	6	6	6	6	6	6	5	5	6	6	88
Hour Total	12	12	12	15	12	12	12	12	12	14	12	12	13	12	174
TOTAL	51				60					63					174
Fraction	0.425				0.4					0.42					
Mar 1b	10	10	10	10	8	8	8	8	8	11	10	10	10	10	131
Mar 2b	3	3	3	4	4	4	4	4	4	7	6	6	6	6	64
Hour Total	13	13	13	14	12	12	12	12	12	18	16	16	16	16	195
TOTAL	53				60					82					195
Fraction	0.4140625				0.375					0.5125					
Apr 1a	4	4	4	6	4	4	4	4	5	7	7	7	7	7	74
Apr 2a	10	10	10	10	8	8	8	8	8	9	8	8	8	8	121
Hour Total	14	14	14	16	12	12	12	12	13	16	15	15	15	15	195
TOTAL	58				61					76					195
Fraction	0.483333333				0.406666667					0.506666667					
Apr 1b	6	6	6	7	6	6	6	6	6	8	8	8	8	8	95
Apr 2b	6	6	6	8	7	7	7	7	7	8	6	6	6	6	93
Hour Total	12	12	12	15	13	13	13	13	13	16	14	14	14	14	188
TOTAL	51				65					72					188
Fraction	0.425				0.433333333					0.48					
May 1a	6	7	7	8	4	4	4	4	4	5	4	4	4	4	69
May 2a	3	3	3	6	5	5	5	5	5	5	4	4	4	4	61
Hour Total	9	10	10	14	9	9	9	9	9	10	8	8	8	8	130
TOTAL	43				45					42					130
Fraction	0.358333333				0.3					0.28					
May 1b	1	1	1	1	1	1	1	1	1	1	1	2	2	2	17
May 2b	4	4	4	4	2	2	2	2	3	3	1	1	1	1	34
Hour Total	5	5	5	5	3	3	3	3	4	4	2	3	3	3	51
TOTAL	20				16					15					51
Fraction	0.15625				0.1					0.09375					

BOS to IAD Unreliability Measures for 2-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	3	3	3	4	3	3	3	3	3	3	3	3	3	3	43
Nov 2a	5	5	5	7	6	6	6	6	6	7	6	6	6	6	83
Hour Total	8	8	8	11	9	9	9	9	9	10	9	9	9	9	126
TOTAL	35				45					46					126
Fraction	0.291666667				0.3					0.306666667					
Nov 1b	1	1	1	1	1	1	1	1	1	1	1	1	1	1	14
Nov 2b	7	7	7	10	9	9	10	10	10	12	9	9	9	9	127
Hour Total	8	8	8	11	10	10	11	11	11	13	10	10	10	10	141
TOTAL	35				53					53					141
Fraction	0.291666667				0.353333333					0.353333333					
Dec 1a	5	5	5	5	3	3	3	3	3	4	3	3	3	3	51
Dec 2a	8	8	8	9	7	7	7	7	7	9	7	7	7	7	105
Hour Total	13	13	13	14	10	10	10	10	10	13	10	10	10	10	156
TOTAL	53				50					53					156
Fraction	0.441666667				0.333333333					0.353333333					
Dec 1b	5	5	5	7	5	5	5	5	5	8	7	8	8	8	86
Dec 2b	8	8	8	9	5	5	5	5	5	6	5	5	5	5	84
Hour Total	13	13	13	16	10	10	10	10	10	14	12	13	13	13	170
TOTAL	55				50					65					170
Fraction	0.4296875				0.3125					0.40625					
Jan 1a	8	9	9	10	9	9	9	9	9	10	8	8	8	8	123
Jan 2a	11	11	11	13	11	11	11	11	11	14	11	11	11	11	159
Hour Total	19	20	20	23	20	20	20	20	20	24	19	19	19	19	282
TOTAL	82				100					100					282
Fraction	0.683333333				0.666666667					0.666666667					
Jan 1b	9	9	9	11	8	8	8	8	8	8	7	7	7	7	114
Jan 2b	9	9	9	10	10	10	10	10	10	12	8	8	8	8	131
Hour Total	18	18	18	21	18	18	18	18	18	20	15	15	15	15	245
TOTAL	75				90					80					245
Fraction	0.5859375				0.5625					0.5					
Feb 1a	3	3	3	5	5	5	5	5	5	6	5	5	5	5	65
Feb 2a	6	6	6	10	10	10	10	10	10	10	8	8	8	8	120
Hour Total	9	9	9	15	15	15	15	15	15	16	13	13	13	13	185
TOTAL	42				75					68					185
Fraction	0.35				0.5					0.453333333					
Feb 1b	6	6	6	7	5	5	5	5	5	7	5	5	5	5	77
Feb 2b	8	8	8	9	8	8	8	8	8	9	9	9	9	9	118
Hour Total	14	14	14	16	13	13	13	13	13	16	14	14	14	14	195
TOTAL	58				65					72					195
Fraction	0.557692308				0.5					0.553846154					
Mar 1a	4	4	4	6	6	6	6	6	6	7	5	5	5	4	74
Mar 2a	7	7	7	8	5	5	5	5	5	5	5	5	6	6	81
Hour Total	11	11	11	14	11	11	11	11	11	12	10	10	11	10	155
TOTAL	47				55					53					155
Fraction	0.391666667				0.366666667					0.353333333					
Mar 1b	10	10	10	10	8	8	8	8	8	11	9	9	9	9	127
Mar 2b	2	2	2	3	3	3	3	3	3	6	5	5	5	5	50
Hour Total	12	12	12	13	11	11	11	11	11	17	14	14	14	14	177
TOTAL	49				55					73					177
Fraction	0.3828125				0.34375					0.45625					
Apr 1a	4	4	4	4	2	2	2	2	3	5	5	5	5	5	52
Apr 2a	10	10	10	10	8	8	8	8	8	9	7	7	7	7	117
Hour Total	14	14	14	14	10	10	10	10	11	14	12	12	12	12	169
TOTAL	56				51					62					169
Fraction	0.466666667				0.34					0.413333333					
Apr 1b	6	6	6	7	6	6	6	6	6	8	8	8	8	8	95
Apr 2b	5	5	5	5	4	4	4	4	4	6	4	4	4	4	62
Hour Total	11	11	11	12	10	10	10	10	10	14	12	12	12	12	157
TOTAL	45				50					62					157
Fraction	0.375				0.333333333					0.413333333					
May 1a	5	6	6	6	2	2	2	2	2	3	3	3	3	3	48
May 2a	3	3	3	4	3	3	3	3	3	3	3	3	3	3	43
Hour Total	8	9	9	10	5	5	5	5	5	6	6	6	6	6	91
TOTAL	36				25					30					91
Fraction	0.3				0.166666667					0.2					
May 1b	1	1	1	1	1	1	1	1	1	1	1	2	2	2	17
May 2b	0	0	0	0	0	0	0	0	1	1	1	1	1	1	6
Hour Total	1	1	1	1	1	1	1	1	2	2	2	3	3	3	23
TOTAL	4				6					13					23
Fraction	0.03125				0.0375					0.08125					

CLE to IAD Unreliability Measures for 2-Week Period – 12,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	1	1	1	2	1	1	1	1	1	2	2	2	2	2	20
Nov 2a	11	11	11	12	9	9	9	9	9	11	10	10	10	10	141
Hour Total	12	12	12	14	10	10	10	10	10	13	12	12	12	12	161
TOTAL	50				50					61					161
Fraction	0.416666667				0.333333333					0.406666667					
Nov 1b	5	5	5	5	5	5	5	5	5	5	4	4	4	4	66
Nov 2b	13	13	13	13	11	11	11	11	11	13	13	13	13	13	172
Hour Total	18	18	18	18	16	16	16	16	16	18	17	17	17	17	238
TOTAL	72				80					86					238
Fraction	0.6				0.533333333					0.573333333					
Dec 1a	5	5	5	9	7	7	7	7	7	7	7	7	7	7	94
Dec 2a	8	8	8	9	9	9	9	9	9	10	10	10	10	10	128
Hour Total	13	13	13	18	16	16	16	16	16	17	17	17	17	17	222
TOTAL	57				80					85					222
Fraction	0.475				0.533333333					0.566666667					
Dec 1b	12	12	12	14	14	14	14	14	14	15	10	11	11	11	178
Dec 2b	13	13	13	15	14	14	14	14	14	14	11	11	11	11	182
Hour Total	25	25	25	29	28	28	28	28	28	29	21	22	22	22	360
TOTAL	104				140					116					360
Fraction	0.8125				0.875					0.725					
Jan 1a	10	10	10	12	10	10	10	10	10	12	10	10	10	10	144
Jan 2a	10	10	10	11	11	11	11	11	11	13	12	12	12	12	157
Hour Total	20	20	20	23	21	21	21	21	21	25	22	22	22	22	301
TOTAL	83				105					113					301
Fraction	0.691666667				0.7					0.753333333					
Jan 1b	9	9	9	11	9	9	9	9	9	9	9	9	9	9	128
Jan 2b	11	11	11	11	9	9	9	9	9	14	12	12	12	12	151
Hour Total	20	20	20	22	18	18	18	18	18	23	21	21	21	21	279
TOTAL	82				90					107					279
Fraction	0.640625				0.5625					0.66875					
Feb 1a	6	6	6	8	7	7	7	7	7	8	5	5	5	5	89
Feb 2a	9	9	9	11	10	10	11	11	11	13	12	12	12	11	151
Hour Total	15	15	15	19	17	17	18	18	18	21	17	17	17	16	240
TOTAL	64				88					88					240
Fraction	0.533333333				0.586666667					0.586666667					
Feb 1b	8	8	8	9	8	8	8	8	8	8	7	7	7	7	109
Feb 2b	10	10	10	11	11	11	11	11	11	11	11	11	11	11	151
Hour Total	18	18	18	20	19	19	19	19	19	19	18	18	18	18	260
TOTAL	74				95					91					260
Fraction	0.711538462				0.730769231					0.7					
Mar 1a	6	7	7	9	7	7	7	7	7	9	9	9	9	8	108
Mar 2a	8	8	8	10	10	10	10	10	10	13	12	12	12	12	145
Hour Total	14	15	15	19	17	17	17	17	17	22	21	21	21	20	253
TOTAL	63				85					105					253
Fraction	0.525				0.566666667					0.7					
Mar 1b	13	13	13	13	11	11	11	11	11	12	11	11	11	11	163
Mar 2b	7	7	7	8	8	8	8	8	8	9	9	9	9	9	114
Hour Total	20	20	20	21	19	19	19	19	19	21	20	20	20	20	277
TOTAL	81				95					101					277
Fraction	0.6328125				0.59375					0.63125					
Apr 1a	10	10	10	11	8	8	8	8	8	9	8	9	9	9	125
Apr 2a	5	5	5	6	6	6	6	6	7	8	7	7	7	7	88
Hour Total	15	15	15	17	14	14	14	14	15	17	15	16	16	16	213
TOTAL	62				71					80					213
Fraction	0.516666667				0.473333333					0.533333333					
Apr 1b	6	6	6	8	6	6	7	7	7	8	6	6	6	6	91
Apr 2b	9	9	9	9	8	8	8	8	8	8	6	6	6	6	108
Hour Total	15	15	15	17	14	14	15	15	15	16	12	12	12	12	199
TOTAL	62				73					64					199
Fraction	0.516666667				0.486666667					0.426666667					
May 1a	5	5	5	7	6	6	6	6	6	8	6	6	6	6	84
May 2a	5	5	5	6	5	5	5	5	5	8	6	6	6	6	78
Hour Total	10	10	10	13	11	11	11	11	11	16	12	12	12	12	162
TOTAL	43				55					64					162
Fraction	0.358333333				0.366666667					0.426666667					
May 1b	3	3	3	6	5	5	6	6	6	7	5	6	6	6	73
May 2b	11	11	11	11	7	7	7	7	8	9	4	4	5	5	107
Hour Total	14	14	14	17	12	12	13	13	14	16	9	10	11	11	180
TOTAL	59				64					57					180
Fraction	0.4609375				0.4					0.35625					

CLE to IAD Unreliability Measures for 2-Week Period – 10,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	1	1	1	2	1	1	1	1	1	2	2	2	2	2	20
Nov 2a	11	11	11	12	8	8	8	8	8	10	9	9	9	9	131
Hour Total	12	12	12	14	9	9	9	9	9	12	11	11	11	11	151
TOTAL		50				45					56				151
Fraction		0.416666667				0.3					0.373333333				
Nov 1b	4	4	4	5	4	4	4	4	4	5	4	4	4	4	58
Nov 2b	13	13	13	13	11	11	11	11	11	13	13	13	13	13	172
Hour Total	17	17	17	18	15	15	15	15	15	18	17	17	17	17	230
TOTAL		69				75					86				230
Fraction		0.575				0.5					0.573333333				
Dec 1a	5	5	5	9	6	6	6	6	6	7	7	7	7	7	89
Dec 2a	7	7	7	9	9	9	9	9	9	10	10	10	10	10	125
Hour Total	12	12	12	18	15	15	15	15	15	17	17	17	17	17	214
TOTAL		54				75					85				214
Fraction		0.45				0.5					0.566666667				
Dec 1b	11	11	11	14	14	14	14	14	14	14	9	10	10	10	170
Dec 2b	13	13	13	15	13	13	13	13	13	14	11	11	11	11	177
Hour Total	24	24	24	29	27	27	27	27	27	28	20	21	21	21	347
TOTAL		101				135					111				347
Fraction		0.7890625				0.84375					0.69375				
Jan 1a	10	10	10	12	10	10	10	10	10	12	10	10	10	10	144
Jan 2a	10	10	10	11	11	11	11	11	11	13	12	12	12	12	157
Hour Total	20	20	20	23	21	21	21	21	21	25	22	22	22	22	301
TOTAL		83				105					113				301
Fraction		0.691666667				0.7					0.753333333				
Jan 1b	9	9	9	10	8	8	8	8	8	9	8	8	8	8	118
Jan 2b	11	11	11	11	9	9	9	9	9	14	12	12	12	12	151
Hour Total	20	20	20	21	17	17	17	17	17	23	20	20	20	20	269
TOTAL		81				85					103				269
Fraction		0.6328125				0.53125					0.64375				
Feb 1a	5	5	5	8	7	7	7	7	7	8	5	5	5	5	86
Feb 2a	9	9	9	11	10	10	11	11	11	13	12	12	12	11	151
Hour Total	14	14	14	19	17	17	18	18	18	21	17	17	17	16	237
TOTAL		61				88					88				237
Fraction		0.508333333				0.586666667					0.586666667				
Feb 1b	8	8	8	9	8	8	8	8	8	8	7	7	7	7	109
Feb 2b	10	10	10	11	11	11	11	11	11	11	11	11	11	11	151
Hour Total	18	18	18	20	19	19	19	19	19	19	18	18	18	18	260
TOTAL		74				95					91				260
Fraction		0.711538462				0.730769231					0.7				
Mar 1a	6	7	7	9	7	7	7	7	7	9	9	9	9	8	108
Mar 2a	8	8	8	10	10	10	10	10	10	13	12	12	12	12	145
Hour Total	14	15	15	19	17	17	17	17	17	22	21	21	21	20	253
TOTAL		63				85					105				253
Fraction		0.525				0.566666667					0.7				
Mar 1b	13	13	13	13	11	11	11	11	11	12	11	11	11	11	163
Mar 2b	7	7	7	8	8	8	8	8	8	9	9	9	9	9	114
Hour Total	20	20	20	21	19	19	19	19	19	21	20	20	20	20	277
TOTAL		81				95					101				277
Fraction		0.6328125				0.59375					0.63125				
Apr 1a	9	9	9	9	6	6	6	6	6	7	7	7	7	7	101
Apr 2a	5	5	5	6	6	6	6	6	7	8	7	7	7	7	88
Hour Total	14	14	14	15	12	12	12	12	13	15	14	14	14	14	189
TOTAL		57				61					71				189
Fraction		0.475				0.406666667					0.473333333				
Apr 1b	6	6	6	7	5	5	6	6	6	7	6	6	6	6	84
Apr 2b	9	9	9	9	8	8	8	8	8	8	6	6	6	6	108
Hour Total	15	15	15	16	13	13	14	14	14	15	12	12	12	12	192
TOTAL		61				68					63				192
Fraction		0.508333333				0.453333333					0.42				
May 1a	5	5	5	6	5	5	5	5	5	6	4	4	4	4	68
May 2a	5	5	5	6	5	5	5	5	5	8	6	6	6	6	78
Hour Total	10	10	10	12	10	10	10	10	10	14	10	10	10	10	146
TOTAL		42				50					54				146
Fraction		0.35				0.333333333					0.36				
May 1b	2	2	2	3	2	2	3	3	3	5	4	5	5	5	46
May 2b	9	9	9	9	5	5	5	5	6	7	4	4	4	4	85
Hour Total	11	11	11	12	7	7	8	8	9	12	8	9	9	9	131
TOTAL		45				39					47				131
Fraction		0.3515625				0.24375					0.29375				

CLE to IAD Unreliability Measures for 2-Week Period – 8,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	0	0	0	0	0	0	0	0	0	1	1	1	1	1	5
Nov 2a	8	8	8	10	8	8	8	8	8	10	9	9	9	9	120
Hour Total	8	8	8	10	8	8	8	8	8	11	10	10	10	10	125
TOTAL		34				40					51				125
Fraction		0.283333333				0.266666667					0.34				
Nov 1b	1	1	1	2	2	2	2	2	2	3	3	3	3	3	30
Nov 2b	12	12	12	13	10	10	10	10	10	13	13	13	13	13	164
Hour Total	13	13	13	15	12	12	12	12	12	16	16	16	16	16	194
TOTAL		54				60					80				194
Fraction		0.45				0.4					0.533333333				
Dec 1a	5	5	5	8	5	5	5	5	5	6	6	6	6	6	78
Dec 2a	7	7	7	9	9	9	9	9	9	9	9	9	9	9	120
Hour Total	12	12	12	17	14	14	14	14	14	15	15	15	15	15	198
TOTAL		53				70					75				198
Fraction		0.441666667				0.466666667					0.5				
Dec 1b	10	10	10	13	13	13	13	13	13	13	7	8	8	8	152
Dec 2b	13	13	13	15	11	11	11	11	11	12	9	9	9	9	157
Hour Total	23	23	23	28	24	24	24	24	24	25	16	17	17	17	309
TOTAL		97				120					92				309
Fraction		0.7578125				0.75					0.575				
Jan 1a	9	9	9	11	8	8	8	8	8	11	10	10	10	10	129
Jan 2a	9	9	9	11	11	11	11	11	11	13	12	12	12	12	154
Hour Total	18	18	18	22	19	19	19	19	19	24	22	22	22	22	283
TOTAL		76				95					112				283
Fraction		0.633333333				0.633333333					0.746666667				
Jan 1b	9	9	9	10	8	8	8	8	8	8	7	7	7	7	113
Jan 2b	11	11	11	11	9	9	9	9	9	14	12	12	12	12	151
Hour Total	20	20	20	21	17	17	17	17	17	22	19	19	19	19	264
TOTAL		81				85					98				264
Fraction		0.6328125				0.53125					0.6125				
Feb 1a	5	5	5	8	7	7	7	7	7	8	5	5	5	5	86
Feb 2a	9	9	9	11	10	10	11	11	11	13	12	12	12	11	151
Hour Total	14	14	14	19	17	17	18	18	18	21	17	17	17	16	237
TOTAL		61				88					88				237
Fraction		0.508333333				0.586666667					0.586666667				
Feb 1b	7	7	7	7	7	7	7	7	7	7	6	6	6	6	94
Feb 2b	10	10	10	11	11	11	11	11	11	11	11	11	11	11	151
Hour Total	17	17	17	18	18	18	18	18	18	18	17	17	17	17	245
TOTAL		69				90					86				245
Fraction		0.663461538				0.692307692					0.661538462				
Mar 1a	6	7	7	8	6	6	6	6	6	9	9	9	9	8	102
Mar 2a	8	8	8	10	10	10	10	10	10	13	12	12	12	12	145
Hour Total	14	15	15	18	16	16	16	16	16	22	21	21	21	20	247
TOTAL		62				80					105				247
Fraction		0.516666667				0.533333333					0.7				
Mar 1b	11	11	11	12	10	10	10	10	10	12	11	11	11	11	151
Mar 2b	7	7	7	7	7	7	7	7	7	9	9	9	9	9	108
Hour Total	18	18	18	19	17	17	17	17	17	21	20	20	20	20	259
TOTAL		73				85					101				259
Fraction		0.5703125				0.53125					0.63125				
Apr 1a	9	9	9	9	6	6	6	6	6	7	7	7	7	7	101
Apr 2a	5	5	5	5	5	5	5	5	5	5	4	4	4	4	66
Hour Total	14	14	14	14	11	11	11	11	11	12	11	11	11	11	167
TOTAL		56				55					56				167
Fraction		0.466666667				0.366666667					0.373333333				
Apr 1b	4	4	4	7	5	5	6	6	6	7	5	5	5	5	74
Apr 2b	5	5	5	6	5	5	5	5	5	6	5	5	5	5	72
Hour Total	9	9	9	13	10	10	11	11	11	13	10	10	10	10	146
TOTAL		40				53					53				146
Fraction		0.333333333				0.353333333					0.353333333				
May 1a	4	4	4	5	2	2	2	2	2	3	3	3	3	3	42
May 2a	2	2	2	4	4	4	4	4	4	6	4	4	4	4	52
Hour Total	6	6	6	9	6	6	6	6	6	9	7	7	7	7	94
TOTAL		27				30					37				94
Fraction		0.225				0.2					0.246666667				
May 1b	2	2	2	3	2	2	3	3	3	4	3	4	4	4	41
May 2b	3	3	3	4	2	2	2	2	3	4	3	3	3	3	40
Hour Total	5	5	5	7	4	4	5	5	6	8	6	7	7	7	81
TOTAL		22				24					35				81
Fraction		0.171875				0.15					0.21875				

CLE to IAD Unreliability Measures for 2-Week Period – 6,000 feet

	0600-0700	0700-0800	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800	1800-1900	1900-2000	TOTAL
Nov 1a	0	0	0	0	0	0	0	0	0	1	1	1	1	1	5
Nov 2a	8	8	8	10	8	8	8	8	8	10	9	9	9	9	120
Hour Total	8	8	8	10	8	8	8	8	8	11	10	10	10	10	125
TOTAL			34				40					51			125
Fraction			0.283333333				0.266666667					0.34			
Nov 1b	1	1	1	1	1	1	1	1	1	3	3	3	3	3	24
Nov 2b	12	12	12	13	10	10	10	10	10	13	13	13	13	13	164
Hour Total	13	13	13	14	11	11	11	11	11	16	16	16	16	16	188
TOTAL			53				55					80			188
Fraction			0.441666667				0.366666667					0.533333333			
Dec 1a	4	4	4	4	2	2	2	2	2	3	3	3	3	3	41
Dec 2a	7	7	7	9	9	9	9	9	9	9	8	8	8	8	116
Hour Total	11	11	11	13	11	11	11	11	11	12	11	11	11	11	157
TOTAL			46				55					56			157
Fraction			0.383333333				0.366666667					0.373333333			
Dec 1b	10	10	10	13	13	13	13	13	13	13	7	8	8	8	152
Dec 2b	8	8	8	12	10	10	10	10	10	12	9	9	9	9	134
Hour Total	18	18	18	25	23	23	23	23	23	25	16	17	17	17	286
TOTAL			79				115					92			286
Fraction			0.6171875				0.71875					0.575			
Jan 1a	9	9	9	11	8	8	8	8	8	10	9	9	9	9	124
Jan 2a	8	8	8	11	11	11	11	11	11	13	12	12	12	12	151
Hour Total	17	17	17	22	19	19	19	19	19	23	21	21	21	21	275
TOTAL			73				95					107			275
Fraction			0.608333333				0.633333333					0.713333333			
Jan 1b	7	7	7	10	8	8	8	8	8	8	7	7	7	7	107
Jan 2b	11	11	11	11	9	9	9	9	9	14	12	12	12	12	151
Hour Total	18	18	18	21	17	17	17	17	17	22	19	19	19	19	258
TOTAL			75				85					98			258
Fraction			0.5859375				0.53125					0.6125			
Feb 1a	5	5	5	8	7	7	7	7	7	8	5	5	5	5	86
Feb 2a	8	8	8	11	9	9	10	10	10	12	11	11	11	11	139
Hour Total	13	13	13	19	16	16	17	17	17	20	16	16	16	16	225
TOTAL			58				83					84			225
Fraction			0.483333333				0.553333333					0.56			
Feb 1b	7	7	7	7	7	7	7	7	7	7	6	6	6	6	94
Feb 2b	10	10	10	10	10	10	10	10	10	10	10	10	10	10	140
Hour Total	17	17	17	17	17	17	17	17	17	17	16	16	16	16	234
TOTAL			68				85					81			234
Fraction			0.653846154				0.653846154					0.623076923			
Mar 1a	5	6	6	8	6	6	6	6	6	8	8	8	8	7	94
Mar 2a	7	7	7	9	9	9	9	9	9	13	12	12	12	12	136
Hour Total	12	13	13	17	15	15	15	15	15	21	20	20	20	19	230
TOTAL			55				75					100			230
Fraction			0.458333333				0.5					0.666666667			
Mar 1b	10	10	10	11	8	8	8	8	8	11	10	10	10	10	132
Mar 2b	4	4	4	5	5	5	5	5	5	7	7	7	7	7	77
Hour Total	14	14	14	16	13	13	13	13	13	18	17	17	17	17	209
TOTAL			58				65					86			209
Fraction			0.453125				0.40625					0.5375			
Apr 1a	8	8	8	8	4	4	4	4	4	5	5	5	5	5	77
Apr 2a	5	5	5	5	4	4	4	4	4	4	2	2	2	2	52
Hour Total	13	13	13	13	8	8	8	8	8	9	7	7	7	7	129
TOTAL			52				40					37			129
Fraction			0.433333333				0.266666667					0.246666667			
Apr 1b	3	3	3	5	4	4	5	5	5	6	5	5	5	5	63
Apr 2b	4	4	4	4	2	2	2	2	2	3	2	2	2	2	37
Hour Total	7	7	7	9	6	6	7	7	7	9	7	7	7	7	100
TOTAL			30				33					37			100
Fraction			0.25				0.22					0.246666667			
May 1a	4	4	4	4	1	1	1	1	1	2	2	2	2	2	31
May 2a	2	2	2	2	2	2	2	2	2	3	2	2	2	2	29
Hour Total	6	6	6	6	3	3	3	3	3	5	4	4	4	4	60
TOTAL			24				15					21			60
Fraction			0.2				0.1					0.14			
May 1b	2	2	2	3	1	1	2	2	2	3	3	4	4	4	35
May 2b	0	0	0	0	0	0	0	0	1	2	1	1	1	1	7
Hour Total	2	2	2	3	1	1	2	2	3	5	4	5	5	5	42
TOTAL			9				9					24			42
Fraction			0.0703125				0.05625					0.15			

Vita

The author, Melinda M. Gates, was born in Anchorage, Alaska on January 27, 1980. She graduated from Virginia Polytechnic Institute and State University (Virginia Tech) in May 2002 with a Bachelor's of Science in Industrial and Systems Engineering. She remained at Virginia Tech to earn her Master's of Science in Industrial and Systems Engineering with a concentration in Operations Research. Melinda was enrolled in the International Dual Master's Degree Program, where she spent a semester at Ecole des Mines de Nantes in Nantes, France. She will earn dual degress from Virginia Tech and Ecole des Mines. Melinda will continue her career in Reston, Virginia as an analyst for Accenture.