ABSTRACT

SOUTHWELL, JOHN THOMAS. Equivalent Safe Response Model for Evaluating the Closed Loop Handling Characteristics of UAS to Contribute to the Safe Integration of UAS into the National Airspace System. (Under the direction of Dr. Charles E. Hall Jr.).

As unmanned aerial systems are introduced into the national airspace system, specific outer-loop handling qualities and performance requirements are needed to maintain a safe environment. Current sources on the subject have yet to determine appropriate concrete requirements for this situation. Equivalent Safe Response attempts to aid in this area by defining necessary and applicable requirements for unmanned aerial system and by developing a method to evaluate a given unmanned aerial system's performance and outer-loop handling qualities. The Equivalent Safe Response requirements were primarily derived from the Federal Aviation Administration's Instrument Rating Practical Test Standards. Equivalent Safe Response provides a way of evaluating an unmanned aerial system by using the Piccolo Software-in-the-Loop simulations, a specific obstacle course designed to stress each requirement, and a MATLAB program that was written to analyze and evaluate the performance of the unmanned aerial system. As a test case, the ArcWulf unmanned aerial system, flown by the Aerial Robotics Club at North Carolina State University was evaluated using the Equivalent Safe Response model. The results of the evaluation indicated that the ArcWulf could successfully comply with the requirements in the three environmental conditions that were tested. The evaluation also highlighted areas of performance where the ArcWulf was the weakest. This allowed theoretical modifications to be made to the airframe and the autopilot parameters to improve the outer-loop performance in these areas.

Equivalent Safe Response Model for Evaluating the Closed Loop Handling Characteristics of
UAS to Contribute to the Safe Integration of UAS into the National Airspace System

by
John Thomas Southwell

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Aerospace Engineering

Raleigh, North Carolina

2010

APPROVED BY:

Dr. Ashok Gopalarathnam

Dr. Sharon R. Lubkin

Dr. Charles E. Hall Jr.
Chair of Advisory Committee

DEDICATION

This research is dedicated to my grandfathers and great grandfathers. They have shaped my life and passed down the gifts that I have used to become an engineer. Thank you.

BIOGRAPHY

John Southwell was born in 1985 in Wilkesboro, North Carolina to Paul and Jackie Southwell. He has one older brother, Adam and sister-in-law, Aubrey. He moved with his family to Currituck, North Carolina in 2001, and graduated from Currituck County High School in 2004. In August of that year, John began attending North Carolina State University and majored in Aerospace Engineering. As a sophomore, John was selected to be an officer candidate in the United States Navy Nuclear Propulsion Officer Candidate Program. In the fall of 2007, John was selected to be a team leader of one of three teams in the aircraft senior design course. He graduated Cum Laude with a Bachelors Degree in 2008. After graduation, he reported to Officer Candidate School in Newport, Rhode Island. During training John developed tendonitis in his right knee and spent the next six months in physical therapy. In December of 2008, John was honorably discharged from the United States Navy. In January, he returned to North Carolina State University to pursue his Master's Degree in Aerospace Engineering, with a focus on airworthiness of unmanned aerial systems. John became the lead graduate student on a contract with NAVAIR to develop a model to evaluate the outer-loop handling qualities of unmanned aerial systems to aid in the safe integration into the national airspace system. John currently resides in Raleigh, with his yellow Labrador Tybee.

ACKNOWLEDGMENTS

I would like to take this opportunity to give a special thanks to my family. Mom and Dad, you have always been there to offer a word of advice or just an ear to listen. Thank you so much for all the sacrifices you have made throughout my life so that I could follow my dreams and focus on my education. I am so fortunate to have such great parents. Adam and Aubrey, it has been a blessing having you in Raleigh. You have become more than just family to me, but friends also. Thank you for all the great times, whether it was playing games all night or watching movies, you guys were always there to help me relax and have fun.

I would also like to thank my advisor Dr. Charles Hall. He has always pushed me to do the best work that I could, both in my undergraduate and graduate career. He has given me numerous opportunities to learn and gain experience that has made me a better engineer and that I will be able to use for the rest of my career. I would also like to thank my flight instructor and friend, Cecil Edgerton. His encouragement and confidence in my abilities has pushed me to succeed, not only in learning to fly, but in other areas of my life as well. I have immensely enjoyed the hours in the air and time afterwards at Zaxby's, he has always been able to make me laugh. I would also like to give a special thank you to Pops for making it possible for me to follow my dreams and learn to fly.

Finally, I would like to thank my girlfriend and best friend, Myra. You have been one of the best surprises in my life. Who would have thought after all those years since we had sat in the same classrooms in high school? You have truly been a blessing in my life and have always been there for me, even when I was frustrated and grumpy from work. You have made me so happy and I cannot wait to see what life has in store for us.

TABLE OF CONTENTS

LIST OF FIGURES	x
LIST OF TABLES	xi
LIST OF ACRONYMS	xii
Chapter 1 Introduction.....	1
1.1 Current Guidance for Handling Qualities of Manned and Unmanned Aircraft	3
1.2 Expectations of Air Traffic Control	8
1.2.1 Flight Rules	9
1.2.2 Performance Based Navigation and Required Navigation Performance	12
Chapter 2 TCAS and UAS.....	17
2.1 Background of TCAS	17
2.2 TCAS Logic	18
2.3 TCAS and UAS	21
Chapter 3 Equivalent Safe Response Requirements.....	26
3.1 Altitude	26
3.2 Heading, Airspeed, and Bank Angle	27
3.3 System Performance	28

3.4	Constant Requirements	28
3.5	Flight Phase Requirements	29
3.5.1	Departure Requirements	29
3.5.2	En-route Requirements	31
3.5.3	Arrival Requirements	31
3.5.4	Approach Requirements	32
3.6	Environmental Effects and ESR	36
Chapter 4	Equivalent Safe Response Model	37
4.1	The ESR Course	37
4.1.1	Climb Rate and Descent Rate Tests	40
4.1.2	Turn Rate Test	40
4.1.3	Straight and Level Test	40
4.1.4	Approach Test	41
4.1.5	RNP Test	41
4.1.6	Departure Phase	42
4.1.7	En-route Phase	44
4.1.8	Arrival Phase	45

4.1.9	Approach Phase	47
4.2	ESR MATLAB Code	49
Chapter 5	ESR evaluation of the ArcWulf	63
5.1	Modification of the ArcWulf model to reduce flight path deviations.....	65
Chapter 6	Conclusion	71
References		73
APPENDIX		76
Appendix A	ESR MATLAB Code.....	77
A.1	Main ESR Code	77
A.2	TAS Target Function	146
A.3	Flight Phase Analysis Function	147
A.4	ESR climb Function.....	150
A.5	ESR Straight Level Function	151
A.6	ESR Turn Function	152
A.7	ESR Final Approach Function.....	153
A.8	ESR Missed Approach Function	155
Appendix B	ArcWulf ESR Results	157

B.1	Calm Conditions Results	157
B.2	Light Turbulence Condition	182
B.3	14 knot wind from the East Condition.....	207
B.4	ArcWulf model with increased weight/inertia in light turbulence condition	232
B.5	ArcWulf model with increased weight/inertia and autopilot tuning in light turbulence condition.....	257
B.6	Comparison of Original and Modified ArcWulf ESR Results	282

LIST OF FIGURES

Figure 1.1: Cooper-Harper Rating Scale	4
Figure 1.2: Total System Error and components	13
Figure 1.3: Visualization of flight path with containment area	14
Figure 1.4: Typical RNP levels from en-route to approach.....	15
Figure 3.1: ESR Final approach containment volume	33
Figure 3.2: Side view of final approach containment volume	35
Figure 3.3: Top view of final approach containment volume.....	35
Figure 4.1: The ESR course as viewed in the Piccolo software	39
Figure 4.2: The ESR course altitude profile	39
Figure 4.3: Departure phase of the ESR course	42
Figure 4.4: En-route phase of the ESR course	44
Figure 4.5: Arrival phase of the ESR course	46
Figure 4.6: Approach phase of the ESR course	47
Figure 5.1: ArcWulf course tracking without (left) and with (right) turbulence	64
Figure 5.2: ArcWulf course tracking without (left) and with (right) added weight	68

LIST OF TABLES

Table 1.1: Typical RNP levels	14
Table 1.2: Assumed FTE values	15
Table 1.3: PBN vertical TSE for level flight	16
Table 1.4: PBN vertical TSE for climbs/descents along a specified vertical profile.....	16
Table 3.1: ESR requirements for maintaining altitude in level flight.....	27
Table 3.2: ESR requirements for maintaining altitude in climbing/descending flight	27
Table 3.3: ESR constant requirements.....	29
Table 4.1: ESR simulation lateral deviation requirements	41
Table 4.2: Flight phase legs and ESR tests	57
Table 5.1: Modified ArcWulf Weight and Inertias.....	67
Table 5.2: Average Straight and Level Performance Comparison with Additional Weight ..	68
Table 5.3: Piccolo Lateral Gains Before and After Tuning	69
Table 5.4: Straight and Level Performance Comparison with Additional Weight and Tuning	69

LIST OF ACRONYMS

AGL	Above Ground Level
AIM	Aeronautical Information Manual
ALIM	Amount of Vertical Separation
ATC	Air Traffic Control
CDI	Course Direction Indicator
CPA	Closest Point of Approach
ESR	Equivalent Safe Response
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulation
FMS	Flight Management System
FTE	Flight Technical Error
GA	General Aviation
GPS	Global Positioning System
IAS	Indicated Airspeed
IFR	Instrument Flight Rules
ILS	Instrument Landing System
MTOW	Maximum Takeoff Weight
NACA	National Advisory Committee for Aeronautics
NAS	National Airspace System

NSE	Navigation System Error
OAT	Outside Air Temperature
PBN	Performance Based Navigation
PDE	Path Definition Error
RA	Resolution Advisory
RC	Radio Controlled
RNP	Required Navigation Performance
SIL	Software-in-the-Loop
SSA	Sense and Avoid
STC	Supplemental Type Certificate
TA	Traffic Advisory
TAS	True Airspeed
TCAS	Traffic Collision Avoidance System
TSE	Total System Error
UAS	Unmanned Aerial System
USAR	Unmanned Aerial Vehicles Systems Airworthiness Requirements
VFR	Visual Flight Rules

Chapter 1 Introduction

In recent years there has been an increased interest and demand for operations of Unmanned Aerial Systems (UAS) in the National Airspace System (NAS). To date the military has been the prime user of UAS. In this point in time, the military is bringing home units that have both been trained and operated UAS only overseas in warzones. For these units to remain combat ready they must be allowed to operate their UAS state side, and with the limited number of controlled ranges available a need exists for the military to be able to have access to the NAS to operate and train with UAS. The military is not the only entity that currently uses UAS. UAS have also been used by U.S. Customs to patrol borders. Other organizations are also currently researching the possibility of using UAS for such things as wildfire reconnaissance and law enforcement.

UAS are often preferred in applications that are considered too “dull, dirty, or dangerous” for manned aircraft. As these applications for UAS continue to be developed for the military, commercial, and civil sectors the demand for UAS access to the NAS will only increase. Currently, the NAS is one of the most complex aviation systems in the world, and on average, about 50,000 flights traverse the NAS each day. As UAS are introduced into this already busy and crowded airspace significant safety concerns arise. One of the major concerns is how UAS will interact with other traffic in the NAS.

Before UAS are allowed increased access to the NAS, it must be shown that UAS are capable of operating alongside manned aircraft without increased risk to third-parties in the air or on the ground. For this reason, the Flight Research group at North Carolina State University was tasked to determine the closed outer-loop handling qualities required for the safe interaction between UAS and Air Traffic Controllers (ATC). This report presents the model and set of requirements referred to as Equivalent Safe Response (ESR) that has been developed for evaluating UAS closed outer-loop handling qualities. It is the hope of the author that this research will aid in the safe integration of UAS into the NAS.

In the remainder of Chapter 1, the current available resources on manned and unmanned aircraft handling qualities will be reviewed and will show how ESR contributes in this area. Chapter 2 will discuss the Traffic Collision Avoidance System (TCAS) in the context of unmanned aviation. Next, Chapter 3 will present the ESR requirements, explain how the requirements were chosen, and the size and types of UAS that could be evaluated with the ESR model. In Chapter 4, a model designed to test UAS with the ESR requirements is presented along with the results from an evaluation of a small UAS operated by the Aerial Robotics Club at North Carolina State University. Finally, the ESR model and MATLAB implementation are presented in Appendix A, along with the MATLAB output files in Appendix B.

1.1 Current Guidance for Handling Qualities of Manned and Unmanned Aircraft

Extensive research on aircraft handling qualities has been ongoing since the Wright Brothers' first flight in 1903. The National Advisory Committee for Aeronautics (NACA) published several of the first technical reports on aircraft flying qualities, for example, "Flight Measurements of the Dynamic Longitudinal Stability of Several Airplanes and a Correlation of the Measurements with Pilots' Observations of Handling Characteristics". In this report, the dynamic longitudinal stability characteristics of eight airplanes with well known handling qualities was measured to attempt to correlate the measured stability with pilots' opinions of the aircraft's handling qualities. To date, handling qualities of manned aircraft are still ultimately evaluated by pilots. These evaluations are typically qualitative instead of quantitative, so in 1969 a scale called the Cooper-Harper Qualities Rating Scale was developed. This scale enabled qualitative judgments about an aircraft's handling qualities to be assigned a quantitative value between 1 and 10 (See Fig. 1.1 [1]). The Cooper-Harper Scale is currently the standard in manned aviation for measuring handling qualities.

Figure 1.1: Cooper-Harper Rating Scale

This method becomes very difficult to apply to UAS since a human pilot is not aboard the aircraft and also would not normally be flying the aircraft. Some UAS still require external pilots to land the aircraft; however, the current trend in the UAS community is to make all portions of flight automated. The military has several sets of guidelines for manned aviation, which include, MIL-STD-1797A entitled, “Flying Qualities of Piloted Aircraft”, and MIL-HDBK-516 entitled, “Airworthiness Certification Criteria”. MIL-HDBK-516, which was

released in 2004, contains top level recommendations, for example, “Verify that the design exhibits safe vehicle flying characteristics...” [2]. The question that arises from guidelines like these is: what actually qualifies as safe? This is one of the questions that this research attempts to answer by providing clear requirements for UAS.

In manned aviation, closed-loop handling qualities are normally evaluated in piloted simulations and flight tests. The goal of these tests are to achieve consistent Cooper-Harper ratings, which is accomplished by creating maneuver libraries of tasks that are to be performed along with the success criteria. This is similar to the method that is used to evaluate UAS closed-loop handling qualities in ESR, as will be discussed in the following chapters.

In 1990, Robert K. Heffley of Robert Heffley Engineering performed research on the outer-loop control factors for carrier operated aircraft, specifically during the approach phase [3]. The primary question that Heffley attempted to answer was: what design factors affect the outer-loop performance of the aircraft? This research was valuable because if the relationships between design factors and outer-loop performance were known, the aircraft could be designed with a higher confidence of having the outer-loop performance that was desired and could possibly prevent a design modification needed later to improve the outer-loop performance issues discovered in flight tests. Heffley concluded that the current Navy flying qualities requirements did not explicitly address outer-loop control factors. The majority of the requirements currently used are focused on inner-loop handling qualities.

ESR provides a way to evaluate the outer-loop performance of a UAS in either a simulation or flight test setting.

The question that arises when transitioning to unmanned aircraft handling qualities is: how does the absence of a pilot affect the requirements? This is the issue that must be addressed when designing a UAS since the system will most likely be flown primarily by an autopilot. Using exclusively all the traditional manned aviation handling qualities while designing a UAS would be unnecessary since most manned handling qualities requirements are meant to keep the pilot comfortable. It could also limit the potential of the systems performance.

As the number of UAS operations has exponentially increased in the past decade, so has the need for a set of guidelines and requirements. To date these requirements have been adaptation from manned aviation and have left some of the unique aspects of unmanned aviation unaddressed. In 2009, STANAG 4671 entitled, “Unmanned Aerial Vehicles Systems Airworthiness Requirements (USAR)”, was released by the NATO Standardization Agency. This document provides guidelines for fixed wing UAS systems with a maximum takeoff weight (MTOW) of more than 150 kg (330 lbs) and less than 20,000 kg (44,090 lbs). In general, the STANAG 4761 gives brief overviews of handling qualities requirements without providing any qualitative reasoning. Some examples of these are listed below.

“The UAV must be safely controllable and maneuverable during all flight phases...” [4]

“When operated in the automatic control mode the UAV should be shown to have acceptable controllability, maneuverability and stability characteristics throughout the flight envelope protection maintained by the flight control system...” [5]

The STANAG 4671 provides a framework for UAS handling qualities requirements, however, actual quantities are left to be agreed upon between the applicant and certifying authority. One of the goals of ESR is to fill in these gaps.

In 2004, the Office of the Secretary of Defense released the “Airspace Integration Plan for Unmanned Aviation”. In this document, three categories were proposed for defining standards for UAS. Larger UAS, like the Global Hawk and Predator, would be in category 3 (CAT III), and would be treated similarly to manned aircraft. CAT III would operate according to Visual Flight Rules (VFR) and Instrument Flight Rules (IFR) regulations and procedures and would be required to meet requirements in 14 CFR Part 91. CAT III UAS would also have access to all airspace classifications. CAT II would consist of UAS such as the Pioneer, Shadow, and Hunter and would be treated similarly to ultra-lights, light-sport, or restricted category aircraft. This category would have an equivalent level of safety to certified aircraft with a slightly lower level of reliability. The requirements that would have to be met would be similar to 14 CFR Part 103, 91, and 101. The required operating conditions would be Visual Meteorological Conditions, similar to the Restricted Category Aircraft operations in the absence of ATC. CAT II would have access to airspace classified as E, G, and non-joint-use D. The final category, CAT I, would consist of UAS such as the

Raven and Dragon Eye. These UAS would be treated similarly to RC model aircraft, whose requirements are outlined in AC 91-57. These include only VFR line-of-sight operations in uncontrolled airspace. This would be airspace classified as G below 1,200 ft AGL. This approach is similar to the one that was taken to develop ESR, which consists of requirements directly from such sources as 14 CFR 91, which will be discussed in greater detail in the following sections. The focus of this research was primarily for UAS in CAT I and II, however, ESR does have the potential to be applied to larger UAS.

Even though the handling qualities requirements for UAS are a large portion of the airspace integration problem, it is not the only factor that should be considered. Another consideration is the actual airspace system that the UAS will be integrated into. In the case of the NAS, consideration must be given to the types of air traffic a UAS might encounter. In the NAS, ATC controls the majority of the air traffic from departure to en-route to approach and landing. To be able to be successfully integrated into the NAS the expectations of ATC must be known first.

1.2 Expectations of Air Traffic Control

Air Traffic Control is responsible for directing aircraft operations both in the air and on the ground, to separate aircraft to prevent collisions, to organize the flow of traffic to decrease delays, and also to provide information and support to pilots. When a pilot receives instructions from ATC it is normally in the form of an altitude, heading, and airspeed to fly. The ability of a pilot to maintain a given flight path becomes increasingly important when

operating near busy airports, since any deviation could cause the aircraft to cross another aircraft's flight path. The ability of a manned aircraft to maintain a given flight path is a combination of pilot skill and the open loop handling qualities of the aircraft. Similarly, the ability of a UAS to maintain a given flight path is a combination of the autopilot, flight director, stability augmentation system, etc., and the open loop handling qualities of the aircraft. As long as a UAS is able to perform and follow ATC instruction as well as or better than a manned aircraft, proper separation between aircraft should be able to be maintained without greatly increasing workload on the air traffic controller.

1.2.1 Flight Rules

The Federal Aviation Administration (FAA) created two sets of regulations for manned aviation, Visual Flight Rules (VFR) and Instrument Flight Rules (IFR). These sets of rules specify what weather conditions a pilot can operate an aircraft under, as well as the procedures that should be followed to operate the aircraft safely with other traffic in the NAS.

To operate an aircraft under VFR, the weather conditions, in general, must be clear enough to allow the pilot to see outside the cockpit, to control the aircraft's attitude, to navigate, and to avoid obstacles and other aircraft. Extensive research is currently being conducted into See-and-Avoid technology for UAS. Ideally, this technology would allow a UAS to "see" other aircraft and avoid them. However, a See-and-Avoid technology that can perform as well as the human eye has not been developed, in part due to the difficulty in quantifying the criteria.

This means that VFR is not applicable to UAS operations because of the fact that UAS are primarily flown by autopilots, use GPS navigation, and the “pilot” or “operator” in command generally uses top-down map views in the ground stations to send high-level commands (fly to this waypoint, loiter over this position, return to base, etc.).

Instrument Flight Rules is a set of regulations and procedures for operating an aircraft by only referring to the aircraft instrument panel (i.e. not using outside visual cues). Currently, all scheduled airline flights are operated under IFR, and is also used for the limited UAS operations that are allowed. Instrument Flight Rules are applicable to UAS because of the nature that UAS are operated. The specific procedures and regulations for IFR operations are found in the Federal Aviation Regulations, or FARs. These regulations govern all aviation activities in the NAS, but the IFR specific regulations have been collected in two sources by the FAA, the Instrument Flying Handbook and the Instrument Procedures Handbook. The pertinent requirements from each handbook are listed below.

- Use of the standard rate turn ($3^\circ/\text{s}$) when executing a majority of instrument procedures [6].
- Rate of change of altitude in climbs and descents should vary between 500 ft/min and 1,000 ft/min [7].
- Normal standard rate turn bank angle should be less than 30° [8].
- Normal glide slope angle is 3° , but could be as high as 4.5° for obstacle clearance [9].

These requirements are applicable to all flight phases (departure, en-route, arrival, and approach) and can be directly applied to UAS because the requirements are not dependent on a human pilot flying the aircraft.

To obtain an Instrument Rating, a pilot must pass the Instrument Rating Practical Test administered by the FAA. This test includes a simulated IFR flight to demonstrate the pilot's ability to meet the Instrument Rating Practical Test Standards, which are listed below [10].

- Straight and level flight, climbs, turns, and descents
 - Maintains altitude within ± 100 ft during level flight
 - Maintains headings within $\pm 10^\circ$
 - Maintains airspeed within ± 10 knots
 - Maintains $\pm 5^\circ$ of the required bank angle during turns
- Prior to beginning the final approach segment
 - Maintains altitude within ± 100 ft
 - Maintains headings within $\pm 10^\circ$
 - Maintains airspeed within ± 10 knots
 - Allows less than $\frac{3}{4}$ -scale deflection of the Course Direction Indicator (CDI)
- While on the final approach segment
 - Maintains airspeed within ± 10 knots
 - Allows less than $\frac{3}{4}$ -scale deflection of the Course Direction Indicator (CDI)

- During a missed approach procedure
 - Maintains altitude within ± 100 ft
 - Maintains headings within $\pm 10^\circ$
 - Maintains airspeed within ± 10 knots

The Instrument Flight Rules and Instrument Rating Practical Test Standards provide both performance and handling requirements that can be directly applied to UAS since the requirements must be met by the complete system in manned aviation (aircraft handling qualities, performance, and pilot abilities). In this case a pilot's ability to maintain a desired flight path can be directly correlated to that of an autopilot's. The Instrument Flight Rules provide many of the essential standards needed to keep operations in the NAS safe.

However, the current development of the NextGen system, more specifically, Performance Based Navigation (PBN) could further increase safety, while also increasing the capacity for air traffic.

1.2.2 Performance Based Navigation and Required Navigation Performance

The development of PBN and Required Navigation Performance (RNP) has marked a shift in aviation regulations from sensor-based to performance-based navigation. Until PBN has been completely implemented in the NAS, some airspace requires aircraft to be equipped with specific navigation equipment with known accuracies. Operating in densely trafficked airspace would require more expensive and accurate equipment. Performance Based Navigation differs from this by defining system level requirements. Aircraft systems

performance requirements are defined in terms of accuracy, integrity, availability, continuity, and functionality, and vary depending on the proposed operations in a particular airspace.

Primarily, PBN specifies allowable deviations from a desired flight path in the lateral, vertical, and longitudinal directions. These allowable deviations are specified in terms of Total System Error (TSE), which is illustrated in Fig. 1.2 [11], below, which shows the lateral TSE and the component errors that comprise it; however, the same definitions can be used for the vertical direction as well. Defining allowable deviations creates a containment area (See Fig. 1.3) around the desired flight path that the aircraft must remain inside [12].

Figure 1.2: Total System Error and components

Figure 1.3: Visualization of flight path with containment area

Allowable deviations vary along the flight path depending upon the airspace and flight phase. The typical RNP levels for each flight phase are listed below, which are equal to the allowable lateral deviation in nautical miles (See Table 1.1). Figure 1.4 illustrates the typical increase in RNP from the en-route to approach phase [13].

Table 1.1: Typical RNP levels

Flight Phase	RNP Level
Departure	1.0
En-route	2.0
Arrival	1.0
Approach	0.3

Figure 1.4: Typical RNP levels from en-route to approach

The ICAO Manual on Required Navigation Performance, which was the initial research effort that has become PBN, includes the allowable lateral FTE values which are listed below [14]. As specified in the PBN manual, these lateral specifications must be met on a 3σ basis or 95% of the time.

Table 1.2: Assumed FTE values

Flight Phase	FTE (NM)
Oceanic	0.25
En-route	0.25
Terminal	0.25
Approach	0.125

In the vertical direction PBN specifies TSE as a function of altitude and either level flight or climbs/descents. The vertical TSE for level flight is presented in Table 1.3. When an aircraft is tracking a specified vertical profile or angled flight path (climbing or descending flight), the TSE allowed is larger than for level flight, as presented in Table 1.4. The allowed deviations in the longitudinal direction (distance to waypoint) have not yet been defined in PBN, but are planned to be a part of NextGen.

Table 1.3: PBN vertical TSE for level flight

Altitude	TSE
$\leq 5,000$ ft	± 158 ft
$> 5,000$ ft	± 245 ft

Table 1.4: PBN vertical TSE for climbs/descents along a specified vertical profile

Altitude	TSE
$\leq 5,000$ ft	± 224 ft
$> 5,000$ ft $\leq 10,000$ ft	± 335 ft
$> 10,000$ ft	± 372 ft

The Flight Rules given by the FAA and the Performance Based Navigation requirements used in conjunction with ATC keep one of the busiest airspace systems in the world a safe and controlled space. As UAS are introduced into the NAS, airspace will become increasingly busy and therefore has caused much debate about the collision avoidance capabilities UAS should have, which is the topic of the next chapter.

Chapter 2 TCAS and UAS

A major concern for many about allowing UAS increased access to the NAS is the increased risk of midair collisions. With UAS ranging in size from ounces to the size of a Boeing 737, and without a pilot onboard to visually acquire other traffic the risk of a midair collision is very real. It is due to this concern that the FAA published FAA Order 7610.4 regarding remotely operated aircraft (ROA), “ROA flying in the NAS are required to be equipped with a collision avoidance system that provides an equivalent level of safety, comparable to see-and-avoid requirements for manned aircraft.” Even though this research does not specifically address the collision avoidance issue, the ability of two aircraft to perform an evasive maneuver to avoid one another does depend majorly on each aircraft’s particular handling qualities and performance envelope.

In manned aviation a system called Traffic Alert and Collision Avoidance System (TCAS) has helped reduce the number of midair collisions for more than 20 years and is now required on all large passenger aircraft. A logical first step to trying to equip a UAS with a collision avoidance system is to start with a system that has already been operationally tested and accepted as safe by the FAA and the industry.

2.1 Background of TCAS

The initial concept and development of an airborne collision avoidance system began in the 1950’s. The demand for such a system was fueled by a series of midair collisions in the

United States involving commercial, military, and private aircraft. In the 1970's the Beacon Collision Avoidance System (BCAS) was developed and installed on all commercial, military, and a large number of general aviation aircraft. BCAS was successful in decreasing the number of midair collisions, and in 1978 the FAA initiated the development of TCAS. In 1981, The FAA decided to use BCAS as the foundation of TCAS with some additional capabilities.

After the initial development and testing of TCAS, in service evaluations began in 1982 with Piedmont Airlines. During these evaluations an updated version of TCAS was developed, TCAS II, and was eventually certified under a Supplemental Type Certificate (STC) and deemed ready for a more widespread implementation. In 1991 the largest operational evaluation of TCAS began, which was known as the TTP [15]. This has led to updated versions of TCAS, the most recent being Version 7.1. Today, the FAA requires TCAS I or II to be installed on any aircraft with 10 to 30 seats, and TCAS II is required internationally in aircraft with more than 30 seats, or weighing more than 15,000 kg [16].

2.2 TCAS Logic

The goal of TCAS is to create a protected volume around an aircraft, which is accomplished through the TCAS logic. At the heart of the TCAS logic is the tau concept, developed by Dr. John S Morrell of Bendix. The concept of tau uses time instead of distance to an encounter. To implement this concept, TCAS interrogates transponders in the vicinity and tracks slant range, altitude, and bearing, depending on the replies it receives. If the TCAS

logic concludes that the flight path of another aircraft will encroach on its own protected volume, a traffic advisory (TA) or resolution advisory (RA), depending on the tau, or the time to the closest point of approach (CPA), will be given to the pilot. If an RA is needed, TCAS has been designed to only suggest changes in altitude until a set amount of vertical separation (ALIM) is attained between the two aircraft. ALIM varies from 300 – 700 ft depending on the operating altitude. Heading changes are not considered in an RA from the TCAS logic, however in version 7, the horizontal distance between two aircraft is also considered in order to reduce unnecessary alerts by TCAS.

TCAS is required to provide reliable surveillance of air traffic out to 14 NM, but only considers “proximate traffic” to be within 6 NM and ± 1200 ft. TCAS II was also only designed to operate in traffic densities of 0.3 aircraft per sq. NM or less [17]. This is equivalent to 24 aircraft in a 5 NM radius and was the highest envisioned traffic density of the next 20 years in 2000.

Once TCAS concludes that an RA is needed, the first step is to select the sense, up or down, which will provide the most vertical separation. Once this is selected, the RA strength is selected that will be the least disruptive to the existing flight path and a vertical speed or pitch angle to fly is displayed to the pilot. This is initially a 0.25g vertical acceleration until attaining a vertical rate of 1500 ft/min. If a second RA is needed during the same conflict, the RA will be strengthened to a 0.35g vertical acceleration until attaining a vertical rate of 2500 ft/min. However, if needed the RA strength can be increased up to 4400 ft/min. If the

intruding aircraft is also equipped with TCAS, the maneuver information that is being presented to the pilot will be sent to the TCAS unit onboard the intruding aircraft so that the maneuver can be coordinated. If the intruding aircraft is only equipped with a transponder TCAS will still present the pilot with TAs and RAs, but any RA maneuver will not be coordinated with the other aircraft. If an intruding aircraft is not equipped with a transponder, TCAS will not provide any advisory to the pilot.

As discussed in the previous section, TCAS was tailored primarily for use in commercial aircraft. This led to requirements in vertical acceleration and rates that are possible for the majority of commercial aircraft while providing a balance between safety and the frequency of TCAS alerts. However, these requirements can become unattainable by many general aviation (GA) aircraft in certain flight conditions. For this reason, TCAS has been designed to be configurable for a particular aircraft's performance limitations. These limitations will inhibit an RA that cannot be attained, and can be provided by program pins in the TCAS connector or from an input from a flight management system (FMS) [15].

During an RA maneuver, excursions from the assigned altitude should be 300 – 500 ft to resolve any conflict [15]. However, from operational experience these deviations have been observed to be much larger, and have been primarily contributed to many pilots' unwillingness to decrease the aircraft's vertical rate after receiving a reduced RA from TCAS.

Although TCAS has been shown to reduce the overall number of midair collisions to 10% of that without TCAS, it is not a perfect system. Some midair collisions can be induced by TCAS. This could happen when an aircraft is performing an RA maneuver to avoid one aircraft that places it in the path of another. TCAS has also had an adverse effect on controllers and the ATC system. TCAS operates on the same frequency as the ATC system, which is 1030 and 1090 MHz. To avoid interference with the ATC radars, the TCAS logic has been programmed to decrease the interrogation rate and power allocation as the number of TCAS units in an area increases. So as the traffic density increases, the rate at which TCAS updates decreases and the chance of a midair collision increases. TCAS can also cause confusion between controllers and pilots during an RA maneuver, which is commonly due to a lack of communication between the controller and pilot.

2.3 TCAS and UAS

Due to the success of TCAS in manned aviation it is an obvious choice to attempt to apply the same system to unmanned aviation and could be a first step towards a “sense and avoid” system. Two of the major concerns with a TCAS unit onboard a UAS are the performance limitations of some UAS and the communication latency between the UAS and the ground station.

A study was conducted at MIT in 2006 on the safety benefits of adding TCAS to the Global Hawk by simulating encounters between a Global Hawk and other traffic [18]. Even though the Global Hawk could not meet the vertical acceleration and rate requirements of TCAS in

all portions of its flight envelope it was concluded that these performance limitations had only a slight effect on the risk ratio, which was defined as the reduction of collision risk due to the use of TCAS. However, the ability to inhibit RA rates in TCAS was not included in the simulation. In this study, after millions of simulated encounters between a Global Hawk and a conventional aircraft, the risk ratio of the Global Hawk with TCAS was found to be 0.003 – 0.079 and was compared to that of a conventional aircraft which was 0.004 – 0.058. The study concluded that the major issue was communication latency. Any delay longer than 15 seconds resulted in the use of TCAS to cause an increase in the risk of a collision over that of just using a Mode S transponder.

It is possible that the risk ratio for the Global Hawk with TCAS could be reduced further if the RA rates that were not attainable by the Global Hawk were inhibited in the TCAS unit. This would further increase safety during situations when a UAS with TCAS encountered another TCAS equipped aircraft or another aircraft equipped only with a transponder by enabling TCAS to plan the required RA maneuver with vertical accelerations and rates that the aircraft could attain. This is described in AC 20-151A, which reads, “The collision avoidance maneuvers posted as RAs by TCAS II assume an aircraft’s ability to safely achieve them. If it’s likely the required response to CLIMB and INCREASE CLIMB RAs are beyond the performance capability of the aircraft, then TCAS II must know beforehand so it can change strategy and issue an alternative RA [19].” When rates are not inhibited, TCAS will plan, and in the case of a TCAS/TCAS encounter, coordinate an RA maneuver

based on the standard rates. If either aircraft cannot meet these rates it could lead to a reduced amount of vertical separation or even a midair collision. If needed, safety could be further increased by increasing the tau at which a TA and RA are given. This would give aircraft more time to maneuver, but would also increase the frequency of TCAS alerts. Currently, tau ranges from 15 to 35 seconds depending on the operating altitude. From operational experience in manned aviation, these values of tau provide a balance between safety and the frequency of alerts.

TCAS has been very successful in manned aviation and played a part in reducing the amount of midair collisions to one in 100 million flight hours even though the number of flight hours flown by commercial aircraft has quadrupled since 1970 [20]. However, as the airspace becomes more populated and diversified, TCAS may not be the best solution for collision avoidance.

One issue is that TCAS only provides RAs in the vertical direction. This causes aircraft to deviate from the altitude assigned to them by ATC, by frequently more than 500 ft, which provided them with only 1000 ft of vertical separation from other aircraft in areas of high traffic densities. With such a significant reduction in separation, it is easy to see how TCAS could potentially introduce some midair collision scenarios. In contrast to vertical separation, ATC places much larger spacing laterally between aircraft, usually in terms of nautical miles. Providing lateral RAs would give aircraft more area to maneuver while

reducing the amount of induced midair collisions. The best solution to this problem will probably be a combination of vertical and lateral RAs.

In the case of the TCAS being used on UAS, one of the major concerns was that UAS could not meet the vertical acceleration and rates required by TCAS in an RA maneuver. These vertical acceleration and rates were set with the flight envelope of commercial aircraft in mind. Even though commercial aircraft remain to be the prime users of TCAS, the number of TCAS units that will be used on GA aircraft and possibly UAS could increase in the coming years. This will cause the current accelerations and rates to become increasingly irrelevant as most of the users may not be able to meet them. The current version of TCAS does provide a solution to this problem however by allowing the TCAS unit to be configured with the limitations of the aircraft. The combination of this capability and the tau concept removes the requirement that UAS must be able to meet the standard accelerations and rates to be able to use and benefit from TCAS.

The other major issue with UAS and TCAS is the communication latency. The study at MIT concluded that any latency over 15 seconds negated the use of TCAS over a Mode S transponder. With many UAS operating beyond line of sight the communications delay between the UAS operator and the UAS could become an issue if the pilot is in the loop for RA maneuvers. This issue could be solved by linking the RAs from TCAS to the Flight Management System (FMS), which would allow the UAS to perform the RA without operator intervention.

Linking TCAS or a Sense and Avoid system (SSA) to the FMS has a couple of major advantages. By taking the pilot out of the loop in avoidance maneuvers, the communication latency issue is eliminated, but the UAS also maintains the ability to avoid other aircraft in a lost link situation. In the lost link situation, one of the worst case scenarios would be for the UAS to deviate from its approved flight plan and fly a course through a highly trafficked area. Having TCAS or a SSA system linked to the FMS would decrease the risk of a collision, increasing safety while the link is reestablished with the UAS control station.

Although TCAS may not be the best solution to the collision avoidance problem, it could provide increased safety for UAS operations in the NAS in the near term. However, TCAS could only be used on UAS that are large enough to carry a transponder. Current commercially available transponders weigh 1.5 lbs and up, which could not feasibly be added to most UAS below 30 lbs without significantly affecting performance and flying qualities, so another solution to the collision avoidance problem would need to be found for UAS of this size.

TCAS has provided a significant increase in safety in the NAS by providing collision avoidance assistance against transponder equipped aircraft; however, both manned and unmanned aircraft would benefit from a Sense and Avoid system that provided collision avoidance against aircraft without a transponder.

Chapter 3 Equivalent Safe Response Requirements

The previous chapters discussed manned aviation standards for both handling qualities and performance. This chapter will explain how the applicable requirements from the instrument flight rules (IFR) and performance-based navigation (PBN) were combined to define the Equivalent Safe Response (ESR) requirements for UAS for all flight phases (departure, en-route, arrival, and approach) needed for operating in the NAS.

The ESR requirements address all the primary handling and performance characteristics needed for a UAS to safely operate in the NAS. These include requirements for maintaining commanded altitude, heading, airspeed, and bank angle, while also specifying required turn, climb, descent, and flight path following performance. These requirements are applicable during certain flight phases, which will be discussed in Section 3.5.

3.1 Altitude

The IFR and PBN both contain requirements on maintaining altitude that can be directly applied to UAS. Since the PBN requirements specify total system error (TSE), they can be used in conjunction with the IFR requirements for maintaining altitude, which is defined as a flight technical error (FTE). This difference allows the requirements for the Path Definition Error (PDE) and Navigation System Error (NSE) which can be determined by subtracting the FTE from the TSE. For example, a UAS flying level at an altitude below 5,000 ft is required

to have a vertical TSE of less than 158 ft, a FTE of less than 100 ft (from the IFR requirement), and therefore a combined PDE and NSE of less than 58 ft, which would be the required accuracy for the instrumentation being used. Table 3.1 presents the ESR requirements for maintaining altitude during level flight and climbing/descending flight in Table 3.2. It should be noted that due to the fact that ATC uses pressure altitude when controlling traffic each type of UAS (Predator, Shadow, etc.) would need to go through a pitot-static system calibration. This calibration would require analysis to determine the proper distance the pitot tube would need to be placed away from the aircraft to be able to measure pressure altitude accurately. The accuracy of the pitot-static system would have to meet the requirements for PDE and NSE.

Table 3.1: ESR requirements for maintaining altitude in level flight

Altitude	FTE	TSE	PDE + NSE
≤5,000 ft	±100 ft	±158 ft	±58 ft
>5,000 ft	±100 ft	±245 ft	±145 ft

Table 3.2: ESR requirements for maintaining altitude in climbing/descending flight

Altitude	FTE	TSE	PDE + NSE
≤5,000 ft	±100 ft	±224 ft	±124 ft
>5,000 ft ≤10,000 ft	±100 ft	±335 ft	±235 ft
>10,000 ft	±100 ft	±372 ft	±272 ft

3.2 Heading, Airspeed, and Bank Angle

The requirements presented in ESR for heading, airspeed, and bank angle all come from the IFR requirements. In terms of heading, the system is expected to maintain the commanded

heading $\pm 10^\circ$ in all phases of flight, as well as maintaining the commanded airspeed ± 10 knots or 5%, whichever is greater, throughout the entire flight. The bank angle of the UAS should never exceed 30° during the flight and the UAS should maintain the commanded bank angle $\pm 5^\circ$ during turning flight.

3.3 System Performance

There are three important maneuvers that require adequate system performance to comply with ATC instructions when operating in the NAS: turns, climbs, and descents. When instructed to change heading or performing an instrument procedure, a turn rate of $3^\circ/\text{s}$ is typically expected by ATC as discussed in Section 1.2.1. So, a UAS operating in the NAS under ATC control must be capable of performing a $3^\circ/\text{s}$ turn. Similarly, when ATC instructs an aircraft to change altitudes, ATC expects certain a vertical rate to be used. The vertical rate used in this situation must be at least 500 ft/min and generally no greater than 1,000 ft/min. Finally, the UAS must be capable of meeting all of the RNP requirements for the airspace of operation. These include the allowable lateral, vertical, and longitudinal (once defined) deviations from the desired flight path.

3.4 Constant Requirements

During a flight through the NAS, a UAS should have to constantly meet certain performance and/or handling requirements as defined in ESR to be able to safely interact with ATC. Some of the requirements stated previously are applicable to the majority of flight phases

(departure, en-route, arrival, and approach). These requirements are considered constant. These requirements are presented in Table 3.3 and are applicable in all flight phases except for when the UAS is on final approach, which will be examined in the next section.

Table 3.3: ESR constant requirements

Maintain altitude	± 100 ft
Maintain heading	$\pm 10^\circ$
Maintain airspeed	± 10 knots or 5%
Capable of vertical speeds	≥ 500 ft/min

3.5 Flight Phase Requirements

During each flight phase a UAS might be required to change altitude, course, and/or airspeed. Due to the nature of each flight phase and the typical traffic densities associated with them, the requirements placed on such maneuvers will vary depending on flight phase. In the following sections, each flight phase will be discussed using examples of the possible maneuvers that a UAS may be required to perform and the requirements for that particular flight phase.

3.5.1 Departure Requirements

The departure phase is one of the most dangerous phases of the flight because the aircraft is at low altitudes, operating near stall speed, and possibly operating in a busy airspace. No ESR requirements for the UAS deployment method were established because of the large variance of methods that are currently used. These methods can vary from a conventional wheeled takeoff to a rail launch to even a hand launch. The requirements for the actual

deployment of the UAS should be driven by the operational requirements and deployment method. However, once the UAS attains controlled flight, certain ESR requirements must be met to ensure safety.

On departure, the UAS must be able to meet the RNP for the airspace it is operating in. The RNP level would need to be verified before flight commenced, however for this phase of flight, the typical RNP level would be either 1 or 2. This corresponds to an allowable lateral TSE of $\pm(1 \text{ to } 2)$ NM and an allowable lateral FTE of ± 0.125 NM. After the UAS is in controlled flight the UAS would most likely initiate a climb to the cruise altitude cleared by ATC. During the climb out the UAS would have to meet the allowable vertical TSE and FTE previously specified in Table 3.2 as well as establish a climb rate of at least 500 ft/min. During this phase it might also be necessary to alter course to follow the departure procedure for the given airfield or to avoid other traffic. During the turn to the instructed heading the UAS would have to turn at a rate of at least $3^\circ/\text{s}$. If the UAS is instructed to fly straight and level at a given altitude and heading, the UAS must be able to meet certain requirements for maintain altitude, heading, and airspeed. The TSE and FTE requirements for altitude as stated in Table 3.1 must be met, while heading must be maintained to $\pm 10^\circ$ and airspeed to ± 10 knots or 5%, whichever is larger. After safely completing the departure phase the UAS will then be on the en-route phase.

3.5.2 En-route Requirements

The En-route phase makes up the majority of the UAS flight. In this phase, the UAS could be in transit to an objective, orbiting a prescribed location while providing surveillance, etc.

The ESR requirements stated below are the minimums that must be met to maintain safety along the UAS flight path with other air traffic, however, the operational requirements for the UAS will most likely be more stringent for the UAS to successfully be able to perform the desired mission.

While en-route, the RNP level could vary from typically 1 to 5, which corresponds to an allowable lateral TSE of $\pm(1 \text{ to } 5)$ NM and an FTE of ± 0.25 NM. Since the majority of the en-route phase will most likely be spent in straight and level cruise flight, the vertical TSE and FTE requirements would be as stated in Table 3.1. However, if the UAS is commanded to climb or descend then Table 3.2 should be used. These requirements are in addition to the constant requirements detailed in section 3.4.

3.5.3 Arrival Requirements

The UAS will be beginning the arrival phase after completing the task it was instructed to perform and has entered the airspace where the flight will be terminated. The RNP levels for this phase are typically in the same range as the en-route phase (1 to 5) and an FTE of ± 0.25 NM. The specifications for vertical TSE in Table 3.1 and Table 3.2 should be used where applicable. Also in this phase, the constant ESR requirements must be met, as well as the UAS must be able to perform a standard rate turn ($3^\circ/\text{s}$) for traffic separation purposes and performing any IFR procedures required to get in proper position for the approach phase.

3.5.4 Approach Requirements

The approach phase begins when the UAS enters the pattern for landing. The RNP levels for the approach phase are typically between 0.3 and 1 ($\pm(0.3 \text{ to } 1)$ NM) and ± 0.125 NM FTE.

From Table 3.2, below 5,000 ft the UAS must maintain a vertical TSE of less than 224 ft during the descent and 158 ft in level flight and an FTE of less than 100 ft at all times. Any turns required in the approach phase as the UAS transits the pattern should be performed at $3^\circ/\text{s}$.

These are the general requirements that could be applied to all UAS no matter what flight termination method is used. To define the requirements for final approach for a UAS performing a conventional landing the Instrument Rating Practical Test Standards were used along with the procedures for conducting an Instrument Landing System (ILS) approach. The Instrument Landing System provides precision guidance to an aircraft on approach to land on a runway. By using a combination of high intensity lighting arrays and radio signals, ILS defines a three dimensional cone for an aircraft to fly down that ends at the runway threshold. By using the Instrument Rating Practical Test Standards for conducting a precision approach using ILS, a three dimensional containment can be defined for a UAS on final approach to perform a conventional landing. To aid in visualization, a model of the containment volume was created and can be seen below in Fig. 3.1.

Figure 3.1: ESR Final approach containment volume

This volume was modeled using the specifications given in the AIM for the ILS approach. The centerline of the volume is a 3° glide slope that ends at the runway threshold at a height of approximately 74 ft above the runway. In manned aviation the top and bottom boundaries of this containment volume would be 0.75° above and below the glide slope. This causes the allowable vertical errors to decrease as the aircraft nears the runway threshold. In a similar manner, the sides of the containment volume are typically 2.5° to either side of the centerline also causing the allowable lateral errors to decrease as the distance to the runway threshold decreases. However, the Instrument Rating Practical Test Standards requires that no more than a $\frac{3}{4}$ deflection of the CDI be allowed while on final approach. A full deflection of the CDI would indicate that the aircraft was at the outer limits of the ILS band, so to define the

containment volume for UAS on final approach the degrees that the sides and top and bottom limits were multiplied by $\frac{3}{4}$, which gives 0.525° above and below the glide slope and 1.875° to either side of the centerline.

To define the size of the containment area at the threshold of the runway it was assumed that the cone originated on the centerline at a distance of approximately 8000 ft. This was determined from the specifications given in the AIM regarding the width of the ILS band at the runway threshold, which was specified as typically being 700 ft. If full scale deflections to either side of the centerline were used, 2.5° , the distance between the origin of the ILS band and the runway threshold could be determined. This distance was then used to determine the width of the containment volume at the runway threshold. Using the $\frac{3}{4}$ deflections requirement, 1.875° , a width of 524 ft was determined. Similarly for the height, the distance and $\frac{3}{4}$ deflections were used to determine that the containment volume height at the runway threshold was 147 ft. If the glide slope is placed at a height above the runway threshold of approximately 74 ft, then the lowest boundary of the containment volume terminates at the surface of the runway threshold (See Fig. 3.2 and 3.3).

Figure 3.2: Side view of final approach containment volume

Figure 3.3: Top view of final approach containment volume

As distance from the runway threshold increases, the allowable lateral and vertical deviations become larger. These were not allowed to become larger than those specified for the approach phase (Maintain altitude ± 100 ft FTE and lateral deviation ± 0.125 NM FTE) and are held constant until the distance to the runway threshold has decreased enough that these allowable deviations become smaller.

3.6 Environmental Effects and ESR

The environment can be one of the greatest threats in aviation. Aircraft flying in the NAS deal with such environmental effects as wind, turbulence, precipitation, etc every day. In manned aviation, aircraft are expected to be able to meet the requirements discussed in Chapter 1 regardless of the weather conditions. The PBN manual describes the expectations of ATC of an aircraft flying in weather that causes some requirements not to be met: “If aircraft cannot maintain navigation accuracy notify ATC and coordinate a plan of action and obtain a revised clearance.” This is the way that manned aviation handles environmental effects, and this approach can be applied to unmanned aviation as well. The requirements presented in ESR are independent of environmental effects and if they cannot be met due to weather, ATC should be advised immediately and a new clearance obtained. A new clearance could include a UAS being routed around a weather system or a block of altitude being cleared of other traffic for the UAS.

Chapter 4 Equivalent Safe Response Model

Once the requirements for ESR were determined, a method of evaluating a UAS was developed. The simulation that is detailed in the following sections allows the evaluation of the closed outer-loop handling qualities of a UAS by requiring a model of the UAS to negotiate an obstacle course. After the UAS has completed the course, the performance of the UAS is analyzed using a MATLAB code that was developed to verify that the ESR requirements have been met.

4.1 The ESR Course

The software that was chosen to develop the ESR course was the Piccolo 2.1.1 by Cloud Cap. This software was chosen because it is one of the most widely used and flexible autopilot technologies available. The software-in-the-loop (SIL) feature of the Piccolo allows users to input a model of the UAS, set the necessary gains and parameters in the Piccolo, and run the ESR course. A Piccolo simulation model is an input text file that contains an aerodynamics, propulsion, sensor, actuator, launcher, inertia, and landing gear model of the UAS. A step by step guide to creating a Piccolo simulator model is available in the Piccolo documentation [21]. The SIL feature also includes the option of adding environmental effects to the simulation, which can be used to further evaluate the handling qualities and performance of the UAS.

The environmental effects that are available in the Piccolo software are wind, turbulence, and thermals. Further research discovered that the turbulence model implemented in the Piccolo software was the Dryden turbulence filters, which is used for body velocities and angular rates. The Dryden Wind Turbulence Model (Discrete) uses the Dryden spectral representation to add turbulence to an aerospace model by using filtered band limited white noise [22]. This turbulence model is the mathematical representation in the Military Specification MIL-F-8785C and Military Handbook MIL-HDBK-1797. The use of environmental effects in the ESR course allows users to gain more confidence in the types of weather conditions the UAS could operate in without causing increased risk to other air traffic in the NAS.

The ESR course consists of the four flight phases discussed earlier, departure, en-route, arrival, and approach. This makes it possible to test the UAS in the appropriate conditions it has been designed for. This tests for instance any performance differences between cruise altitude and approach altitude, or UAS that change configuration between flight phases. The course has initially been designed for small UAS and consists of 52 waypoints; altitude varies from approximately 75 ft to 3,000 ft, and is approximately 22 NM in length.

However, the altitude and length of the course can be adjusted to accommodate all sizes of UAS. The entire ESR course as viewed in the Piccolo software is shown in Fig. 4.1, along with the altitude profile in Fig. 4.2.

Figure 4.1: The ESR course as viewed in the Piccolo software

Figure 4.2: The ESR course altitude profile

In the ESR course, the UAS will execute six types of tests: climb rate, descent rate, turn rate, straight and level, approach test, and RNP test. Each type of test has prescribed criteria and the UAS receives a pass/fail grade based on the performance of the UAS during that test.

The analysis and grading is performed by the ESR code written in MATLAB (See Appendix A).

4.1.1 Climb Rate and Descent Rate Tests

The climb rate and descent rate tests consisted of an instantaneous change in commanded altitude of at least 500 ft and required that the UAS attain an average vertical rate of at least 500 ft/min. The test began once the change in altitude was commanded and ended at the next waypoint or when the UAS was within 40 ft of the commanded altitude. This was due to the decrease in vertical rate by the autopilot when nearing the commanded altitude to decrease overshoot.

4.1.2 Turn Rate Test

The turn rate test is a set of five waypoints that required the UAS to perform a set of “S” turns. This allowed the collection of multiple data sets for turns in both directions. To receive a passing grade for the turn rate test, the UAS had to turn at an average rate of greater than or equal to $3^{\circ}/s$.

4.1.3 Straight and Level Test

Each flight phase contains a straight and level test. In this test the UAS is given a commanded altitude, course, and airspeed and is required to maintain altitude ± 100 ft, course $\pm 10^{\circ}$, and airspeed ± 10 knots.

4.1.4 Approach Test

The approach test is executed in the approach phase on final approach to the landing point.

During this test the UAS is commanded to fly a 3° glide slope from a pattern altitude of 500 ft to a height of 75 ft above the runway. Glide slope and lateral deviations are limited to $\pm 0.525^\circ$ and $\pm 1.875^\circ$ respectively, as discussed in Section 3.5.4. In addition, the UAS is required to maintain the commanded airspeed ± 10 knots. The ESR course also includes a missed approach test. This test uses the same containment volume as described in Section 3.5.4, except that the UAS is initially at the runway threshold and must remain inside the containment volume until reaching the pattern altitude of 500 ft.

4.1.5 RNP Test

While the UAS is performing any of the other five types of tests, the system is also being evaluated on the lateral deviations from the commanded course. One grade is given for the RNP test for each flight phase, and in each flight phase the UAS must meet the allowable lateral FTE for that phase as specified in Table 4.1. In the ESR simulation the PDE and NSE are assumed to be negligible, so the allowable deviations are reduced from the TSE values to the FTE values.

Table 4.1: ESR simulation lateral deviation requirements

Flight Phase	FTE (NM)
Departure	0.125
En-route	0.25
Arrival	0.25
Approach	0.125

4.1.6 Departure Phase

In the departure phase of the ESR course, the UAS must perform four tests: a climb rate test, a descent rate test, a straight and level test, and finally a turn rate test. Since ESR does not include requirements for the UAS launch procedure, the ESR course begins at waypoint 0, which is located beyond the end of the runway at an altitude of 500 ft. This allows the flexibility to accurately simulate the deployment method for the UAS without altering the course. The departure phase portion of the course is shown in Fig. 4.3 below. As stated before, the departure phase begins at waypoint 0 and ends at waypoint 11.

Figure 4.3: Departure phase of the ESR course

The first test the UAS will encounter in the departure phase is the climb rate test which is performed between waypoint 0 and 1. Once waypoint 0 has been reached the UAS receives an instantaneous increase in commanded altitude of 1,000 ft. The UAS will climb to reduce the error in commanded altitude. Once the UAS has climbed to within 40 ft of 1,500 ft or has reached waypoint 1 the climb rate test is completed. If the UAS has not reached 1,500 ft by waypoint 1, then the UAS proceeds to waypoint 2 where it enters into an orbit. The UAS will remain in the orbit until it has climbed to 1,500 ft. Each climb rate and descent rate test is followed by an orbit so that the UAS will be in proper position to begin the next test. This is to prevent a failure in a climb or decent rate test to affect the UAS performance in following tests. Once reaching 1,500 ft the UAS proceeds to waypoint 3 where it will begin the next test.

The next test that is executed is the descent rate test. Once the UAS reaches waypoint 3 it receives an instantaneous decrease in commanded altitude of 500 ft. Like in the climb rate test, the test is finished when the UAS reaches waypoint 4 or within 40 ft of 1,000 ft.

Similarly, if the UAS is unable to descend to 1,000 ft before reaching waypoint 4, the UAS enters another orbit until the commanded altitude has been reached. At waypoint 6 the UAS performs the straight and level test at 1,000 ft to waypoint 7. This waypoint also marks the beginning of the turn rate tests, which is a series of four turns, two in each direction.

Waypoints 8 – 11 were positioned to force a small UAS (~30 lbs) with an airspeed of approximately 30 knots to be forced to perform steep turns to stay on course. If evaluating a

larger and faster UAS it could require the distance between these waypoints to be increased so that the UAS could maintain course. Once waypoint 11 is reached, the departure phase has ended and the UAS is then proceeding into the en-route phase.

4.1.7 En-route Phase

The En-route phase of the ESR course consists of a climb rate, descent rate, and straight and level test. A turn rate test is also included in this phase; however, it is not graded since there is no turn rate requirement for the en-route phase. The turn rate that the UAS is capable of in the en-route phase is still important and should be recorded for ATC reference. The en-route phase as viewed from the Piccolo Command Center is shown in Fig. 4.4 below.

Figure 4.4: En-route phase of the ESR course

The en-route phase of the ESR course begins at waypoint 11 and ends at waypoint 22. In a similar manner as the departure phase the UAS will first be required to perform a climb rate test. This involves an increase in commanded altitude of 2,000 ft. An altitude of 3,000 ft was chosen because it was representative of the operating altitudes for small UAS. The climb rate test is completed once the UAS has reached waypoint 12 or is within 40 ft of 3,000 ft. If the UAS has not reached 3,000 ft it enters an orbit until it has. After reaching waypoint 14, the en-route descent rate test is performed. The UAS is commanded to descend 500 ft to an altitude of 2,500 ft. Once the descent rate test has been completed and the UAS is level at 2,500 ft, it proceeds to waypoint 17. This waypoint is the beginning of the straight and level portion of the en-route phase which ends at waypoint 18. Waypoints 18 through 22 are the en-route turn rate test, which are not graded and only for reference as previously mentioned. Once the UAS reaches waypoint 22, the en-route phase of the simulation is complete and the next phase begins.

4.1.8 Arrival Phase

The arrival phase in the ESR course consists of waypoints 22 through 33, and is similar to the en-route phase as seen in Fig. 4.5 below. This phase directs the UAS back to the airfield and ends with the UAS entering the approach pattern for landing. In the arrival phase the UAS performs four major tests: climb rate, descent rate, turn rate, and straight and level.

Figure 4.5: Arrival phase of the ESR course

At the beginning of the arrival phase the UAS is commanded to climb 500 ft to an altitude of 3,000 ft for the climb rate test and next to descend 2,000 ft for the descent rate test. This large descent is required to have the UAS at the pattern altitude to begin the approach phase. As in the previous flight phases, the final test is the turn rate test, which for this phase is graded. After completing the turn rate test in the arrival phase, the UAS enters the pattern around the airfield and begins the approach phase.

4.1.9 Approach Phase

The approach phase in the ESR course contains the most tests to be performed of all the flight phases. The approach phase of a real flight in the NAS can be one of the most dangerous and risky portions of the flight because of operations at lower altitudes in closer proximity to other air traffic. This is why it is necessary to have more tests during this phase in the ESR course. The approach phase accounts for the remaining waypoints, 33 through 51, which can be seen in Fig. 4.6 below.

Figure 4.6: Approach phase of the ESR course

The type of tests that are performed in this phase are descent rate, turn rate, approach, missed approach, and straight and level. Once the UAS has started the approach phase a left handed

pattern is used. The first test to be performed is the descent rate test. In this phase, the UAS is commanded to descend 500 ft from an initial altitude of 1,000 ft. An altitude of 500 ft is used for the remaining portions of the flight phase except for the approach and missed approach tests. Each turn in this phase is used to test the turn rate of the UAS. These include the turns at waypoints 37, 38, 44, 45, 48, and 49. Waypoint 39 marks the beginning of the first approach test. During the approach test the UAS is commanded to follow a 3° glide slope down from an altitude of 500 ft to 75 ft. Once at waypoint 40, a missed approach is simulated by commanding the UAS to climb back up to 500 ft. During the missed approach test, the UAS follows a given climb out angle similar to the final approach test. Once the missed approach test is completed by reaching waypoint 41, the pattern is reversed and the UAS enters a right handed pattern. This is simply to test the UAS performance in both the left hand and right hand patterns. Once waypoint 46 is reached, the UAS performs a straight and level test at 500 ft. After performing the two remaining turn rate tests, the final approach test is performed. The final approach test is a duplicate of the previous final approach test performed and once the UAS reaches waypoint 51 at an altitude of 75 ft the UAS has completed the ESR course.

Once the ESR course has been completed, the Piccolo Command Center can be closed. To analyze and grade the performance of the UAS the log file that is output by the Piccolo Command Center is used as an input file to the ESR MATLAB code, which is discussed further in the following section.

4.2 ESR MATLAB Code

The primary component of the ESR model is the ESR MATLAB code. This code was written to read the log file that is output from the Piccolo Command Center after a UAS has flown the ESR course and to output a graded ESR report. This ESR report is a text file that contains the overall ESR grade and also the results from each flight phase and each individual test. The ESR code and supporting functions have been included in Appendix A.

Firstly, the ESR code was written primarily to read the Piccolo log file, however it was written so that the format of the input file can easily be changed in the code to accept files from different sources. Only simulations using the ESR course can be analyzed using the code since the waypoint latitudes and longitudes from the ESR course are used to determine the required ground track from waypoint to waypoint. This, however, could easily be altered to accommodate different courses.

The ESR code allows the input file to be located anywhere that can be accessed by the computer running the code. Once the code is started, all previously open MATLAB figures will be closed, the workspace and command window will be cleared, and a directory window will open allowing the user to select the input file. The Piccolo log file has a unique format which contains over 120 columns of either numbers or characters, which are delimited by spaces. The ESR code was written to extract the necessary data from the log file for the evaluation, which consists of 19 columns of the log file. The columns and their original units that are extracted include:

- Clock (milliseconds)
- Year
- Month
- Day
- Latitude (radians)
- Longitude (radians)
- Course (radians)
- Altitude (m)
- True Airspeed (TAS) (m/s)
- Outside Air Temperature (OAT) (°C)
- Static Pressure (Pa)
- Dynamic Pressure (Pa)
- Bank Angle (radians)
- Target Indicated Airspeed (IAS) (m/s)
- Target Waypoint
- Target Altitude (m)
- Target Bank Angle (radians)
- Lateral Track (m)
- Distance to Waypoint (m)

All of the units are converted to English units (feet, degrees, and seconds), with the exception of outside air temperature (OAT), and static and dynamic pressure, which will be explained further later in this section. Due to the way that the Piccolo software writes the log file, some of the extracted columns can contain an additional entry than other columns. This becomes a problem because certain operations with arrays in MATLAB require the arrays to be the same size. The issue was solved by determining the size of the largest array, which was always the clock array because it was the first to be written, and then deleting the last entry of the clock array and any array with the same number of entries as the clock array. Once all of the arrays were the same size the latitude, longitude, and altitude of each waypoint from the ESR course were read.

In the ESR code each leg of the ESR course, or portions between the waypoints, were each referenced by the three letter flight phase abbreviation (dep, enr, arv, app) and the leg number, where each leg of the flight phase was numbered sequentially. For example, the variable name for any variable pertaining to the 3rd leg of the en-route phase began with the label enr3.

To be able to analyze each leg of the course we first needed to determine the flight time and row of the arrays that each leg began. This was accomplished using the Waypoint array, which contains the waypoint number that was being tracked at each time step in the log file. The time step used in the Piccolo SIL simulation varies from 0.001 – 0.1 seconds. A simple for loop and embedded if statements assigned the first occurrence of a waypoint as the row position for the start of that leg. This was then used to declare the flight time that each leg started. The end of a leg was simply determined by subtracting one from the starting row position of the following leg.

Next, a for loop was used to determine the target course between each waypoint, which is used later to determine the deviations in course by the UAS. The equation used in MATLAB to calculate the course between two points, A and B, is shown below [23].

$$tc = mod(atan2(\sin(lonB - lonA) \cos(latB), \cos(latA) \sin(latB) - \sin(latA) \cos(latB) \cos(lonB - lonA)), 2\pi) \quad (1)$$

Where

tc = course from point A to point B

$lonA$ = longitude of point A

$latA$ = latitude of point A

$lonB$ = longitude of point B

$latB$ = latitude of point B

An array containing the target course for each leg of the ESR course was created once each sequential pair of waypoints was evaluated with the above equation.

The Piccolo log file records the UAS's true airspeed (TAS), but records the commanded airspeed as indicated airspeed (IAS). This presented a problem because IAS is only equivalent to TAS if the air density at the aircraft's location is equal to the standard sea level value of air density. To be able to measure the deviation between the UAS airspeed and the commanded airspeed, a function was written called TASTarget.m (See Appendix A) to convert the commanded airspeed to TAS using the following equation [24].

$$V_{true} = V_{indicated} \sqrt{\frac{\rho_s}{\rho}} \quad (2)$$

Where

ρ_s = standard sea level density of air

ρ = density of air at the location of the aircraft

The air density at the location of the aircraft was able to be determined from the recorded static and dynamic pressures and outside air temperature (OAT) that was extracted from the log file with the equation shown below. However, before determining the air density, the OAT was converted to Kelvin (K) to have consistent units.

$$\rho = \frac{p_0 + p}{R(OAT)} \quad (3)$$

Where

p_0 = static pressure

p = dynamic pressure

R = specific gas constant

OAT = outside air temperature

Once all the data had been organized and converted to proper units, the flight data for each leg in which a test was performed in the ESR simulation was analyzed. A function called `flight_phase_analysis.m` (See Appendix A) was written to perform this task. This function was used to determine a majority of the necessary values needed for evaluating the UAS performance in the ESR course.

This function was given the UAS's actual performance (altitude, course, TAS, etc.) along with the commanded values and the beginning and ending row positions of the arrays for the flight leg being analyzed. The function then returned the data listed below.

- Lateral deviation from the commanded flight path in NM

- Deviations from commanded altitude, course, and airspeed over time
- Maximum deviations from commanded altitude, course, and airspeed
- Vertical rate and turn rate over time
- Average vertical rate and turn rate

These values were determined for each flight leg for which a test was performed even though not all of the values were used in the evaluation of the test. The values not used to evaluate the performance of the UAS in a particular test were simply ignored. This allowed the use of one function to calculate all of the necessary values which allowed any revisions that were needed to be implemented more quickly. The calculation of unnecessary values did not negatively affect the performance of the MATLAB code appreciably. The code was able to run in less than a minute.

In the Flight Phase Analysis function, the deviations in altitude and airspeed were simply calculated by taking the difference between the Altitude and Target Altitude array, and the TAS and Target TAS array, respectively. The maximum deviations in these arrays were then found using the max function. The absolute value of each array was first found and then the max function returned the maximum deviation in commanded altitude and airspeed.

The course deviation, θ was calculated by using the following equation.

$$\theta = \text{acos}(v1 \cdot v2) \quad (4)$$

Where

$v1$ = unit vector in the direction of the commanded course

$v2$ = unit vector in the direction of the UAS course

The dot product of the two vectors was evaluated with the equation below.

$$v1 \cdot v2 = \sin(v1) \sin(v2) + \cos(v1) \cos(v2) \quad (5)$$

In a similar manner to the altitude and airspeed deviations mentioned before, these formulas were used to create an array of the course deviation from the commanded value at each time step. The maximum was then found by using the max function as before.

Next, the turn rate was calculated using equations 4 and 5 where

$v1$ = unit vector in the direction of the UAS course

$v2$ = unit vector in the direction of the UAS course 10 time steps ahead of $v1$

Equation 4 was then divided by the time difference in seconds between $v1$ and $v2$ to determine the turn rate at that time step. This calculation was repeated to create an array consisting of the turn rate of the UAS at each time step. The calculation was ended when the UAS was within 10° of the commanded course. Once the UAS was within 10° of the commanded course the UAS would begin to roll out of the turn. Since the ESR requirement pertains to the sustainable turn rate this portion was excluded from the calculation. To determine the average turn rate, the mean function was used with the turn rate array. If the leg of the flight being analyzed did not include a turn and the UAS had maintained the

commanded course, this calculation was skipped since the deviation between actual and commanded course was less than 10° .

To determine the average vertical rate of the UAS during a flight leg, the difference was taken between the altitude at each time step and the altitude 20 time steps ahead, which was then divided by the difference in time in minutes. As with the turn rate calculation, an array was defined by the vertical rate at each time step and then the mean function was used to determine the average climb rate during the climb. The calculation was ended once the end of the leg was reached or if the altitude of the UAS was within 40 ft of the target altitude. Including the climb rates during the leveling off of the climb would have lowered the average and not accurately portrayed the attainable climb rate of the UAS.

Finally, the maximum lateral deviation from the commanded course was determined. The log file from the Piccolo Command Center included this lateral deviation at each time step, as stated previously. Once an entire flight phase had been analyzed using the Flight Phase Analysis function, the lateral deviations array contained the lateral deviation at each time step throughout each test in the flight phase. Since the RNP requirements were defined on a three standard deviation basis in the Performance Based Navigation Manual by ICAO, three times the standard deviation of the array of lateral deviations was used to define the lateral deviation value for each flight phase, which was eventually used to grade the RNP performance of the UAS.

Each leg of each of the flight phases were analyzed in the same manner with the Flight Phase Analysis function. Table 4.2 includes each flight leg and the test that was performed during the leg.

Table 4.2: Flight phase legs and ESR tests

Flight Phase Leg	ESR Test
Departure 2	Climb rate
Departure 5	Descent rate
Departure 8	Straight and level
Departure 9	Turn rate
Departure 10	Turn rate
Departure 11	Turn rate
Departure 12	Turn rate
En-route 1	Climb rate
En-route 4	Descent rate
En-route 7	Straight and level
En-route 8	Turn rate
En-route 9	Turn rate
En-route 10	Turn rate
En-route 11	Turn rate
Arrival 1	Climb rate
Arrival 4	Descent rate
Arrival 7	Straight and level
Arrival 8	Turn rate
Arrival 9	Turn rate
Arrival 10	Turn rate
Arrival 11	Turn rate
Approach 1	Descent rate
Approach 5	Turn rate
Approach 6	Turn rate
Approach 7	Approach
Approach 8	Missed Approach
Approach 12	Turn rate
Approach 13	Turn rate
Approach 14	Straight and level
Approach 16	Turn rate
Approach 17	Turn rate
Approach 18	Approach

Once all the analysis of the UAS performance in each flight phase was complete, the calculated performance data was evaluated to determine whether the ESR requirements had been met. To accomplish this, several functions were written. These functions included: ESRclimb, ESRturn, ESRstraightlevel, ESRfinalapproach, and ESRmisapproach. In the case of a climb or descent rate test the ESRclimb function was used. This function was given the average vertical rate calculated during the test and returned a character string of “pass” or “fail” for both climb and descent rate. If the absolute value of the average vertical rate was greater than or equal to 500 ft/min, then a grade of pass was given for both climb rate and descent rate. However, once the grades were returned to the main ESR code, the grade that was not applicable was simply ignored (i.e. if the test was a climb rate test the descent rate test grade was ignored). In a similar manner, the ESRturn function was given the average turn rate during a turn rate test and returned a character string of “pass” or “fail”. If the average turn rate was less than 3°/s than the function returned a grade of Fail.

The remaining functions, ESRstraightlevel, ESRfinalapproach, and ESRmisapproach were slightly more complex than the previous functions by requiring multiple parameters to be met. The ESRstraightlevel function was given the maximum altitude, heading, and airspeed deviations for the straight and level test in a flight phase, and returned a grade for each requirement along with an overall grade for the test. If the altitude deviation was greater than 100 ft, than that portion of the test was failed. Similarly, the heading deviation was required to be less than 10° and the airspeed deviation less than 10 knots or 16.9 ft/s to receive a

passing grade for either. The overall grade for the straight and level test depended on the individual requirements. If all three requirements had passed then a grade of pass was given for the entire test, however, if only one requirement was not met, the test grade would be Fail.

The ESRfinalapproach function was used to grade each approach test. To grade the UAS's performance during this maneuver the function was given altitude deviations over the time period of the test, maximum airspeed deviation, lateral deviations over the time of the test, and the distance to the next waypoint over time. In the approach test the requirements for lateral and vertical deviations from flight path are dependent on the distance to the next waypoint as discussed in Section 3.5.4. This waypoint ends the test and represents the end of the glide slope at a point 75 ft above the runway threshold. To determine the allowable altitude deviation along the flight path the following equation was used.

$$\text{Allowable Alt Dev} = 73.5 + \text{Dist to WayPt}(\tan(0.525^\circ)) \quad (6)$$

If the value calculated from equation 6 was larger than 100 ft, then the allowable altitude deviation was declared to be 100 ft. This was done to prevent the allowable altitude deviations from increasing once the UAS was on final approach. If the UAS was within 2,846 ft of the runway threshold then the allowable altitude linearly decreased until the crossing the runway threshold. At the threshold the allowable altitude deviation was ± 73.5 ft. If the UAS altitude deviation was ever more than the allowable deviation then a grade of

fail was given. Similarly, to determine the allowable lateral deviation the equation below was used.

$$\text{Allowable Lat Dev} = 262 + \text{Dist to WayPt}(\tan(1.875^\circ)) \quad (7)$$

If the lateral deviation of the UAS during the final approach was found to be greater than the allowable deviation calculated in equation 7, then a grade of fail was given.

Lastly, if the maximum airspeed deviation during the final approach was greater than 10 knots (16.9 ft/s), then the grade of fail was also given. Once the altitude, lateral, and airspeed deviations had been graded the overall grade was determined. If all three parameters had been met, then the UAS was given a grade of pass for the final approach test.

The ESRmisapproach function was very similar to the ESRfinalapproach function, so it will not be discussed in as much detail. The differences between the functions were the equations used to determine the allowable altitude and lateral deviations. This was because the allowable lateral deviations increase as the UAS climbs away from the runway on a missed approach. The equations that were used to determine the allowable altitude and lateral deviations are given below, respectively.

$$\text{Allowable Alt Dev} = 73.5 + \text{Dist from WayPt}(\tan(0.525^\circ)) \quad (8)$$

$$\text{Allowable Lat Dev} = 262 + \text{Dist from WayPt}(\tan(1.875^\circ)) \quad (9)$$

Again, the maximum airspeed deviation was also required to be less than 10 knots (16.9 ft/s).

The last test to be graded in each flight phase was the RNP test. As stated earlier, the PBN manual states that an aircraft must meet the lateral deviation specifications for a particular airspace on a 3σ basis or 95% containment. As discussed previously, this value was found for each flight phase and used to grade the RNP test with the requirements specified in Table 4.1. Once all of the ESR tests in each flight phase had been graded, the flight phases and overall ESR grade were determined. In a similar grading manner as discussed previously, if any ESR test in a flight phase has been given a grade of fail, then the flight phase was given the same grade. This was the same for the overall ESR grade, if a flight phase was failed, then the overall grade was Fail. In other words, if any requirement was not met the UAS failed the ESR evaluation.

Once all the grades are assigned the ESR code is able to create an output file with a summary of the ESR evaluation and also able to create figures to illustrate the UAS's performance in each test. Menus were created to give the user the option of saving the results, plotting the results, and which flight phase to plot. If the user selects to save the ESR results, a directory menu is opened to allow the user to select the location they would like to save the results to and to create the file name. The user will then be prompted in the MATLAB command window for the name of the UAS being evaluated and the weather conditions during the test. These are included in the ESR results file along with the date the actual test was performed, which is determined from the input file to the ESR code. Once the results file has been created a menu will be opened giving the user the option to plot the results. If the user

selects to plot then a secondary menu is opened and request the user to select the flight phase to plot. These plots have been predefined to illustrate the main requirements for each test and will be automatically saved if the user elects to save the results.

If the user opens the ESR results file, which is typically saved as a text file, the header of the file will contain the name of the UAS that was evaluated, the date of the simulation, and the weather conditions that were simulated. Following this will be the overall ESR grade, followed by the individual flight phase grades. Below this is a detailed breakdown of each test with the actual performance of the UAS versus the ESR requirement. The next chapter details the evaluation of a small UAS using the ESR model and demonstrates further how the ESR model is used.

Chapter 5 ESR evaluation of the ArcWulf

The ArcWulf is a small UAS that is currently being operated by the Aerial Robotics Club at North Carolina State University, which won the 2010 AUVSI Student Competition. The ArcWulf was chosen as a test case for the ESR model for multiple reasons. The ArcWulf is a 24 lb UAS with an 8 ft wingspan and a cruise velocity of 30 knots, which is a relevant data point for the small UAS currently being operated by the military. The ArcWulf was also equipped and operated with the Piccolo autopilot system, so a Piccolo model and tailored gains and parameters were available.

The ArcWulf was evaluated in three different environmental conditions; calm, light turbulence, and a steady 14 knot (24 ft/s) wind from the East. In each condition, after the ArcWulf model had navigated the ESR course in the Piccolo SIL simulation, the log file was used as the input to the ESR MATLAB code. The code had typically finished the evaluation in less than 11 seconds and the ESR results file was ready to be reviewed and plots generated. The computer running the MATLAB code was equipped with an Intel Core 2 Duo 2.26GHz processor and 4 GB of RAM. In the simulation with calm conditions the ArcWulf passed the ESR evaluation by a large margin. The ESR results file and plots for each condition have been included in Appendix B. As would be expected, the ArcWulf

deviations were larger when environmental effects were simulated. However, in all three cases the ArcWulf successfully passed the ESR evaluation.

The ArcWulf was the closest of any of the three conditions to failing an ESR requirement in the simulation with light turbulence. The turbulence had the greatest effect on the ArcWulf's ability to hold a commanded course to $\pm 10^\circ$. This can be seen in Fig. 5.1, which plots the actual course of the ArcWulf and the commanded course during the straight and level test in the approach phase. The plot on the left is in calm conditions, while the plot on the right is with light turbulence. In calm conditions the ArcWulf's maximum course deviation was 2° . When light turbulence was included in the simulation the maximum deviation increased to 7.3° , almost four times larger than in calm conditions.

Figure 5.1: ArcWulf course tracking without (left) and with (right) turbulence

By meeting all of the ESR requirements confidence was gained that the ArcWulf UAS could safely operate in the NAS. This evaluation also served to test the operation of the ESR

course and code. Some needed modifications to the code and course were highlighted while conducting these evaluations of the ArcWulf, but the current version has been given in this document.

5.1 Modification of the ArcWulf model to reduce flight path deviations

After the initial ESR evaluations of the ArcWulf were complete, it became clear that the more rigorous ESR requirement to meet during turbulent conditions was maintaining course to $\pm 10^\circ$. The ArcWulf model was modified in an attempt to reduce the course deviations in straight and level flight in light turbulence conditions. The modification to the ArcWulf model was in two forms: weight and tuning the gains of the Piccolo autopilot.

In aerospace projects it is almost always advantageous to reduce the weight rather than add to it. However, the ArcWulf has a relatively light wing loading for its size and by increasing the weight of the ArcWulf and therefore the moment of inertias the accelerations due to turbulence could be reduced. This would potentially increase the ability of the ArcWulf to remain on course during turbulent conditions.

During the Piccolo SIL simulations, it was noted that the ArcWulf was most affected by turbulence in roll. The oscillations in roll were therefore the most likely cause of the course deviations, so the roll inertia of the ArcWulf model was increased by adding weight to either wing tip. Adding weight at either wing tip would not only increase the roll inertia of the aircraft but also the yaw inertia, which would further increase the ability of the ArcWulf to

reject disturbances while in straight and level flight. The total amount of weight that was added was approximately 0.2 lbs, which was less than a 1% increase in the weight of the ArcWulf. Assuming that the added mass was located on the y-axis of the body axes of the aircraft and that the x and z-axis intersected at the midpoint between the two masses, the new moment of inertias of the ArcWulf model were calculated using the parallel axis theorem and the following equations. The original file that contained the ArcWulf model was written in metric units, so those units were carried through the following calculations. Equation 10 was used to calculate the moment of inertias of the additional masses about the respective axes.

$$I_{ii'} = \frac{m^2}{2m} d^2 \quad (10)$$

Where

$I_{ii'}$ = moment of inertia of the additional mass about the indicated axis

m = mass at each wing tip

d = wing span

Using equation 10, where m was equal to 0.046 kg and d was equal to 2.36 m, the following inertias were calculated.

$$I_{xx'} = 0.1281 \text{ kg m}^2$$

$$I_{yy'} = 0 \text{ kg m}^2$$

$$I_{zz'} = 0.1281 \text{ kg m}^2$$

These moments of inertia were then added to the original corresponding inertias in the ArcWulf model. Table 5.1 below displays the changes in the total weight and moments of inertia of the ArcWulf.

Table 5.1: Modified ArcWulf Weight and Inertias

	Weight (kg)	I_{xx} (kg m ²)	I_{yy} (kg m ²)	I_{zz} (kg m ²)
Original	10.658	0.5389	1.2662	1.8421
Modified	10.750	0.6670	1.2662	1.9702
Percent Change	0.9%	23.8%	0.0%	7.0%

Once the ArcWulf model had been updated with the new weight and inertias it was evaluated using the ESR model in light turbulence conditions, which the complete results can be found in Appendix B. Since the main goal was to decrease deviations during the straight and level test, the average deviations during all of the straight and level tests of the ESR course were compared with and without the additional weight. These values are presented in Table 5.2 below, where a green percentage change indicates an improvement and red indicates a decrease in performance. Figure 5.2 also illustrates the change in course deviation performance by comparing the performance of the ArcWulf during the straight and level test in the Approach phase with and without the added weight.

Table 5.2: Average Straight and Level Performance Comparison with Additional Weight

	Original	Modified	Percent Change
Altitude Deviation (ft)	2	3	50%
Course Deviation (°)	5.4	1.3	76%
Airspeed Deviation(ft/s)	3.5	3.8	9%

Figure 5.2: ArcWulf course tracking without (left) and with (right) added weight

As seen in Table 5.2, increasing the weight and moments of inertia of the ArcWulf provided a significant decrease in average course deviation during straight and level flight in light turbulence conditions. However, the ArcWulf's ability to maintain altitude and airspeed suffered slightly. Even though the average course deviations were decreased it was desired to increase the overall performance of the ArcWulf in straight and level flight. To accomplish this a gain tuning process was conducted with the Piccolo autopilot and the modified ArcWulf model. The Piccolo software includes documentation that describes how

to tune the autopilot gains to obtain the desired aircraft model performance and response [25]. This process could differ between autopilot manufacturers since varying control laws maybe be used. However, in general, the gains of an autopilot will be tunable. The changes that were made to the Piccolo gains are listed below.

Table 5.3: Piccolo Lateral Gains Before and After Tuning

	Original	After Tuning
Roll rate error integral to aileron	5.0	1.0
Tracker convergence	0.3	0.5
Heading error derivative to turn rate	0.1	0.0

The roll rate error integral to aileron and heading error derivative to turn rate gains were decreased to decrease the fast bank oscillations that were being observed in the simulation as recommended by the Piccolo control laws tuning documentation. Also increasing the tracker convergence allowed the ArcWulf to more slowly converge on a track segment, which also decreased the fast bank oscillations. The comparison of the average straight and level performance of the original ArcWulf model and the model with the added weight and tuned gains is presented in Table 5.4 below.

Table 5.4: Straight and Level Performance Comparison with Additional Weight and Tuning

	Original	Added Weight and Tuning	Percent Change
Altitude Deviation (ft)	2	2	0%
Course Deviation (°)	5.4	2.6	52%
Airspeed Deviation(ft/s)	3.5	2.9	17%

With the combination of the increased moments of inertia and the tuning of the Piccolo gains the average straight and level flight path following performance of the ArcWulf was increased. The modifications to the ArcWulf did have a negative effect on the turn rate of the aircraft, however, and therefore caused much higher lateral deviations in the turn rate tests. This can be seen in the RNP level test in the full comparison between the original and modified ArcWulf in Appendix B.

Chapter 6 Conclusion

As UAS are introduced into the NAS, specific outer-loop handling qualities and performance requirements are needed to maintain a safe environment. Current sources on the subject have yet to determine appropriate concrete requirements for this situation. ESR attempts to aid in this area by defining necessary and applicable requirements for UAS and by developing a method to evaluate a given UAS's performance and outer-loop handling qualities. The ESR requirements were primarily derived from the FAA Instrument Rating Practical Test Standards, which are used to license pilots in manned aviation for instrument flight. By meeting these requirements and others from sources such as PBN, the FARs, and the AIM, the interaction between UAS and ATC should be similar to that of manned aircraft and should reduce any additional workload that ATC may incur.

ESR also provides a way of evaluating a UAS by using the Piccolo SIL simulations, a specific obstacle course designed to stress each requirement, and a MATLAB program that was written to analyze and evaluate the performance of the UAS. Other autopilot/simulation software besides the Piccolo could be used to obtain the data needed to use the MATLAB program however. By using the ESR model, a user can evaluate the outer-loop handling qualities and performance characteristics of a UAS while still in the design phase. As a test case, the ArcWulf UAS, flown by the Aerial Robotics Club at North Carolina State University was evaluated using the ESR model. The results of the evaluation indicated that

the ArcWulf could successfully comply with the ESR requirements in the three environmental conditions that were tested. The evaluation also highlighted areas of performance where the ArcWulf was the weakest. This allowed theoretical modifications to be made to the UAS airframe and the autopilot parameters to improve the outer-loop performance in these areas. The ESR model would also be beneficial to certifying authorities that were attempting to certify a UAS that they were given no data for. This could be done by following the steps outlined in Chapter 4. It is the hope of the author that the ESR model will contribute to the safe integration of UAS into the NAS and also the safe interaction between UAS and ATC.

References

- [1] SP-3300 Flight Research at Ames, 1940-1997, “Handling Qualities Rating Scale,”
Retrieved August 12, 2010, from Website: <http://history.nasa.gov/SP-3300/fig66.htm>
- [2] “Airworthiness Certification Criteria,” MIL-HDBK-516B, Department of Defense,
2005, pg. 41.
- [3] Robert K. Heffley, “Outer-loop Control Factors for Carrier Aircraft,” RHE-NAV-90-
TR-1, Robert Heffley Engineering, December 1990.
- [4] “Unmanned Aerial Vehicles Systems Airworthiness Requirements (USAR),”
STANAG 4671, NATO Standardization Agency, September 2009, pg. 1-B-10.
- [5] “Unmanned Aerial Vehicles Systems Airworthiness Requirements (USAR),”
STANAG 4671, NATO Standardization Agency, September 2009, pg. 1-B-12.
- [6] “Instrument Flying Handbook,” FAA-H-8083-15A With editorial change, U.S.
Department of Transportation, Federal Aviation Administration, 2008, pg. 5-19.
- [7] “Instrument Flying Handbook,” FAA-H-8083-15A With editorial change, U.S.
Department of Transportation, Federal Aviation Administration, 2008, pg. 5-37.
- [8] “Instrument Flying Handbook,” FAA-H-8083-15A With editorial change, U.S.
Department of Transportation, Federal Aviation Administration, 2008, pg. 5-53.
- [9] “Instrument Flying Handbook,” FAA-H-8083-15A With editorial change, U.S.
Department of Transportation, Federal Aviation Administration, 2008, pg. 7-41.

- [10] “Instrument Rating Practical Test Standards for Airplane, Helicopter, and Powered Lift,” FAA-S-8081-4E w/ change 1, U.S. Department of Transportation, Federal Aviation Administration, January 2010.
- [11] “Performance-based Navigation (PBN) Manual,” Doc 9613 AN937 Third Edition, International Civil Aviation Organization, 2008, pg. II-A-2-2.
- [12] “FAA’s NextGen Implementation Plan 2009,” Federal Aviation Administration, 2009, pg. 25.
- [13] “Instrument Procedures Handbook,” FAA-H-8261-1A, U.S. Department of Transportation, Federal Aviation Administration, 2007, pg. 1-13.
- [14] “Manual on Required Navigation Performance (RNP),” Doc 9613 AN/937 Second Edition, International Civil Aviation Organization, 1999, pg. 39.
- [15] “Introduction to TCAS II Version 7,” Federal Aviation Administration, U.S. Department of Transportation, November 2000.
- [16] “A bill to amend the Airport and Airway Improvement Act of 1982 for the purpose of extending the authorization of appropriations for airport and airway improvements, and for other purposes,” Pub. L. no. 100-223, December 1987.
- [17] “Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II),” RTCA DO-185B, RTCA, Inc., 2007.

- [18] Thomas B. Billingsley, "Safety Analysis of TCAS on Global Hawk using Airspace Encounter Models," Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2006.
- [19] "Airworthiness Approval of Traffic Alert And Collision Avoidance Systems (TCAS II), Versions 7.0 & 7.1 Associated Mode S Transponders," AC No: 20-151A, Federal Aviation Administration, U.S. Department of Transportation, September 2009.
- [20] Kuchar, J.,K., and Drumm, A., C., "The Traffic Alert Collision Avoidance System," Lincoln Laboratory Journal, Vol. 16, Num. 2, November 2007.
- [21] Vaglienti, B., Niculescu, M., and Hammitt, J., "Piccolo Simulator," Cloud Cap Technology, December 2009.
- [22] The MathWorks, "Dryden Wind Turbulence Model (Discrete)," Retrieved June 15, 2010, from The MathWorks Website:
<http://www.mathworks.com/access/helpdesk/help/toolbox/aeroblks/drydenwindturbulencemodeldiscrete.html>
- [23] Ed Williams, "Aviation Formulary V1.45," Retrieved June 10, 2010, from Aviation Formulary V1.45 Website: <http://williams.best.vwh.net/avform.htm>
- [24] Anderson, J. D., *Introduction to Flight*, 5th ed., New York, NY, 2005, pp. 173.
- [25] Vaglienti, B., "Setup and tuning of Piccolo control laws for 2.0.x," Cloud Cap Technology, Version 2.0.4, June 20, 2007.

APPENDIX

Appendix A ESR MATLAB Code

A.1 Main ESR Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Equivalent Safe Response Model                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% NAVAIR Airworthiness Airspace Integration Project
% John Southwell
% August 2010

% This code was written to accept the formatted log file that is output by
% the Piccolo 2.1.1 by Cloud Cap. Only simulations using the ESR course
% can be analyzed since the waypoint latitudes and longitudes are used to
% determine the required ground track. Other courses could be analyzed
% with this code with some modifications. Also, the UAS must reach all 51
% waypoints or an error will occur when running the code.

% This code allows the input file to be located anywhere that can be
% accessed from this computer. Once the input file has been selected the
% log file is read and organized into individual arrays containing data
% from specific columns from the log file. Note: Some data from the log
% file was not used in this code and therefore was not assigned a
variable.
% However, the matrix data contains all of the log file data, so a new
% vairable could easily be made if needed by assigning it to the proper
% column in the matrix data. Once the data has been analyzed the user
% will be prompted if he/she would like to save the results. If the user
% selects Yes, a .dat file will be created and will contain the ESR
results
% from the given input file. The user will also be prompted for the
% aircraft name and weather conditions during the flight or simulation.
If
% the user chooses not to save the results no file will be created. Next,
% the user will be prompted if he/she would like to plot any of the data.
% If the user selects Yes, the another menu appears and the user must
% select which flight phase to plot. Once a flight phase is selected
% pre-formatted plots will be generated for each test in that flight
phase.
% If the user chose to save the results the plots will be saved in .emf
% format.

% Units [ft, sec, deg] (NM for lateral deviations)

close all;
```



```

if size(lat,1) == n;
    lat(n,:) = [];
end
if size(lon,1) == n;
    lon(n,:) = [];
end
if size(Direction,1) == n;
    Direction(n,:) = [];
end
if size(Alt,1) == n;
    Alt(n,:) = [];
end
if size(TAS,1) == n;
    TAS(n,:) = [];
end
if size(Roll,1) == n;
    Roll(n,:) = [];
end
if size(AGL,1) == n;
    AGL(n,:) = [];
end
if size(Waypoint,1) == n;
    Waypoint(n,:) = [];
end
if size(IASTarget,1) == n;
    IASTarget(n,:) = [];
end
if size(AltTarget,1) == n;
    AltTarget(n,:) = [];
end
if size(BankTarget,1) == n;
    BankTarget(n,:) = [];
end
if size(HdgTarget,1) == n;
    HdgTarget(n,:) = [];
end
if size(VRateTarget,1) == n;
    VRateTarget(n,:) = [];
end
if size(Track_X,1) == n;
    Track_X(n,:) = [];
end
if size(Track_Y,1) == n;
    Track_Y(n,:) = [];
end
progressbar(189/1305);

%Input ESR course waypoint latitudes, longitudes, and altitudes (MSL)
%(degrees, ft)

```

```

waypoint_0 = [37.612806 -122.355130 499.92];
waypoint_1 = [37.607162 -122.341786 1500.16];
waypoint_2 = [37.605937 -122.338674 1500.16];
waypoint_3 = [37.608360 -122.336979 1500.16];
waypoint_4 = [37.617349 -122.330401 999.84];
waypoint_5 = [37.619795 -122.328634 999.84];
waypoint_6 = [37.622226 -122.326960 999.84];
waypoint_7 = [37.630196 -122.321054 999.84];
waypoint_8 = [37.633279 -122.325297 999.84];
waypoint_9 = [37.632085 -122.313217 999.84];
waypoint_10 = [37.636842 -122.322245 999.84];
waypoint_11 = [37.636279 -122.311404 999.84];
waypoint_12 = [37.646175 -122.317282 2999.92];
waypoint_13 = [37.648892 -122.318246 2999.92];
waypoint_14 = [37.651494 -122.319240 2999.92];
waypoint_15 = [37.656260 -122.328728 2500.00];
waypoint_16 = [37.657686 -122.331694 2500.00];
waypoint_17 = [37.658554 -122.334893 2500.00];
waypoint_18 = [37.662391 -122.349802 2500.00];
waypoint_19 = [37.666869 -122.350559 2500.00];
waypoint_20 = [37.660059 -122.354673 2500.00];
waypoint_21 = [37.666852 -122.356379 2500.00];
waypoint_22 = [37.660776 -122.359770 2500.00];
waypoint_23 = [37.653894 -122.356050 2999.92];
waypoint_24 = [37.651421 -122.354605 2999.92];
waypoint_25 = [37.649098 -122.352841 2999.92];
waypoint_26 = [37.641990 -122.345803 999.84];
waypoint_27 = [37.639859 -122.343554 999.84];
waypoint_28 = [37.638092 -122.346044 999.84];
waypoint_29 = [37.629405 -122.359220 999.84];
waypoint_30 = [37.624009 -122.355349 999.84];
waypoint_31 = [37.627431 -122.363878 999.84];
waypoint_32 = [37.621242 -122.359401 999.84];
waypoint_33 = [37.624169 -122.366337 999.84];
waypoint_34 = [37.630084 -122.381456 518.37];
waypoint_35 = [37.631374 -122.384564 518.37];
waypoint_36 = [37.632543 -122.387526 518.37];
waypoint_37 = [37.637307 -122.399677 518.37];
waypoint_38 = [37.632957 -122.403281 518.37];
waypoint_39 = [37.631622 -122.400214 518.37];
waypoint_40 = [37.620788 -122.374274 73.41];
waypoint_41 = [37.609901 -122.348306 518.37];
waypoint_42 = [37.608732 -122.345320 518.37];
waypoint_43 = [37.607393 -122.342266 518.37];
waypoint_44 = [37.606453 -122.339991 518.37];
waypoint_45 = [37.602346 -122.343036 518.37];
waypoint_46 = [37.603620 -122.346049 518.37];
waypoint_47 = [37.627911 -122.403372 518.37];
waypoint_48 = [37.629232 -122.406431 518.37];

```

```

waypoint_49 = [37.632989 -122.403303 518.37];
waypoint_50 = [37.631605 -122.400180 518.37];
waypoint_51 = [37.620782 -122.374276 73.41];
waypoint_geo = zeros(45,3);
waypoint_geo(1,:) = waypoint_0;
waypoint_geo(2,:) = waypoint_1;
waypoint_geo(3,:) = waypoint_2;
waypoint_geo(4,:) = waypoint_3;
waypoint_geo(5,:) = waypoint_4;
waypoint_geo(6,:) = waypoint_5;
waypoint_geo(7,:) = waypoint_6;
waypoint_geo(8,:) = waypoint_7;
waypoint_geo(9,:) = waypoint_8;
waypoint_geo(10,:) = waypoint_9;
waypoint_geo(11,:) = waypoint_10;
waypoint_geo(12,:) = waypoint_11;
waypoint_geo(13,:) = waypoint_12;
waypoint_geo(14,:) = waypoint_13;
waypoint_geo(15,:) = waypoint_14;
waypoint_geo(16,:) = waypoint_15;
waypoint_geo(17,:) = waypoint_16;
waypoint_geo(18,:) = waypoint_17;
waypoint_geo(19,:) = waypoint_18;
waypoint_geo(20,:) = waypoint_19;
waypoint_geo(21,:) = waypoint_20;
waypoint_geo(22,:) = waypoint_21;
waypoint_geo(23,:) = waypoint_22;
waypoint_geo(24,:) = waypoint_23;
waypoint_geo(25,:) = waypoint_24;
waypoint_geo(26,:) = waypoint_25;
waypoint_geo(27,:) = waypoint_26;
waypoint_geo(28,:) = waypoint_27;
waypoint_geo(29,:) = waypoint_28;
waypoint_geo(30,:) = waypoint_29;
waypoint_geo(31,:) = waypoint_30;
waypoint_geo(32,:) = waypoint_31;
waypoint_geo(33,:) = waypoint_32;
waypoint_geo(34,:) = waypoint_33;
waypoint_geo(35,:) = waypoint_34;
waypoint_geo(36,:) = waypoint_35;
waypoint_geo(37,:) = waypoint_36;
waypoint_geo(38,:) = waypoint_37;
waypoint_geo(39,:) = waypoint_38;
waypoint_geo(40,:) = waypoint_39;
waypoint_geo(41,:) = waypoint_40;
waypoint_geo(42,:) = waypoint_41;
waypoint_geo(43,:) = waypoint_42;
waypoint_geo(44,:) = waypoint_43;
waypoint_geo(45,:) = waypoint_44;

```

```

waypoint_geo(46,:) = waypoint_45;
waypoint_geo(47,:) = waypoint_46;
waypoint_geo(48,:) = waypoint_47;
waypoint_geo(49,:) = waypoint_48;
waypoint_geo(50,:) = waypoint_49;
waypoint_geo(51,:) = waypoint_50;
waypoint_geo(52,:) = waypoint_51;

fprintf('%s','Input data read. ');
toc(data_read)
progressbar(299/1305);
preanalysis = tic;

% Determine start and end time/index for each flight phase
n = size(Waypoint,1);
for i=1:1:n;
    if Waypoint(i,1) == 0;
        if TAS(i,1) > 0;
            dep1_start_i = i;
            dep1_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 1;
        dep1_end_i = i-1;
        dep1_end_time = clock(i-1,1);
        dep2_start_i = i;
        dep2_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_2 = 0;
    if Waypoint(i,1) == 2;
        orbit_2 = 1;
        dep2_end_i = i-1;
        dep2_end_time = clock(i-1,1);
        dep3_start_i = i;
        dep3_start_time = clock(i,1);
        j = i;
        break
    end
end
if orbit_2 == 0;
    if Waypoint(i,1) == 3;
        dep2_end_i = i-1;

```



```

        dep2_end_time = clock(i-1,1);
        dep3_start_i = i;
        dep3_start_time = clock(i,1);
        j = i;
        break
    end
end
end
for i=j:1:n;
    if Waypoint(i,1) == 3;
        dep3_end_i = i-1;
        dep3_end_time = clock(i-1,1);
        dep4_start_i = i;
        dep4_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 4;
        dep4_end_i = i-1;
        dep4_end_time = clock(i-1,1);
        dep5_start_i = i;
        dep5_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_5 = 0;
    if Waypoint(i,1) == 5;
        orbit_5 = 1;
        dep5_end_i = i-1;
        dep5_end_time = clock(i-1,1);
        dep6_start_i = i;
        dep6_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_5 == 0;
        if Waypoint(i,1) == 6;
            dep5_end_i = i-1;
            dep5_end_time = clock(i-1,1);
            dep6_start_i = i;
            dep6_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
end

```

```

end
for i=j:1:n;
    if Waypoint(i,1) == 6;
        dep6_end_i = i-1;
        dep6_end_time = clock(i-1,1);
        dep7_start_i = i;
        dep7_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 7;
        dep7_end_i = i-1;
        dep7_end_time = clock(i-1,1);
        dep8_start_i = i;
        dep8_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 8;
        dep8_end_i = i-1;
        dep8_end_time = clock(i-1,1);
        dep9_start_i = i;
        dep9_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 9;
        dep9_end_i = i-1;
        dep9_end_time = clock(i-1,1);
        dep10_start_i = i;
        dep10_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 10;
        dep10_end_i = i-1;
        dep10_end_time = clock(i-1,1);
        dep11_start_i = i;
        dep11_start_time = clock(i,1);
        j = i;
        break
    end
end

```

```

    end
end
for i=j:1:n;
    if Waypoint(i,1) == 11;
        dep11_end_i = i-1;
        dep11_end_time = clock(i-1,1);
        dep12_start_i = i;
        dep12_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 12;
        dep12_end_i = i-1;
        dep12_end_time = clock(i-1,1);
        enr1_start_i = i;
        enr1_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_13 = 0;
    if Waypoint(i,1) == 13;
        orbit_13 = 1;
        enr1_end_i = i-1;
        enr1_end_time = clock(i-1,1);
        enr2_start_i = i;
        enr2_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_13 == 0;
        if Waypoint(i,1) == 14;
            enr1_end_i = i-1;
            enr1_end_time = clock(i-1,1);
            enr2_start_i = i;
            enr2_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 14;
        enr2_end_i = i-1;
        enr2_end_time = clock(i-1,1);
        enr3_start_i = i;

```

```

        enr3_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 15;
        enr3_end_i = i-1;
        enr3_end_time = clock(i-1,1);
        enr4_start_i = i;
        enr4_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_16 = 0;
    if Waypoint(i,1) == 16;
        orbit_16 = 1;
        enr4_end_i = i-1;
        enr4_end_time = clock(i-1,1);
        enr5_start_i = i;
        enr5_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_16 == 0;
        if Waypoint(i,1) == 17;
            enr4_end_i = i-1;
            enr4_end_time = clock(i-1,1);
            enr5_start_i = i;
            enr5_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 17;
        enr5_end_i = i-1;
        enr5_end_time = clock(i-1,1);
        enr6_start_i = i;
        enr6_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 18;

```

```

        enr6_end_i = i-1;
        enr6_end_time = clock(i-1,1);
        enr7_start_i = i;
        enr7_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 19;
        enr7_end_i = i-1;
        enr7_end_time = clock(i-1,1);
        enr8_start_i = i;
        enr8_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 20;
        enr8_end_i = i-1;
        enr8_end_time = clock(i-1,1);
        enr9_start_i = i;
        enr9_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 21;
        enr9_end_i = i-1;
        enr9_end_time = clock(i-1,1);
        enr10_start_i = i;
        enr10_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 22;
        enr10_end_i = i-1;
        enr10_end_time = clock(i-1,1);
        enr11_start_i = i;
        enr11_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;

```

```

    if Waypoint(i,1) == 23;
        enr11_end_i = i-1;
        enr11_end_time = clock(i-1,1);
        arv1_start_i = i;
        arv1_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_24 = 0;
    if Waypoint(i,1) == 24;
        orbit_24 = 1;
        arv1_end_i = i-1;
        arv1_end_time = clock(i-1,1);
        arv2_start_i = i;
        arv2_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_24 == 0;
        if Waypoint(i,1) == 25;
            arv1_end_i = i-1;
            arv1_end_time = clock(i-1,1);
            arv2_start_i = i;
            arv2_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 25;
        arv2_end_i = i-1;
        arv2_end_time = clock(i-1,1);
        arv3_start_i = i;
        arv3_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 26;
        arv3_end_i = i-1;
        arv3_end_time = clock(i-1,1);
        arv4_start_i = i;
        arv4_start_time = clock(i,1);
        j = i;
        break
    end
end

```

```

    end
end
for i=j:1:n;
    orbit_27 = 0;
    if Waypoint(i,1) == 27;
        orbit_27 = 1;
        arv4_end_i = i-1;
        arv4_end_time = clock(i-1,1);
        arv5_start_i = i;
        arv5_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_27 == 0;
        if Waypoint(i,1) == 28;
            arv4_end_i = i-1;
            arv4_end_time = clock(i-1,1);
            arv5_start_i = i;
            arv5_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 28;
        arv5_end_i = i-1;
        arv5_end_time = clock(i-1,1);
        arv6_start_i = i;
        arv6_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 29;
        arv6_end_i = i-1;
        arv6_end_time = clock(i-1,1);
        arv7_start_i = i;
        arv7_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 30;
        arv7_end_i = i-1;
        arv7_end_time = clock(i-1,1);
        arv8_start_i = i;

```

```

        arv8_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 31;
        arv8_end_i = i-1;
        arv8_end_time = clock(i-1,1);
        arv9_start_i = i;
        arv9_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 32;
        arv9_end_i = i-1;
        arv9_end_time = clock(i-1,1);
        arv10_start_i = i;
        arv10_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 33;
        arv10_end_i = i-1;
        arv10_end_time = clock(i-1,1);
        arv11_start_i = i;
        arv11_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 34;
        arv11_end_i = i-1;
        arv11_end_time = clock(i-1,1);
        appl_start_i = i;
        appl_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_35 = 0;
    if Waypoint(i,1) == 35;
        orbit_35 = 1;
    end
end

```



```

        app1_end_i = i-1;
        app1_end_time = clock(i-1,1);
        app2_start_i = i;
        app2_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_35 == 0;
        if Waypoint(i,1) == 36;
            app1_end_i = i-1;
            app1_end_time = clock(i-1,1);
            app2_start_i = i;
            app2_start_time = clock(i,1);
            j = i;
            break
        end
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 36;
        app2_end_i = i-1;
        app2_end_time = clock(i-1,1);
        app3_start_i = i;
        app3_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 37;
        app3_end_i = i-1;
        app3_end_time = clock(i-1,1);
        app4_start_i = i;
        app4_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 38;
        app4_end_i = i-1;
        app4_end_time = clock(i-1,1);
        app5_start_i = i;
        app5_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;

```

```

    if Waypoint(i,1) == 39;
        app5_end_i = i-1;
        app5_end_time = clock(i-1,1);
        app6_start_i = i;
        app6_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 40;
        app6_end_i = i-1;
        app6_end_time = clock(i-1,1);
        app7_start_i = i;
        app7_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 41;
        app7_end_i = i-1;
        app7_end_time = clock(i-1,1);
        app8_start_i = i;
        app8_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    orbit_42 = 0;
    if Waypoint(i,1) == 42;
        orbit_42 = 1;
        app8_end_i = i-1;
        app8_end_time = clock(i-1,1);
        app9_start_i = i;
        app9_start_time = clock(i,1);
        j = i;
        break
    end
    if orbit_42 == 0;
        if Waypoint(i,1) == 43;
            app8_end_i = i-1;
            app8_end_time = clock(i-1,1);
            app9_start_i = i;
            app9_start_time = clock(i,1);
            j = i;
            break
        end
    end
end

```

```

    end
end
for i=j:1:n;
    if Waypoint(i,1) == 43;
        app9_end_i = i-1;
        app9_end_time = clock(i-1,1);
        app10_start_i = i;
        app10_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 44;
        app10_end_i = i-1;
        app10_end_time = clock(i-1,1);
        app11_start_i = i;
        app11_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 45;
        app11_end_i = i-1;
        app11_end_time = clock(i-1,1);
        app12_start_i = i;
        app12_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 46;
        app12_end_i = i-1;
        app12_end_time = clock(i-1,1);
        app13_start_i = i;
        app13_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 47;
        app13_end_i = i-1;
        app13_end_time = clock(i-1,1);
        app14_start_i = i;
        app14_start_time = clock(i,1);
        j = i;

```

```

        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 48;
        app14_end_i = i-1;
        app14_end_time = clock(i-1,1);
        app15_start_i = i;
        app15_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 49;
        app15_end_i = i-1;
        app15_end_time = clock(i-1,1);
        app16_start_i = i;
        app16_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 50;
        app16_end_i = i-1;
        app16_end_time = clock(i-1,1);
        app17_start_i = i;
        app17_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 51;
        app17_end_i = i-1;
        app17_end_time = clock(i-1,1);
        app18_start_i = i;
        app18_start_time = clock(i,1);
        j = i;
        break
    end
end
for i=j:1:n;
    if Waypoint(i,1) == 43;
        app18_end_i = i;
        app18_end_time = clock(i,1);
        break
    end
end

```

```

end
progressbar(927/1305);
for i=1:1:51;
    lat1 = waypoint_geo(i,1)*pi/180;
    lat2 = waypoint_geo(i+1,1)*pi/180;
    lon1 = waypoint_geo(i,2)*pi/180;
    lon2 = waypoint_geo(i+1,2)*pi/180;
    CourseTarget(i,1) = mod(atan2(sin(lon2-lon1)*cos(lat2),...
        cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)*cos(lon2-lon1)),2*pi)...
        *180/pi;
    if i == 1;
        dep2_CourseTarget = CourseTarget(i,1);
    end
    if i == 2;
        dep3_CourseTarget = CourseTarget(i,1);
    end
    if i == 3;
        dep4_CourseTarget = CourseTarget(i,1);
    end
    if i == 4;
        dep5_CourseTarget = CourseTarget(i,1);
    end
    if i == 5;
        dep6_CourseTarget = CourseTarget(i,1);
    end
    if i == 6;
        dep7_CourseTarget = CourseTarget(i,1);
    end
    if i == 7;
        dep8_CourseTarget = CourseTarget(i,1);
    end
    if i == 8;
        dep9_CourseTarget = CourseTarget(i,1);
    end
    if i == 9;
        dep10_CourseTarget = CourseTarget(i,1);
    end
    if i == 10;
        dep11_CourseTarget = CourseTarget(i,1);
    end
    if i == 11;
        dep12_CourseTarget = CourseTarget(i,1);
    end
    if i == 12;
        enr1_CourseTarget = CourseTarget(i,1);
    end
    if i == 13;
        enr2_CourseTarget = CourseTarget(i,1);
    end
end

```

```

if i == 14;
    enr3_CourseTarget = CourseTarget(i,1);
end
if i == 15;
    enr4_CourseTarget = CourseTarget(i,1);
end
if i == 16;
    enr5_CourseTarget = CourseTarget(i,1);
end
if i == 17;
    enr6_CourseTarget = CourseTarget(i,1);
end
if i == 18;
    enr7_CourseTarget = CourseTarget(i,1);
end
if i == 19;
    enr8_CourseTarget = CourseTarget(i,1);
end
if i == 20;
    enr9_CourseTarget = CourseTarget(i,1);
end
if i == 21;
    enr10_CourseTarget = CourseTarget(i,1);
end
if i == 22;
    enr11_CourseTarget = CourseTarget(i,1);
end
if i == 23;
    arv1_CourseTarget = CourseTarget(i,1);
end
if i == 24;
    arv2_CourseTarget = CourseTarget(i,1);
end
if i == 25;
    arv3_CourseTarget = CourseTarget(i,1);
end
if i == 26;
    arv4_CourseTarget = CourseTarget(i,1);
end
if i == 27;
    arv5_CourseTarget = CourseTarget(i,1);
end
if i == 28;
    arv6_CourseTarget = CourseTarget(i,1);
end
if i == 29;
    arv7_CourseTarget = CourseTarget(i,1);
end
if i == 30;

```

```

        arv8_CourseTarget = CourseTarget(i,1);
end
if i == 31;
    arv9_CourseTarget = CourseTarget(i,1);
end
if i == 32;
    arv10_CourseTarget = CourseTarget(i,1);
end
if i == 33;
    arv11_CourseTarget = CourseTarget(i,1);
end
if i == 34;
    app1_CourseTarget = CourseTarget(i,1);
end
if i == 35;
    app2_CourseTarget = CourseTarget(i,1);
end
if i == 36;
    app3_CourseTarget = CourseTarget(i,1);
end
if i == 37;
    app4_CourseTarget = CourseTarget(i,1);
end
if i == 38;
    app5_CourseTarget = CourseTarget(i,1);
end
if i == 39;
    app6_CourseTarget = CourseTarget(i,1);
end
if i == 40;
    app7_CourseTarget = CourseTarget(i,1);
end
if i == 41;
    app8_CourseTarget = CourseTarget(i,1);
end
if i == 42;
    app9_CourseTarget = CourseTarget(i,1);
end
if i == 43;
    app10_CourseTarget = CourseTarget(i,1);
end
if i == 44;
    app11_CourseTarget = CourseTarget(i,1);
end
if i == 45;
    app12_CourseTarget = CourseTarget(i,1);
end
if i == 46;
    app13_CourseTarget = CourseTarget(i,1);

```

```

end
if i == 47;
    app14_CourseTarget = CourseTarget(i,1);
end
if i == 48;
    app15_CourseTarget = CourseTarget(i,1);
end
if i == 49;
    app16_CourseTarget = CourseTarget(i,1);
end
if i == 50;
    app17_CourseTarget = CourseTarget(i,1);
end
if i == 51;
    app18_CourseTarget = CourseTarget(i,1);
end
end

% Convert target indicated airspeed to true airspeed
TASTarget = TASTarget(Static,Dynamic,OAT,IASTarget,n);

fprintf('%s','Pre-analysis complete. ');
toc(preanalysis);
progressbar(1092/1305);
departure = tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                               Flight Phase Analysis
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Departure Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dep_track_y = 0;

% Departure Phase 2 Climb Rate Test
[dep2_alt_dev,dep2_hdg_dev,dep2_bank_dev,dep2_TAS_dev,dep_track_y,...
    dep2_max_alt_dev,dep2_max_hdg_dev,dep2_max_bank_dev,...
    dep2_max_TAS_dev,dep2_avg_climbrate,dep2_avg_turnrate,...
    dep2_max_turnrate,dep2_max_bankangle,dep2_turnrate,dep2_climbrate]...
=
flight_phase_analysis(clock_sec,dep_track_y,Track_Y,Alt,AltTarget,...
    Direction,dep2_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,dep2_start_i,dep2_end_i);

```



```

% Departure Phase 5 Descent Rate Test
[dep5_alt_dev,dep5_hdg_dev,dep5_bank_dev,dep5_TAS_dev,dep_track_y,...
 dep5_max_alt_dev,dep5_max_hdg_dev,dep5_max_bank_dev,...
 dep5_max_TAS_dev,dep5_avg_climbrate,dep5_avg_turnrate,...
 dep5_max_turnrate,dep5_max_bankangle,dep5_turnrate,dep5_climbrate]...
=
flight_phase_analysis(clock_sec,dep_track_y,Track_Y,Alt,AltTarget,...
 Direction,dep5_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
 clock,dep5_start_i,dep5_end_i);

% Departure Phase 8 Straight and Level Test
[dep8_alt_dev,dep8_hdg_dev,dep8_bank_dev,dep8_TAS_dev,dep_track_y,...
 dep8_max_alt_dev,dep8_max_hdg_dev,dep8_max_bank_dev,...
 dep8_max_TAS_dev,dep8_avg_climbrate,dep8_avg_turnrate,...
 dep8_max_turnrate,dep8_max_bankangle,dep8_turnrate,dep8_climbrate]...
=
flight_phase_analysis(clock_sec,dep_track_y,Track_Y,Alt,AltTarget,...
 Direction,dep8_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
 clock,dep8_start_i,dep8_end_i);

% Departure Phase 9 Turn Rate Test
[dep9_alt_dev,dep9_hdg_dev,dep9_bank_dev,dep9_TAS_dev,dep_track_y,...
 dep9_max_alt_dev,dep9_max_hdg_dev,dep9_max_bank_dev,...
 dep9_max_TAS_dev,dep9_avg_climbrate,dep9_avg_turnrate,...
 dep9_max_turnrate,dep9_max_bankangle,dep9_turnrate,dep9_climbrate]...
=
flight_phase_analysis(clock_sec,dep_track_y,Track_Y,Alt,AltTarget,...
 Direction,dep9_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
 clock,dep9_start_i,dep9_end_i);

% Departure Phase 10 Turn Rate Test
[dep10_alt_dev,dep10_hdg_dev,dep10_bank_dev,dep10_TAS_dev,...
 dep_track_y,dep10_max_alt_dev,dep10_max_hdg_dev,...
 dep10_max_bank_dev,dep10_max_TAS_dev,dep10_avg_climbrate,...
 dep10_avg_turnrate,dep10_max_turnrate,dep10_max_bankangle,...
 dep10_turnrate,dep10_climbrate] = flight_phase_analysis(clock_sec,...
 dep_track_y,Track_Y,Alt,AltTarget,Direction,dep10_CourseTarget,...
 Roll,BankTarget,TAS,TASTarget,clock,dep10_start_i,dep10_end_i);

% Departure Phase 11 Turn Rate Test
[dep11_alt_dev,dep11_hdg_dev,dep11_bank_dev,dep11_TAS_dev,dep_track_y,...
 dep11_max_alt_dev,dep11_max_hdg_dev,dep11_max_bank_dev,...
 dep11_max_TAS_dev,dep11_avg_climbrate,dep11_avg_turnrate,...
 dep11_max_turnrate,dep11_max_bankangle,dep11_turnrate,...
 dep11_climbrate] = flight_phase_analysis(clock_sec,dep_track_y,...
 Track_Y,Alt,AltTarget,Direction,dep11_CourseTarget,Roll,...
 BankTarget,TAS,TASTarget,clock,dep11_start_i,dep11_end_i);

```

```

% Departure Phase 12 Turn Rate Test
[dep12_alt_dev,dep12_hdg_dev,dep12_bank_dev,dep12_TAS_dev,dep_track_y,...
    dep12_max_alt_dev,dep12_max_hdg_dev,dep12_max_bank_dev,...
    dep12_max_TAS_dev,dep12_avg_climbrate,dep12_avg_turnrate,...
    dep12_max_turnrate,dep12_max_bankangle,dep12_turnrate,...
    dep12_climbrate] = flight_phase_analysis(clock_sec,dep_track_y,...
    Track_Y,Alt,AltTarget,Direction,dep12_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,dep12_start_i,dep12_end_i);

% Departure Phase RNP Test
dep_max_track_y = 3*std(dep_track_y(:,1));

dep9_12_turnrate = [dep9_turnrate; dep10_turnrate; dep11_turnrate;...
    dep12_turnrate];

[dep2_esr_vrateline] = ESRplot_alt(dep2_start_i,dep2_end_i,AltTarget,...
    Alt,clock_sec);

[dep5_esr_vrateline] = ESRplot_alt(dep5_start_i,dep5_end_i,...
    AltTarget,Alt,clock_sec);

fprintf('%s','Departure analysis complete. ');
toc(departure);
progressbar(1111/1305);
enroute = tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% En Route Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

enr_track_y = 0;

% En Route Phase 1 Climb Rate Test
[enr1_alt_dev,enr1_hdg_dev,enr1_bank_dev,enr1_TAS_dev,enr_track_y,...
    enr1_max_alt_dev,enr1_max_hdg_dev,enr1_max_bank_dev,...
    enr1_max_TAS_dev,enr1_avg_climbrate,enr1_avg_turnrate,...
    enr1_max_turnrate,enr1_max_bankangle,enr1_turnrate,enr1_climbrate]...
=
flight_phase_analysis(clock_sec,enr_track_y,Track_Y,Alt,AltTarget,...
    Direction,enr1_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,enr1_start_i,enr1_end_i);

% En Route Phase 4 Descent Rate Test
[enr4_alt_dev,enr4_hdg_dev,enr4_bank_dev,enr4_TAS_dev,enr_track_y,...
    enr4_max_alt_dev,enr4_max_hdg_dev,enr4_max_bank_dev,...
    enr4_max_TAS_dev,enr4_avg_climbrate,enr4_avg_turnrate,...
    enr4_max_turnrate,enr4_max_bankangle,enr4_turnrate,enr4_climbrate]...

```

```

=
flight_phase_analysis(clock_sec,enr_track_y,Track_Y,Alt,AltTarget,...
    Direction,enr4_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,enr4_start_i,enr4_end_i);

% En Route Phase 7 Straight and Level Test
[enr7_alt_dev,enr7_hdg_dev,enr7_bank_dev,enr7_TAS_dev,enr_track_y,...
    enr7_max_alt_dev,enr7_max_hdg_dev,enr7_max_bank_dev,...
    enr7_max_TAS_dev,enr7_avg_climbrate,enr7_avg_turnrate,...
    enr7_max_turnrate,enr7_max_bankangle,enr7_turnrate,enr7_climbrate]...
=
flight_phase_analysis(clock_sec,enr_track_y,Track_Y,Alt,AltTarget,...
    Direction,enr7_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,enr7_start_i,enr7_end_i);

% En Route Phase 8 Turn Rate Test
[enr8_alt_dev,enr8_hdg_dev,enr8_bank_dev,enr8_TAS_dev,enr_track_y,...
    enr8_max_alt_dev,enr8_max_hdg_dev,enr8_max_bank_dev,...
    enr8_max_TAS_dev,enr8_avg_climbrate,enr8_avg_turnrate,...
    enr8_max_turnrate,enr8_max_bankangle,enr8_turnrate,enr8_climbrate]...
=
flight_phase_analysis(clock_sec,enr_track_y,Track_Y,Alt,AltTarget,...
    Direction,enr8_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,enr8_start_i,enr8_end_i);

% En Route Phase 9 Turn Rate Test
[enr9_alt_dev,enr9_hdg_dev,enr9_bank_dev,enr9_TAS_dev,enr_track_y,...
    enr9_max_alt_dev,enr9_max_hdg_dev,enr9_max_bank_dev,...
    enr9_max_TAS_dev,enr9_avg_climbrate,enr9_avg_turnrate,...
    enr9_max_turnrate,enr9_max_bankangle,enr9_turnrate,enr9_climbrate]...
=
flight_phase_analysis(clock_sec,enr_track_y,Track_Y,Alt,AltTarget,...
    Direction,enr9_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,enr9_start_i,enr9_end_i);

% En Route Phase 10 Turn Rate Test
[enr10_alt_dev,enr10_hdg_dev,enr10_bank_dev,enr10_TAS_dev,enr_track_y,...
    enr10_max_alt_dev,enr10_max_hdg_dev,enr10_max_bank_dev,...
    enr10_max_TAS_dev,enr10_avg_climbrate,enr10_avg_turnrate,...
    enr10_max_turnrate,enr10_max_bankangle,enr10_turnrate,...
    enr10_climbrate] = flight_phase_analysis(clock_sec,enr_track_y,...
    Track_Y,Alt,AltTarget,Direction,enr10_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,enr10_start_i,enr10_end_i);

% En Route Phase 11 Turn Rate Test
[enr11_alt_dev,enr11_hdg_dev,enr11_bank_dev,enr11_TAS_dev,enr_track_y,...
    enr11_max_alt_dev,enr11_max_hdg_dev,enr11_max_bank_dev,...
    enr11_max_TAS_dev,enr11_avg_climbrate,enr11_avg_turnrate,...

```

```

    enr11_max_turnrate,enr11_max_bankangle,enr11_turnrate,...
    enr11_climbrate] = flight_phase_analysis(clock_sec,enr_track_y,...
    Track_Y,Alt,AltTarget,Direction,enr11_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,enr11_start_i,enr11_end_i);

% En Route Phase RNP Rate Test
enr_max_track_y = 3*std(enr_track_y(:,1));

enr8_11_turnrate = [enr8_turnrate; enr9_turnrate; enr10_turnrate;...
    enr11_turnrate];

[enr1_esr_vrateline] = ESRplot_alt(enr1_start_i,enr1_end_i,AltTarget,...
    Alt,clock_sec);

[enr4_esr_vrateline] = ESRplot_alt(enr4_start_i,enr4_end_i,AltTarget,...
    Alt,clock_sec);

fprintf('%s','En-route analysis complete. ');
toc(enroute);
progressbar(1130/1305);
arrival = tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Arrival Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

arv_track_y = 0;

% Arrival Phase 1 Climb Rate Test
[arv1_alt_dev,arv1_hdg_dev,arv1_bank_dev,arv1_TAS_dev,arv_track_y,...
    arv1_max_alt_dev,arv1_max_hdg_dev,arv1_max_bank_dev,...
    arv1_max_TAS_dev,arv1_avg_climbrate,arv1_avg_turnrate,...
    arv1_max_turnrate,arv1_max_bankangle,arv1_turnrate,arv1_climbrate]...
=
flight_phase_analysis(clock_sec,arv_track_y,Track_Y,Alt,AltTarget,...
    Direction,arv1_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,arv1_start_i,arv1_end_i);

% Arrival Phase 4 Descent Rate Test
[arv4_alt_dev,arv4_hdg_dev,arv4_bank_dev,arv4_TAS_dev,arv_track_y,...
    arv4_max_alt_dev,arv4_max_hdg_dev,arv4_max_bank_dev,...
    arv4_max_TAS_dev,arv4_avg_climbrate,arv4_avg_turnrate,...
    arv4_max_turnrate,arv4_max_bankangle,arv4_turnrate,arv4_climbrate]...
=
flight_phase_analysis(clock_sec,arv_track_y,Track_Y,Alt,AltTarget,...
    Direction,arv4_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,arv4_start_i,arv4_end_i);

```

```

% Arrival Phase 7 Straight and Level Test
[arv7_alt_dev,arv7_hdg_dev,arv7_bank_dev,arv7_TAS_dev,arv_track_y,...
  arv7_max_alt_dev,arv7_max_hdg_dev,arv7_max_bank_dev,...
  arv7_max_TAS_dev,arv7_avg_climbrate,arv7_avg_turnrate,...
  arv7_max_turnrate,arv7_max_bankangle,arv7_turnrate,arv7_climbrate]...
=
flight_phase_analysis(clock_sec,arv_track_y,Track_Y,Alt,AltTarget,...
  Direction,arv7_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
  clock,arv7_start_i,arv7_end_i);

% Arrival Phase 8 Turn Rate Test
[arv8_alt_dev,arv8_hdg_dev,arv8_bank_dev,arv8_TAS_dev,arv_track_y,...
  arv8_max_alt_dev,arv8_max_hdg_dev,arv8_max_bank_dev,...
  arv8_max_TAS_dev,arv8_avg_climbrate,arv8_avg_turnrate,...
  arv8_max_turnrate,arv8_max_bankangle,arv8_turnrate,arv8_climbrate]...
=
flight_phase_analysis(clock_sec,arv_track_y,Track_Y,Alt,AltTarget,...
  Direction,arv8_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
  clock,arv8_start_i,arv8_end_i);

% Arrival Phase 9 Turn Rate Test
[arv9_alt_dev,arv9_hdg_dev,arv9_bank_dev,arv9_TAS_dev,arv_track_y,...
  arv9_max_alt_dev,arv9_max_hdg_dev,arv9_max_bank_dev,...
  arv9_max_TAS_dev,arv9_avg_climbrate,arv9_avg_turnrate,...
  arv9_max_turnrate,arv9_max_bankangle,arv9_turnrate,arv9_climbrate]...
=
flight_phase_analysis(clock_sec,arv_track_y,Track_Y,Alt,AltTarget,...
  Direction,arv9_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
  clock,arv9_start_i,arv9_end_i);

% Arrival Phase 10 Turn Rate Test
[arv10_alt_dev,arv10_hdg_dev,arv10_bank_dev,arv10_TAS_dev,arv_track_y,...
  arv10_max_alt_dev,arv10_max_hdg_dev,arv10_max_bank_dev,...
  arv10_max_TAS_dev,arv10_avg_climbrate,arv10_avg_turnrate,...
  arv10_max_turnrate,arv10_max_bankangle,arv10_turnrate,...
  arv10_climbrate] = flight_phase_analysis(clock_sec,arv_track_y,...
  Track_Y,Alt,AltTarget,Direction,arv10_CourseTarget,Roll,...
  BankTarget,TAS,TASTarget,clock,arv10_start_i,arv10_end_i);

% Arrival Phase 11 Turn Rate Test
[arv11_alt_dev,arv11_hdg_dev,arv11_bank_dev,arv11_TAS_dev,arv_track_y,...
  arv11_max_alt_dev,arv11_max_hdg_dev,arv11_max_bank_dev,...
  arv11_max_TAS_dev,arv11_avg_climbrate,arv11_avg_turnrate,...
  arv11_max_turnrate,arv11_max_bankangle,arv11_turnrate,...
  arv11_climbrate] = flight_phase_analysis(clock_sec,arv_track_y,...
  Track_Y,Alt,AltTarget,Direction,arv11_CourseTarget,Roll,...
  BankTarget,TAS,TASTarget,clock,arv11_start_i,arv11_end_i);

```

```

% Arrival Phase RNP Rate Test
arv_max_track_y = 3*std(arv_track_y(:,1));

arv8_11_turnrate = [arv8_turnrate; arv9_turnrate; arv10_turnrate;...
    arv11_turnrate];

[arv1_esr_vrateline] = ESRplot_alt(arv1_start_i,arv1_end_i,AltTarget,...
    Alt,clock_sec);

[arv4_esr_vrateline] = ESRplot_alt(arv4_start_i,arv4_end_i,AltTarget,...
    Alt,clock_sec);

fprintf('%s','Arrival analysis complete. ');
toc(arrival);
progressbar(1149/1305);
approach = tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Approach Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

app_track_y = 0;

% Approach Phase 1 Descent Rate Test
[app1_alt_dev,app1_hdg_dev,app1_bank_dev,app1_TAS_dev,app_track_y,...
    app1_max_alt_dev,app1_max_hdg_dev,app1_max_bank_dev,...
    app1_max_TAS_dev,app1_avg_climbrate,app1_avg_turnrate,...
    app1_max_turnrate,app1_max_bankangle,app1_turnrate,app1_climbrate]...
= flight_phase_analysis(clock_sec,app_track_y,Track_Y,Alt,...
    AltTarget,Direction,app1_CourseTarget,Roll,BankTarget,TAS,...
    TASTarget,clock,app1_start_i,app1_end_i);

% Approach Phase 5 Turn Rate Test
[app5_alt_dev,app5_hdg_dev,app5_bank_dev,app5_TAS_dev,app_track_y,...
    app5_max_alt_dev,app5_max_hdg_dev,app5_max_bank_dev,...
    app5_max_TAS_dev,app5_avg_climbrate,app5_avg_turnrate,...
    app5_max_turnrate,app5_max_bankangle,app5_turnrate,app5_climbrate]...
=
flight_phase_analysis(clock_sec,app_track_y,Track_Y,Alt,AltTarget,...
    Direction,app5_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,app5_start_i,app5_end_i);

% Approach Phase 6 Turn Rate Test
[app6_alt_dev,app6_hdg_dev,app6_bank_dev,app6_TAS_dev,app_track_y,...
    app6_max_alt_dev,app6_max_hdg_dev,app6_max_bank_dev,...
    app6_max_TAS_dev,app6_avg_climbrate,app6_avg_turnrate,...
    app6_max_turnrate,app6_max_bankangle,app6_turnrate,app6_climbrate]...

```

```

=
flight_phase_analysis(clock_sec,app_track_y,Track_Y,Alt,AltTarget,...
    Direction,app6_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,app6_start_i,app6_end_i);

% Approach Phase 7 Final Approach Test
[app7_alt_dev,app7_hdg_dev,app7_bank_dev,app7_TAS_dev,app_track_y,...
    app7_max_alt_dev,app7_max_hdg_dev,app7_max_bank_dev,...
    app7_max_TAS_dev,app7_avg_climbrate,app7_avg_turnrate,...
    app7_max_turnrate,app7_max_bankangle,app7_turnrate,app7_climbrate]...
=
flight_phase_analysis(clock_sec,app_track_y,Track_Y,Alt,AltTarget,...
    Direction,app7_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,app7_start_i,app7_end_i);

% Approach Phase 8 Missed Approach Test
[app8_alt_dev,app8_hdg_dev,app8_bank_dev,app8_TAS_dev,app_track_y,...
    app8_max_alt_dev,app8_max_hdg_dev,app8_max_bank_dev,...
    app8_max_TAS_dev,app8_avg_climbrate,app8_avg_turnrate,...
    app8_max_turnrate,app8_max_bankangle,app8_turnrate,app8_climbrate]...
=
flight_phase_analysis(clock_sec,app_track_y,Track_Y,Alt,AltTarget,...
    Direction,app8_CourseTarget,Roll,BankTarget,TAS,TASTarget,...
    clock,app8_start_i,app8_end_i);

% Approach Phase 12 Turn Rate Test
[app12_alt_dev,app12_hdg_dev,app12_bank_dev,app12_TAS_dev,app_track_y,...
    app12_max_alt_dev,app12_max_hdg_dev,app12_max_bank_dev,...
    app12_max_TAS_dev,app12_avg_climbrate,app12_avg_turnrate,...
    app12_max_turnrate,app12_max_bankangle,app12_turnrate,...
    app12_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app12_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app12_start_i,app12_end_i);

% Approach Phase 13 Turn Rate Test
[app13_alt_dev,app13_hdg_dev,app13_bank_dev,app13_TAS_dev,app_track_y,...
    app13_max_alt_dev,app13_max_hdg_dev,app13_max_bank_dev,...
    app13_max_TAS_dev,app13_avg_climbrate,app13_avg_turnrate,...
    app13_max_turnrate,app13_max_bankangle,app13_turnrate,...
    app13_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app13_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app13_start_i,app13_end_i);

% Approach Phase 14 Straight and Level Test
[app14_alt_dev,app14_hdg_dev,app14_bank_dev,app14_TAS_dev,app_track_y,...
    app14_max_alt_dev,app14_max_hdg_dev,app14_max_bank_dev,...
    app14_max_TAS_dev,app14_avg_climbrate,app14_avg_turnrate,...
    app14_max_turnrate,app14_max_bankangle,app14_turnrate,...

```

```

    app14_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app14_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app14_start_i,app14_end_i);

% Approach Phase 16 Turn Rate Test
[app16_alt_dev,app16_hdg_dev,app16_bank_dev,app16_TAS_dev,app_track_y,...
    app16_max_alt_dev,app16_max_hdg_dev,app16_max_bank_dev,...
    app16_max_TAS_dev,app16_avg_climbrate,app16_avg_turnrate,...
    app16_max_turnrate,app16_max_bankangle,app16_turnrate,...
    app16_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app16_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app16_start_i,app16_end_i);

% Approach Phase 17 Turn Rate Test
[app17_alt_dev,app17_hdg_dev,app17_bank_dev,app17_TAS_dev,app_track_y,...
    app17_max_alt_dev,app17_max_hdg_dev,app17_max_bank_dev,...
    app17_max_TAS_dev,app17_avg_climbrate,app17_avg_turnrate,...
    app17_max_turnrate,app17_max_bankangle,app17_turnrate,...
    app17_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app17_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app17_start_i,app17_end_i);

% Approach Phase 18 Final Approach Test
[app18_alt_dev,app18_hdg_dev,app18_bank_dev,app18_TAS_dev,app_track_y,...
    app18_max_alt_dev,app18_max_hdg_dev,app18_max_bank_dev,...
    app18_max_TAS_dev,app18_avg_climbrate,app18_avg_turnrate,...
    app18_max_turnrate,app18_max_bankangle,app18_turnrate,...
    app18_climbrate] = flight_phase_analysis(clock_sec,app_track_y,...
    Track_Y,Alt,AltTarget,Direction,app18_CourseTarget,Roll,...
    BankTarget,TAS,TASTarget,clock,app18_start_i,app18_end_i);

% Approach Phase RNP Test
app_max_track_y = 3*std(app_track_y(:,1));

app5_6_turnrate = [app5_turnrate; app6_turnrate];

app12_13_turnrate = [app12_turnrate; app13_turnrate];

app16_17_turnrate = [app16_turnrate; app17_turnrate];

[app1_esr_vrateline] = ESRplot_alt(app1_start_i,app1_end_i,AltTarget,...
    Alt,clock_sec);

[app7_esr_altline,app7_esr_latline] = ESRplot_approach(app7_start_i,...
    app7_end_i,AltTarget,Track_X);

[app8_esr_altline,app8_esr_latline] = ESRplot_misapproach(app8_start_i,...

```



```

    app8_end_i,AltTarget,Track_X);

[app18_esr_altline,app18_esr_latline] = ESRplot_approach(app18_start_i,...
    app18_end_i,AltTarget,Track_X);

fprintf('%s','Approach analysis complete. ');
toc(approach);
progressbar(1175/1305);
ESR_time = tic;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                               ESR Evaluation
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Departure Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Departure Phase 2 Climb Rate Test
[dep2_esr_climbrate,dep2_esr_descentrate] = ESRclimb(dep2_avg_climbrate);

% Departure Phase 5 Descent Rate Test
[dep5_esr_climbrate,dep5_esr_descentrate] = ESRclimb(dep5_avg_climbrate);

% Departure Phase 8 Straight and Level Test
[dep8_esr_alt,dep8_esr_hdg,dep8_esr_TAS,dep8_esr_straightlevel]...
=
ESRstraightlevel(dep8_max_alt_dev,dep8_max_hdg_dev,dep8_max_TAS_dev);

% Departure Phase 9 Turn Rate Test
[dep9_esr_turnrate] = ESRturn(dep9_avg_turnrate);

% Departure Phase 10 Turn Rate Test
[dep10_esr_turnrate] = ESRturn(dep10_avg_turnrate);

% Departure Phase 11 Turn Rate Test
[dep11_esr_turnrate] = ESRturn(dep11_avg_turnrate);

% Departure Phase 12 Turn Rate Test
[dep12_esr_turnrate] = ESRturn(dep12_avg_turnrate);

% Departure Phase RNP Test
if dep_max_track_y < 0.125;
    dep_esr_rnp = 'Pass';
else
    dep_esr_rnp = 'Fail';

```

```

end

TF1 = strcmp('Pass',dep2_esr_climbrate);
TF2 = strcmp('Pass',dep5_esr_descentrate);
TF3 = strcmp('Pass',dep8_esr_straightlevel);
TF4 = strcmp('Pass',dep9_esr_turnrate);
TF5 = strcmp('Pass',dep10_esr_turnrate);
TF6 = strcmp('Pass',dep11_esr_turnrate);
TF7 = strcmp('Pass',dep12_esr_turnrate);
TF8 = strcmp('Pass',dep_esr_rnp);
if (TF1 && TF2 && TF3 && TF4 && TF5 && TF6 && TF7 && TF8) == 1;
    dep_esr = 'Pass';
else
    dep_esr = 'Fail';
end
progressbar(1205/1305);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% En Route Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% En Route Phase 1 Climb Rate Test
[enr1_esr_climbrate,enr1_esr_descentrate] = ESRclimb(enr1_avg_climbrate);

% En Route Phase 4 Descent Rate Test
[enr4_esr_climbrate,enr4_esr_descentrate] = ESRclimb(enr4_avg_climbrate);

% En Route Phase 7 Straight and Level Test
[enr7_esr_alt,enr7_esr_hdg,enr7_esr_TAS,enr7_esr_straightlevel]...
=
ESRstraightlevel(enr7_max_alt_dev,enr7_max_hdg_dev,enr7_max_TAS_dev);

% En Route Phase 8 - 11 Turn Rate Test is not graded and is only for
% reference

% En Route Phase RNP Rate Test
if enr_max_track_y < 0.25;
    enr_esr_rnp = 'Pass';
else
    enr_esr_rnp = 'Fail';
end

TF1 = strcmp('Pass',enr1_esr_climbrate);
TF2 = strcmp('Pass',enr4_esr_descentrate);
TF3 = strcmp('Pass',enr7_esr_straightlevel);
TF8 = strcmp('Pass',enr_esr_rnp);
if (TF1 && TF2 && TF3 && TF8) == 1;
    enr_esr = 'Pass';
else

```

```

        enr_esr = 'Fail';
end
progressbar(1225/1305);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Arrival Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Arrival Phase 1 Climb Rate Test
[arv1_esr_climbrate,arv1_esr_descentrate] = ESRclimb(arv1_avg_climbrate);

% Arrival Phase 4 Descent Rate Test
[arv4_esr_climbrate,arv4_esr_descentrate] = ESRclimb(arv4_avg_climbrate);

% Arrival Phase 7 Straight and Level Test
[arv7_esr_alt,arv7_esr_hdg,arv7_esr_TAS,arv7_esr_straightlevel]...
=
ESRstraightlevel(arv7_max_alt_dev,arv7_max_hdg_dev,arv7_max_TAS_dev);

% Arrival Phase 8 Turn Rate Test
[arv8_esr_turnrate] = ESRturn(arv8_avg_turnrate);

% Arrival Phase 9 Turn Rate Test
[arv9_esr_turnrate] = ESRturn(arv9_avg_turnrate);

% Arrival Phase 10 Turn Rate Test
[arv10_esr_turnrate] = ESRturn(arv10_avg_turnrate);

% Arrival Phase 11 Turn Rate Test
[arv11_esr_turnrate] = ESRturn(arv11_avg_turnrate);

% Arrival Phase RNP Rate Test
if arv_max_track_y < 0.25;
    arv_esr_rnp = 'Pass';
else
    arv_esr_rnp = 'Fail';
end

TF1 = strcmp('Pass',arv1_esr_climbrate);
TF2 = strcmp('Pass',arv4_esr_descentrate);
TF3 = strcmp('Pass',arv7_esr_straightlevel);
TF4 = strcmp('Pass',arv8_esr_turnrate);
TF5 = strcmp('Pass',arv9_esr_turnrate);
TF6 = strcmp('Pass',arv10_esr_turnrate);
TF7 = strcmp('Pass',arv11_esr_turnrate);
TF8 = strcmp('Pass',arv_esr_rnp);
if (TF1 && TF2 && TF3 && TF4 && TF5 && TF6 && TF7 && TF8) == 1;
    arv_esr = 'Pass';

```

```

else
    arv_esr = 'Fail';
end
progressbar(1253/1305);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Approach Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Approach Phase 1 Descent Rate Test
[app1_esr_climbrate,app1_esr_descentrate] = ESRclimb(app1_avg_climbrate);

% Approach Phase 5 Turn Rate Test
[app5_esr_turnrate] = ESRturn(app5_avg_turnrate);

% Approach Phase 6 Turn Rate Test
[app6_esr_turnrate] = ESRturn(app6_avg_turnrate);

% Approach Phase 7 Final Approach Test
[app7_esr_alt,app7_esr_TAS,app7_esr_lat,app7_esr_approach,...
    app7_gs_dev_max,app7_lat_dev_max] = ESRFinalApproach(app7_alt_dev,...
    app7_max_TAS_dev,Track_X,Track_Y,app7_start_i,app7_end_i);

% Approach Phase 8 Missed Approach Test
[app8_esr_alt,app8_esr_TAS,app8_esr_lat,app8_esr_approach,...
    app8_gs_dev_max,app8_lat_dev_max] = ESRMisApproach(app8_alt_dev,...
    app8_max_TAS_dev,Track_X,Track_Y,app8_start_i,app8_end_i);

% Approach Phase 12 Turn Rate Test
[app12_esr_turnrate] = ESRturn(app12_avg_turnrate);

% Approach Phase 13 Turn Rate Test
[app13_esr_turnrate] = ESRturn(app13_avg_turnrate);

% Approach Phase 14 Straight and Level Test
[app14_esr_alt,app14_esr_hdg,app14_esr_TAS,app14_esr_straightlevel]...
    = ESRstraightlevel(app14_max_alt_dev,app14_max_hdg_dev,...
    app14_max_TAS_dev);

% Approach Phase 16 Turn Rate Test
[app16_esr_turnrate] = ESRturn(app16_avg_turnrate);

% Approach Phase 17 Turn Rate Test
[app17_esr_turnrate] = ESRturn(app17_avg_turnrate);

% Approach Phase 18 Final Approach Test
[app18_esr_alt,app18_esr_TAS,app18_esr_lat,app18_esr_approach,...
    app18_gs_dev_max,app18_lat_dev_max] = ESRFinalApproach...

```

```

        (appl8_alt_dev,appl8_max_TAS_dev,Track_X,Track_Y,appl8_start_i,...
        appl8_end_i);

% Approach Phase RNP Test
if app_max_track_y < 0.125;
    app_esr_rnp = 'Pass';
else
    app_esr_rnp = 'Fail';
end

TF1 = strcmp('Pass',app1_esr_descentrate);
TF2 = strcmp('Pass',app5_esr_turnrate);
TF3 = strcmp('Pass',app6_esr_turnrate);
TF4 = strcmp('Pass',app7_esr_approach);
TF5 = strcmp('Pass',app8_esr_approach);
TF6 = strcmp('Pass',app12_esr_turnrate);
TF7 = strcmp('Pass',app13_esr_turnrate);
TF8 = strcmp('Pass',app14_esr_straightlevel);
TF9 = strcmp('Pass',app16_esr_turnrate);
TF10 = strcmp('Pass',app17_esr_turnrate);
TF11 = strcmp('Pass',app18_esr_approach);
TF12 = strcmp('Pass',app_esr_rnp);
progressbar(1284/1305);
if (TF1 && TF2 && TF3 && TF4 && TF5 && TF6 && TF7 && TF8 && TF9 && TF10...
    && TF11 && TF12) == 1;
    app_esr = 'Pass';
else
    app_esr = 'Fail';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Final ESR Grade %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TF1 = strcmp('Pass',dep_esr);
TF2 = strcmp('Pass',enr_esr);
TF3 = strcmp('Pass',arv_esr);
TF4 = strcmp('Pass',app_esr);
if (TF1 && TF2 && TF3 && TF4) == 1;
    final_esr = 'Pass';
else
    final_esr = 'Fail';
end
fprintf('%s','ESR grading complete. ');
toc(ESR_time);
fprintf('%s','Code complete. ');
toc(code_start)
progressbar(1305/1305);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                               ESR Results
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

save = menu('Save the results?', 'Yes', 'No');
if save == 1;
    TF1 = 1;
else
    TF1 = 0;
    TFsavefig = 0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Print Results to File %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if TF1 == 1;
    TFsavefig = 1;

    [FileName, PathName] = uiputfile('*.dat', 'Save Results As', ...
        'ESRresults_');
    file_id = fullfile(PathName, FileName);
    if FileName == 0;
        FileName = 'ESRresults.dat';
    end

    aircraft_id = input('Enter the name of the aircraft being tested:
    ', ...
        's');
    weather_cond = input...
        ('Enter the weather conditions observed during the test: ', 's');

    fid = fopen(file_id, 'wt');
    fprintf(fid, '%s %s\n', 'Aircraft: ', aircraft_id);
    fprintf(fid, '%s %.0f %s %.0f %s %.0f\n', 'Test Date: ', month(1,1), ...
        '/', day(1,1), '/', year(1,1));
    fprintf(fid, '%s %s\n', 'Test Weather Conditions: ', weather_cond);
    fprintf(fid, '%s\n', ' ');
    fprintf(fid, '%s %s\n', 'Final ESR Grade: ', final_esr);
    fprintf(fid, '%s\n', ' ');
    fprintf(fid, '%s\t %s\n', 'Flight Phase', 'Pass/Fail');
    fprintf(fid, '%s\t %s\n', 'Departure', dep_esr);
    fprintf(fid, '%s\t %s\n', 'En Route', enr_esr);
    fprintf(fid, '%s\t\t %s\n', 'Arrival', arv_esr);
    fprintf(fid, '%s\t %s\n', 'Approach', app_esr);
    fprintf(fid, '%s\n', ' ');

    fprintf(fid, '%s\t %s\t\t\t %s\t %s\t\t %s\n', 'Flight Phase', 'Test', ...

```

```

        'Pass/Fail', 'Actual', 'ESR Requirement');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'Departure 2', ...
        'Climb', dep2_esr_climbrate, dep2_avg_climbrate, 'ft/min', ...
        '>=500 ft/min');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'Departure 5', ...
        'Descent', dep5_esr_descentrate, dep5_avg_climbrate, 'ft/min', ...
        '<=-500 ft/min');
fprintf(fid, '%s\t %s\t %s\n', 'Departure 8', 'Straight and Level', ...
        dep8_esr_straightlevel);
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n', ' ', ...
        'Altitude Deviation', dep8_esr_alt, dep8_max_alt_dev, 'ft', ...
        '<=100 ft');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
        'Course Deviation', dep8_esr_hdg, dep8_max_hdg_dev, 'deg', '<=10
deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
        'Airspeed Deviation', dep8_esr_TAS, dep8_max_TAS_dev, 'ft/sec', ...
        '<=16.9 ft/sec (10 knts)');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Departure 9', ...
        'Turn', dep9_esr_turnrate, dep9_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Departure 10', ...
        'Turn', dep10_esr_turnrate, dep10_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Departure 11', ...
        'Turn', dep11_esr_turnrate, dep11_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Departure 12', ...
        'Turn', dep12_esr_turnrate, dep12_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.3f %s\t %s\n', 'Departure', ...
        'RNP Level', dep_esr_rnp, dep_max_track_y, 'NM', '<=0.125 NM');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'En Route 1', ...
        'Climb', enr1_esr_climbrate, enr1_avg_climbrate, 'ft/min', ...
        '>=500 ft/min');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'En Route 4', ...
        'Descent', enr4_esr_descentrate, enr4_avg_climbrate, 'ft/min', ...
        '<=-500 ft/min');
fprintf(fid, '%s\t %s\t %s\n', 'En Route 7', 'Straight and Level', ...
        enr7_esr_straightlevel);
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n', ' ', ...

```

```

        'Altitude Deviation', enr7_esr_alt, enr7_max_alt_dev, 'ft', ...
        '<=100 ft');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
        'Course Deviation', enr7_esr_hdg, enr7_max_hdg_dev, 'deg', '<=10
deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
        'Airspeed Deviation', enr7_esr_TAS, enr7_max_TAS_dev, 'ft/sec', ...
        '<=16.9 ft/sec (10 knts)');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t\t %s\t\t %.3f %s\t %s\n', 'En Route', ...
        'RNP Level', enr_esr_rnp, enr_max_track_y, 'NM', '<=0.25 NM');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'Arrival 1', ...
        'Climb', arv1_esr_climbrate, arv1_avg_climbrate, 'ft/min', ...
        '>=500 ft/min');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.0f %s\t %s\n', 'Arrival 4', ...
        'Descent', arv4_esr_descentrate, arv4_avg_climbrate, 'ft/min', ...
        '<=-500 ft/min');
fprintf(fid, '%s\t %s\t %s\n', 'Arrival 7', 'Straight and Level', ...
        arv7_esr_straightlevel);
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n', ' ', ...
        'Altitude Deviation', arv7_esr_alt, arv7_max_alt_dev, 'ft', ...
        '<=100 ft');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
        'Course Deviation', arv7_esr_hdg, arv7_max_hdg_dev, 'deg', '<=10
deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t %s\n', ' ', ...
        'Airspeed Deviation', arv7_esr_TAS, arv7_max_TAS_dev, 'ft/sec', ...
        '<=16.9 ft/sec (10 knts)');
fprintf(fid, '%s\n', ' ');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Arrival 8', ...
        'Turn', arv8_esr_turnrate, arv8_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Arrival 9', ...
        'Turn', arv9_esr_turnrate, arv9_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Arrival 10', ...
        'Turn', arv10_esr_turnrate, arv10_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Arrival 11', ...
        'Turn', arv11_esr_turnrate, arv11_avg_turnrate, 'deg/sec', ...
        '>=3 deg/sec');
fprintf(fid, '%s\t\t %s\t\t %s\t\t %.3f %s\t %s\n', 'Arrival', ...
        'RNP Level', arv_esr_rnp, arv_max_track_y, 'NM', '<=0.25 NM');
fprintf(fid, '%s\n', ' ');

```



```

fprintf(fid, '%s\t %s\t\t %s\t\t %.0f %s\t %s\n', 'Approach 1', ...
    'Descent', app1_esr_descentrate, app1_avg_climbrate, 'ft/min', ...
    '<=-500 ft/min');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Approach 5', ...
    'Turn', app5_esr_turnrate, app5_avg_turnrate, 'deg/sec', ...
    '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Approach 6', ...
    'Turn', app6_esr_turnrate, app6_avg_turnrate, 'deg/sec', ...
    '>=3 deg/sec');
fprintf(fid, '%s\t %s\t\t %s\n', 'Approach 7', 'Final Approach', ...
    app7_esr_approach);
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n', ' ', ...
    'Altitude Deviation', app7_esr_alt, app7_max_alt_dev, 'ft', ...
    '<=100 ft');
fprintf(fid, '%s\t\t %s\t %s\t\t %.3f %s\t %s\n', ' ', ...
    'Glideslope Deviation', app7_esr_alt, app7_gs_dev_max, 'deg', ...
    '<=0.525 deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.3f %s\t %s\n', ' ', ...
    'Lateral Deviation', app7_esr_lat, app7_lat_dev_max, 'deg', ...
    '<=1.875 deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
    'Airspeed Deviation', app7_esr_TAS, app7_max_TAS_dev, 'ft/sec', ...
    '<=16.9 ft/sec (10 knts)');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t %s\n', 'Approach 8', 'Missed Approach', ...
    app8_esr_approach);
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n', ' ', ...
    'Altitude Deviation', app8_esr_alt, app8_max_alt_dev, 'ft', ...
    '<=100 ft');
fprintf(fid, '%s\t\t %s\t %s\t\t %.3f %s\t %s\n', ' ', ...
    'Climb Angle Deviation', app8_esr_alt, app8_gs_dev_max, 'deg', ...
    '<=0.525 deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.3f %s\t %s\n', ' ', ...
    'Lateral Deviation', app8_esr_lat, app8_lat_dev_max, 'deg', ...
    '<=1.875 deg');
fprintf(fid, '%s\t\t %s\t %s\t\t %.1f %s\t %s\n', ' ', ...
    'Airspeed Deviation', app8_esr_TAS, app8_max_TAS_dev, 'ft/sec', ...
    '<=16.9 ft/sec (10 knts)');
fprintf(fid, '%s\n', ' ');

fprintf(fid, '%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n', 'Approach 12', ...
    'Turn', app12_esr_turnrate, app12_avg_turnrate, 'deg/sec', ...

```

```

        '>=3 deg/sec');
fprintf(fid,'%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n','Approach 13',...
        'Turn',app13_esr_turnrate,app13_avg_turnrate,'deg/sec',...
        '>=3 deg/sec');
fprintf(fid,'%s\t %s\t %s\n','Approach 14','Straight and Level',...
        app14_esr_straightlevel);
fprintf(fid,'%s\n',' ');

fprintf(fid,'%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n',' ',...
        'Altitude Deviation',app14_esr_alt,app14_max_alt_dev,'ft',...
        '<=100 ft');
fprintf(fid,'%s\t\t %s\t %s\t\t %.1f %s\t %s\n',' ',...
        'Course Deviation',app14_esr_hdg,app14_max_hdg_dev,'deg',...
        '<=10 deg');
fprintf(fid,'%s\t\t %s\t %s\t\t %.1f %s\t %s\n',' ',...
        'Airspeed Deviation',app14_esr_TAS,app14_max_TAS_dev,'ft/sec',...
        '<=16.9 ft/sec (10 knts)');
fprintf(fid,'%s\n',' ');

fprintf(fid,'%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n','Approach 16',...
        'Turn',app16_esr_turnrate,app16_avg_turnrate,'deg/sec',...
        '>=3 deg/sec');
fprintf(fid,'%s\t %s\t\t\t %s\t\t %.1f %s\t %s\n','Approach 17',...
        'Turn',app17_esr_turnrate,app17_max_turnrate,'deg/sec',...
        '>=3 deg/sec');
fprintf(fid,'%s\t %s\t\t %s\n','Approach 18','Final Approach',...
        app18_esr_approach);
fprintf(fid,'%s\n',' ');

fprintf(fid,'%s\t\t %s\t %s\t\t %.0f %s\t\t %s\n',' ',...
        'Altitude Deviation',app18_esr_alt,app18_max_alt_dev,'ft',...
        '<=100 ft');
fprintf(fid,'%s\t\t %s\t %s\t\t %.3f %s\t %s\n',' ',...
        'Glideslope Deviation',app18_esr_alt,app18_gs_dev_max,'deg',...
        '<=0.525 deg');
fprintf(fid,'%s\t\t %s\t %s\t\t %.3f %s\t %s\n',' ',...
        'Lateral Deviation',app18_esr_lat,app18_lat_dev_max,'deg',...
        '<=1.875 deg');
fprintf(fid,'%s\t\t %s\t %s\t\t %.1f %s\t %s\n',' ',...
        'Airspeed Deviation',app18_esr_TAS,app18_max_TAS_dev,'ft/sec',...
        '<=16.9 ft/sec (10 knts)');
fprintf(fid,'%s\n',' ');

fprintf(fid,'%s\t %s\t\t %s\t\t %.3f %s\t %s\n','Approach',...
        'RNP Level',app_esr_rnp,app_max_track_y,'NM','<=0.125 NM');
fclose(fid);
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plot Results %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot_input = menu('Plot results?', 'Yes', 'No');
if plot_input == 1;
    TFq = 0;
else
    TFq = 1;
end
while TFq == 0;
user_entry = menu('Select flight phase to plot.', 'Departure', 'En
route', ...
    'Arrival', 'Approach', 'All');
if user_entry == 1;
    TF1 = 1;
else
    TF1 = 0;
end
if user_entry == 2;
    TF2 = 1;
else
    TF2 = 0;
end
if user_entry == 3;
    TF3 = 1;
else
    TF3 = 0;
end
if user_entry == 4;
    TF4 = 1;
else
    TF4 = 0;
end
if user_entry == 5;
    TF5 = 1;
else
    TF5 = 0;
end
if user_entry == 0;
    TFq = 1;
end
if TF5 == 1;
    TF1 = 1;
    TF2 = 1;
    TF3 = 1;
    TF4 = 1;
end

set(0, 'DefaultFigureUnits', 'inches');
set(0, 'DefaultFigurePosition', [1 1 6.5 4.87]);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Departure Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if TF1 == 1;

    % Departure Phase 2 Climb Rate Test
    figure;
    plot(clock_sec,Alt);
    hold on;
    plot(clock_sec,AltTarget,':g','LineWidth',1.5);
    plot(dep2_esr_vrateline(:,1),dep2_esr_vrateline(:,2),'--r',...
        'LineWidth',1.5);
    hold off;
    set(gca,'XGrid','on');
    set(gca,'YGrid','on');
    set(gca,'FontName','Times New Roman');
    set(gca,'FontSize',12);
    xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
    ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
    title('Departure Phase 2: Climb Rate Test','FontName',...
        'Times New Roman','FontSize',14);
    legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
    set(legend1,'Location','SouthEast','FontName','Times New Roman',...
        'FontSize',10);
    xlim([clock_sec(dep2_start_i,1) clock_sec(dep2_end_i,1)]);
    if TFsavefig == 1;
        print -f -dmeta 'Figures\Departure Phase 2 Climb Rate Test ALT'
    end

    % Departure Phase 2 Climb Rate Test
    figure;
    plot(dep2_climbrate(:,1),dep2_climbrate(:,2));
    hold on;
    plot(dep2_climbrate(:,1),dep2_climbrate(:,3),':g','LineWidth',1.5);
    line([clock_sec(dep2_start_i,1) clock_sec(dep2_end_i,1)],[500 500],...
        'LineStyle','--','Color','r','LineWidth',1.5);
    hold off;
    set(gca,'XGrid','on');
    set(gca,'YGrid','on');
    set(gca,'FontName','Times New Roman');
    set(gca,'FontSize',12);
    xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
    ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
        'FontSize',12);
    title('Departure Phase 2: Climb Rate Test','FontName',...
        'Times New Roman','FontSize',14);
    legend1 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...
        'ESR Req');
    set(legend1,'Location','SouthEast','FontName','Times New Roman',...
        'FontSize',10);
    xlim([clock_sec(dep2_start_i,1) clock_sec(dep2_end_i,1)]);

```

```

ylim([490 900]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Departure Phase 2 Climb Rate Test Vertical Rate'
end

% Departure Phase 5 Descent Rate Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
plot(dep5_esr_vrateline(:,1),dep5_esr_vrateline(:,2),'--r',...
    'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Departure Phase 5: Descent Rate Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(dep5_start_i,1) clock_sec(dep5_end_i,1)]);
ylim([990 1510]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Departure Phase 5 Descent Rate Test ALT'
end

% Departure Phase 5 Descent Rate Test
figure;
plot(dep5_climbrate(:,1),dep5_climbrate(:,2));
hold on;
plot(dep5_climbrate(:,1),dep5_climbrate(:,3),':g','LineWidth',1.5);
line([clock_sec(dep5_start_i,1) clock_sec(dep5_end_i,1)],...
    [-500 -500],'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
    'FontSize',12);
title('Departure Phase 5: Descent Rate Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...

```

```

        'ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(dep5_start_i,1) clock_sec(dep5_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Departure Phase 5 Descent Rate Test Vertical Rate'
end

% Departure Phase 8 Straight and Level Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(dep8_start_i:dep8_end_i,1),AltTarget(dep8_start_i:...
    dep8_end_i,1)+100,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(dep8_start_i:dep8_end_i,1),AltTarget(dep8_start_i:...
    dep8_end_i,1)-100,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Departure Phase 8: Straight and Level Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)]);
ylim([890 1110]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Departure Phase 8 Straight and Level Test ALT'
end

% Departure Phase 8 Straight and Level Test
figure;
plot(clock_sec,Direction);
line([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)],...
    [dep8_CourseTarget dep8_CourseTarget],'LineStyle',...
    ':','Color','g','LineWidth',1.5);
line([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)],...
    [dep8_CourseTarget+10 dep8_CourseTarget+10],...
    'LineStyle','--','Color','r','LineWidth',1.5);
line([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)],...
    [dep8_CourseTarget-10 dep8_CourseTarget-10],...
    'LineStyle','--','Color','r','LineWidth',1.5);

```

```

set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Course (deg)','FontName','Times New Roman','FontSize',12);
title('Departure Phase 8: Straight and Level Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Course','Commanded Course','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)]);
ylim([19 42]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Departure Phase 8 Straight and Level Test Course'
end

% Departure Phase 8 Straight and Level Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget,':g','LineWidth',1.5);
line(clock_sec(dep8_start_i:dep8_end_i,1),TASTarget(dep8_start_i:...
    dep8_end_i,1)+16.9,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(dep8_start_i:dep8_end_i,1),TASTarget(dep8_start_i:...
    dep8_end_i,1)-16.9,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
title('Departure Phase 8: Straight and Level Test','FontName',...
      'Times New Roman','FontSize',14);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Airspeed (ft/sec)','FontName','Times New
Roman','FontSize',12);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(dep8_start_i,1) clock_sec(dep8_end_i,1)]);
ylim([34 70]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Departure Phase 8 Straight and Level Test TAS'
end

% Departure Phase 9 - 12 Turn Rate Test
figure;

```

```

p1 = plot(dep9_turnrate(:,1),dep9_turnrate(:,2));
hold on;
p2 = plot(dep10_turnrate(:,1),dep10_turnrate(:,2));
p3 = plot(dep11_turnrate(:,1),dep11_turnrate(:,2));
plot(dep12_turnrate(:,1),dep12_turnrate(:,2));
hold off;
l1 = line([dep9_turnrate(1,1) dep9_turnrate(size(dep9_turnrate,1),...
1)], [dep9_avg_turnrate dep9_avg_turnrate], 'LineStyle', ':', ...
'Color', 'g', 'LineWidth', 1.5);
l2 = line([dep10_turnrate(1,1) dep10_turnrate(size(dep10_turnrate,...
1),1)], [dep10_avg_turnrate dep10_avg_turnrate], 'LineStyle', ':', ...
'Color', 'g', 'LineWidth', 1.5);
l3 = line([dep11_turnrate(1,1)
dep11_turnrate(size(dep11_turnrate,1),...
1)], [dep11_avg_turnrate dep11_avg_turnrate], 'LineStyle', ':', ...
'Color', 'g', 'LineWidth', 1.5);
line([dep12_turnrate(1,1)
dep12_turnrate(size(dep12_turnrate,1),1)], ...
[dep12_avg_turnrate
dep12_avg_turnrate], 'LineStyle', ':', 'Color', ...
'g', 'LineWidth', 1.5);
line([clock_sec(dep9_start_i,1) clock_sec(dep12_end_i,1)], [3 3], ...
'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);

set(get(get(p1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');

set(get(get(p2, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');

set(get(get(p3, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');

set(get(get(l1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');

set(get(get(l2, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');

set(get(get(l3, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
, 'off');
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Turn Rate (deg/sec)', 'FontName', 'Times New
Roman', 'FontSize'...
, 12);

```



```

title('Departure Phase 9 - 12: Turn Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Turn Rate','Avg Turn Rate','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(dep9_start_i,1) dep12_turnrate(size(dep12_turnrate,...
      1),1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Departure Phase 9 12 Turn Rate Test'
end

% Departure Phase RNP Test
figure;
plot(dep_track_y(:,2),dep_track_y(:,1));
line([clock_sec(dep2_start_i,1) clock_sec(dep12_end_i,1)],...
      [dep_max_track_y dep_max_track_y],'LineStyle',':','Color','g',...
      'LineWidth',1.5);
line([clock_sec(dep2_start_i,1) clock_sec(dep12_end_i,1)],...
      [0.125 0.125],'LineStyle','--','Color','r','LineWidth',1.5);
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Lateral Deviation (NM)','FontName','Times New Roman',...
      'FontSize',12);
title('Departure Phase: Lateral Deviation Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Lateral Deviation','95% Containment',...
      'ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(dep2_start_i,1) clock_sec(dep12_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Departure Phase Lateral Deviation Test'
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% En Route Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if TF2 == 1;

    % En Route Phase 1 Climb Rate Test
    figure;
    plot(clock_sec,Alt);
    hold on;
    plot(clock_sec,AltTarget,':g','LineWidth',1.5);
    plot(enr1_esr_vrateline(:,1),enr1_esr_vrateline(:,2),'--r',...
          'LineWidth',1.5);
    hold off;

```

```

set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('En Route Phase 1: Climb Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(enr1_start_i,1) clock_sec(enr1_end_i,1)]);
ylim([950 3050]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\En Route Phase 1 Climb Rate Test ALT'
end

% En Route Phase 1 Climb Rate Test
figure;
plot(enr1_climbrate(:,1),enr1_climbrate(:,2));
hold on;
plot(enr1_climbrate(:,1),enr1_climbrate(:,3),'g','LineWidth',1.5);
line([clock_sec(enr1_start_i,1) clock_sec(enr1_end_i,1)],[500 500],...
      'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
      'FontSize',12);
title('En Route Phase 1: Climb Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend2 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...
      'ESR Req');
set(legend2,'Location','NorthWest','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(enr1_start_i,1) clock_sec(enr1_end_i,1)]);
ylim([490 1300]);
if TFsavefig == 1;
    print -f -dmeta...
        'Figures\En Route Phase 1 Climb Rate Test Vertical Rate'
end

% En Route Phase 4 Descent Rate Test
figure;
plot(clock_sec,Alt);
hold on;

```

```

plot(clock_sec,AltTarget,':g','LineWidth',1.5);
plot(enr4_esr_vrateline(:,1),enr4_esr_vrateline(:,2),'--r',...
     'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('En Route Phase 4: Descent Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(enr4_start_i,1) clock_sec(enr4_end_i,1)]);
ylim([2490 3010]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\En Route Phase 4 Descent Rate Test ALT'
end

% En Route Phase 4 Descent Rate Test
figure;
plot(enr4_climbrate(:,1),enr4_climbrate(:,2));
hold on;
plot(enr4_climbrate(:,1),enr4_climbrate(:,3),':g','LineWidth',1.5);
line([clock_sec(enr4_start_i,1) clock_sec(enr4_end_i,1)],...
     [-500 -500],'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
      'FontSize',12);
title('En Route Phase 4: Descent Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend2 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...
      'ESR Req');
set(legend2,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(enr4_start_i,1) clock_sec(enr4_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\En Route Phase 4 Descent Rate Test Vertical Rate'
end

% En Route Phase 7 Straight and Level Test

```

```

figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(enr7_start_i:enr7_end_i,1),AltTarget(enr7_start_i:...
    enr7_end_i,1)+100,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(enr7_start_i:enr7_end_i,1),AltTarget(enr7_start_i:...
    enr7_end_i,1)-100,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('En Route Phase 7: Straight and Level Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)]);
ylim([2390 2610]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\En Route Phase 7 Straight and Level Test ALT'
end

% En Route Phase 7 Straight and Level Test
figure;
plot(clock_sec,Direction);
line([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)],...
    [enr7_CourseTarget enr7_CourseTarget],'LineStyle',...
    ':','Color','g','LineWidth',1.5);
line([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)],...
    [enr7_CourseTarget+10 enr7_CourseTarget+10],...
    'LineStyle','--','Color','r','LineWidth',1.5);
line([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)],...
    [enr7_CourseTarget-10 enr7_CourseTarget-10],...
    'LineStyle','--','Color','r','LineWidth',1.5);
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Course (deg)','FontName','Times New Roman','FontSize',12);
title('En Route Phase 7: Straight and Level Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Course','Commanded Course','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...

```

```

        'FontSize',10);
xlim([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)]);
ylim([276 300]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\En Route Phase 7 Straight and Level Test Course'
end

% En Route Phase 7 Straight and Level Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget,':g','LineWidth',1.5);
line(clock_sec(enr7_start_i:enr7_end_i,1),TASTarget(enr7_start_i:...
    enr7_end_i,1)+16.9,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(enr7_start_i:enr7_end_i,1),TASTarget(enr7_start_i:...
    enr7_end_i,1)-16.9,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Airspeed (ft/sec)','FontName','Times New
Roman','FontSize',12);
title('En Route Phase 7: Straight and Level Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(enr7_start_i,1) clock_sec(enr7_end_i,1)]);
ylim([35 71]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\En Route Phase 7 Straight and Level Test TAS'
end

% En Route Phase RNP Rate Test
figure;
plot(enr_track_y(:,2),enr_track_y(:,1));
line([clock_sec(enr1_start_i,1) clock_sec(enr11_end_i,1)],...
    [enr_max_track_y enr_max_track_y],'LineStyle',':', 'Color','g',...
    'LineWidth',1.5);
line([clock_sec(enr1_start_i,1) clock_sec(enr11_end_i,1)],...
    [0.25 0.25],'LineStyle','--','Color','r','LineWidth',1.5);
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);

```

```

xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Lateral Deviation (NM)', 'FontName', 'Times New Roman', ...
    'FontSize', 12);
title('En Route Phase: Lateral Deviation Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Lateral Deviation', '95% Containment', ...
    'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman', ...
    'FontSize', 10);
xlim([clock_sec(enr1_start_i,1) clock_sec(enr11_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\En Route Phase Lateral Deviation Test'
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Arrival Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if TF3 == 1;

    % Arrival Phase 1 Climb Rate Test
    figure;
    plot(clock_sec,Alt);
    hold on;
    plot(clock_sec,AltTarget, ':g', 'LineWidth', 1.5);
    plot(arv1_esr_vrateline(:,1), arv1_esr_vrateline(:,2), '--r', ...
        'LineWidth', 1.5);
    hold off;
    set(gca, 'XGrid', 'on');
    set(gca, 'YGrid', 'on');
    set(gca, 'FontName', 'Times New Roman');
    set(gca, 'FontSize', 12);
    xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
    ylabel('Altitude (ft)', 'FontName', 'Times New Roman', 'FontSize', 12);
    title('Arrival Phase 1: Climb Rate Test', 'FontName', ...
        'Times New Roman', 'FontSize', 14);
    legend1 = legend('Aircraft Alt', 'Commanded Alt', 'ESR Req');
    set(legend1, 'Location', 'SouthEast', 'FontName', 'Times New Roman', ...
        'FontSize', 10);
    xlim([clock_sec(arv1_start_i,1) clock_sec(arv1_end_i,1)]);
    ylim([2490 3010]);
    if TFsavefig == 1;
        print -f -dmeta 'Figures\Arrival Phase 1 Climb Rate Test ALT'
    end

    % Arrival Phase 1 Climb Rate Test
    figure;
    plot(arv1_climbrate(:,1), arv1_climbrate(:,2));
    hold on;
    plot(arv1_climbrate(:,1), arv1_climbrate(:,3), ':g', 'LineWidth', 1.5);
    line([clock_sec(arv1_start_i,1) clock_sec(arv1_end_i,1)], [500 500], ...

```

```

        'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
    'FontSize',12);
title('Arrival Phase 1: Climb Rate Test','FontName',...
    'Times New Roman','FontSize',14);
legend2 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...
    'ESR Req');
set(legend2,'Location','SouthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(arv1_start_i,1) clock_sec(arv1_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Arrival Phase 1 Climb Rate Test Vertical Rate'
end

% Arrival Phase 4 Descent Rate Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
plot(arv4_esr_vrateline(:,1),arv4_esr_vrateline(:,2),'--r',...
    'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Arrival Phase 4: Descent Rate Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(arv4_start_i,1) clock_sec(arv4_end_i,1)]);
ylim([990 3010]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Arrival Phase 4 Descent Rate Test ALT'
end

% Arrival Phase 4 Descent Rate Test
figure;
plot(arv4_climbrate(:,1),arv4_climbrate(:,2));

```

```

hold on;
plot(arv4_climbrate(:,1),arv4_climbrate(:,3),':g','LineWidth',1.5);
line([clock_sec(arv4_start_i,1) clock_sec(arv4_end_i,1)],...
     [-500 -500],'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Vertical Rate (ft/min)','FontName','Times New Roman',...
       'FontSize',12);
title('Arrival Phase 4: Descent Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend2 = legend('Aircraft Vertical Rate','Avg Vertical Rate',...
                 'ESR Req');
set(legend2,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(arv4_start_i,1) clock_sec(arv4_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Arrival Phase 4 Descent Rate Test Vertical Rate'
end

% Arrival Phase 7 Straight and Level Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(arv7_start_i:arv7_end_i,1),AltTarget(arv7_start_i:...
    arv7_end_i,1)+100,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(arv7_start_i:arv7_end_i,1),AltTarget(arv7_start_i:...
    arv7_end_i,1)-100,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Arrival Phase 7: Straight and Level Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)]);
ylim([890 1110]);
if TFsavefig == 1;
    print -f -dmeta ...

```



```

        'Figures\Arrival Phase 7 Straight and Level Test ALT'
end

% Arrival Phase 7 Straight and Level Test
figure;
plot(clock_sec,Direction);
hold on;
line([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)],...
     [arv7_CourseTarget arv7_CourseTarget], 'LineStyle',...
     ':', 'Color', 'g', 'LineWidth',1.5);
line([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)],...
     [arv7_CourseTarget+10 arv7_CourseTarget+10],...
     'LineStyle','--', 'Color', 'r', 'LineWidth',1.5);
line([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)],...
     [arv7_CourseTarget-10 arv7_CourseTarget-10],...
     'LineStyle','--', 'Color', 'r', 'LineWidth',1.5);
hold off;
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize',12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize',12);
ylabel('Course (deg)', 'FontName', 'Times New Roman', 'FontSize',12);
title('Arrival Phase 7: Straight and Level Test', 'FontName',...
      'Times New Roman', 'FontSize',14);
legend1 = legend('Aircraft Course', 'Commanded Course', 'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman',...
      'FontSize',10);
xlim([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)]);
ylim([219 241]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Arrival Phase 7 Straight and Level Test Course'
end

% Arrival Phase 7 Straight and Level Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget, ':g', 'LineWidth',1.5);
line(clock_sec(arv7_start_i:arv7_end_i,1),TASTarget(arv7_start_i:...
     arv7_end_i,1)+16.9, 'LineStyle', '--', 'Color', 'r', 'LineWidth',1.5);
line(clock_sec(arv7_start_i:arv7_end_i,1),TASTarget(arv7_start_i:...
     arv7_end_i,1)-16.9, 'LineStyle', '--', 'Color', 'r', 'LineWidth',1.5);
hold off;
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize',12);

```

```

xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Airspeed (ft/sec)', 'FontName', 'Times New
Roman', 'FontSize', 12);
title('Arrival Phase 7: Straight and Level Test', 'FontName', ...
      'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft TAS', 'Commanded TAS', 'ESR Req');
set(legend1, 'Location', 'SouthEast', 'FontName', 'Times New Roman', ...
      'FontSize', 10);
xlim([clock_sec(arv7_start_i,1) clock_sec(arv7_end_i,1)]);
ylim([34 70]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Arrival Phase 7 Straight and Level Test TAS'
end

% Arrival Phase 8 - 11 Turn Rate Test
figure;
p1 = plot(arv8_turnrate(:,1), arv8_turnrate(:,2));
hold on;
p2 = plot(arv9_turnrate(:,1), arv9_turnrate(:,2));
p3 = plot(arv10_turnrate(:,1), arv10_turnrate(:,2));
plot(arv11_turnrate(:,1), arv11_turnrate(:,2));
hold off;
l1 = line([arv8_turnrate(1,1) arv8_turnrate(size(arv8_turnrate,1)...
    ,1)], [arv8_avg_turnrate arv8_avg_turnrate], 'LineStyle', ':', ...
    'Color', 'g', 'LineWidth', 1.5);
l2 = line([arv9_turnrate(1,1) arv9_turnrate(size(arv9_turnrate,1)...
    ,1)], [arv9_avg_turnrate arv9_avg_turnrate], 'LineStyle', ':', ...
    'Color', 'g', 'LineWidth', 1.5);
l3 = line([arv10_turnrate(1,1)
arv10_turnrate(size(arv10_turnrate,1)...
    ,1)], [arv10_avg_turnrate arv10_avg_turnrate], 'LineStyle', ':', ...
    'Color', 'g', 'LineWidth', 1.5);
line([arv11_turnrate(1,1)
arv11_turnrate(size(arv11_turnrate,1),1)], ...
    [arv11_avg_turnrate
arv11_avg_turnrate], 'LineStyle', ':', 'Color', ...
    'g', 'LineWidth', 1.5);
line([clock_sec(arv8_start_i,1) clock_sec(arv11_end_i,1)], [3 3], ...
    'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);

set(get(get(p1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');

set(get(get(p2, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...

```

```

        , 'off');

set(get(get(p3, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');

set(get(get(l1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');

set(get(get(l2, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');

set(get(get(l3, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Turn Rate (deg/sec)', 'FontName', 'Times New
Roman', 'FontSize'...
    , 12);
title('Arrival Phase 8 - 11: Turn Rate Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Turn Rate', 'Avg Turn Rate', 'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman', ...
    'FontSize', 10);
xlim([clock_sec(arv8_start_i, 1) arv11_turnrate(size(arv11_turnrate...
    , 1), 1)]);
ylim([0 30]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Arrival Phase 8 11 Turn Rate Test'
end

% Arrival Phase RNP Rate Test
figure;
plot(arv_track_y(:, 2), arv_track_y(:, 1));
line([clock_sec(arv1_start_i, 1) clock_sec(arv11_end_i, 1)], ...
    [arv_max_track_y arv_max_track_y], 'LineStyle', ':', 'Color', ...
    'g', 'LineWidth', 1.5);
line([clock_sec(arv1_start_i, 1) clock_sec(arv11_end_i, 1)], ...
    [0.25 0.25], 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Lateral Deviation (NM)', 'FontName', 'Times New Roman', ...
    'FontSize', 12);
title('Arrival Phase: Lateral Deviation Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Lateral Deviation', '95% Containment', ...
    'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman', ...

```

```

        'FontSize',10);
xlim([clock_sec(arv1_start_i,1) clock_sec(arv11_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Arrival Phase Lateral Deviation Test'
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Approach Phase %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if TF4 == 1;

    % Approach Phase 1 Descent Rate Test
    figure;
    plot(clock_sec,Alt);
    hold on;
    plot(clock_sec,AltTarget,':g','LineWidth',1.5);
    plot(app1_esr_vrateline(:,1),app1_esr_vrateline(:,2),'--r',...
        'LineWidth',1.5);
    hold off;
    set(gca,'XGrid','on');
    set(gca,'YGrid','on');
    set(gca,'FontName','Times New Roman');
    set(gca,'FontSize',12);
    xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
    ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
    title('Approach Phase 1: Descent Rate Test','FontName',...
        'Times New Roman','FontSize',14);
    legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
    set(legend1,'Location','NorthEast','FontName','Times New Roman',...
        'FontSize',10);
    xlim([clock_sec(app1_start_i,1) clock_sec(app1_end_i,1)]);
    ylim([480 1010]);
    if TFsavefig == 1;
        print -f -dmeta 'Figures\Approach Phase 1 Descent Rate Test ALT'
    end

    % Approach Phase 1 Descent Rate Test
    figure;
    plot(app1_climbrate(:,1),app1_climbrate(:,2));
    hold on;
    plot(app1_climbrate(:,1),app1_climbrate(:,3),':g','LineWidth',1.5);
    line([clock_sec(app1_start_i,1) clock_sec(app1_end_i,1)],[-500 -
500]...
        , 'LineStyle','--', 'Color','r', 'LineWidth',1.5);
    hold off;
    set(gca,'XGrid','on');
    set(gca,'YGrid','on');
    set(gca,'FontName','Times New Roman');
    set(gca,'FontSize',12);
    xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);

```

```

ylabel('Vertical Rate (ft/min)', 'FontName', 'Times New Roman', ...
      'FontSize', 12);
title('Approach Phase 1: Descent Rate Test', 'FontName', ...
      'Times New Roman', 'FontSize', 14);
legend2 = legend('Aircraft Vertical Rate', 'Avg Vertical Rate', ...
      'ESR Req');
set(legend2, 'Location', 'SouthEast', 'FontName', 'Times New Roman', ...
      'FontSize', 10);
xlim([clock_sec(app1_start_i,1) clock_sec(app1_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Approach Phase 1 Descent Rate Test Vertical Rate'
end

% Approach Phase 5 - 6 Turn Rate Test
figure;
p1 = plot(app5_turnrate(:,1), app5_turnrate(:,2));
hold on;
plot(app6_turnrate(:,1), app6_turnrate(:,2));
hold off;
l1 = line([app5_turnrate(1,1) app5_turnrate(size(app5_turnrate,1) ...
      ,1)], [app5_avg_turnrate app5_avg_turnrate], 'LineStyle', ':', ...
      'Color', 'g', 'LineWidth', 1.5);
line([app6_turnrate(1,1) app6_turnrate(size(app6_turnrate,1),1)], ...
      [app6_avg_turnrate app6_avg_turnrate], 'LineStyle', ':', 'Color', ...
      'g', 'LineWidth', 1.5);
line([clock_sec(app5_start_i,1) clock_sec(app6_end_i,1)], [3 3], ...
      'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);

set(get(get(p1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
      , 'off');

set(get(get(l1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
      , 'off');
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Turn Rate (deg/sec)', 'FontName', 'Times New
Roman', 'FontSize'...
      , 12);
title('Approach Phase 5 - 6: Turn Rate Test', 'FontName', ...
      'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Turn Rate', 'Avg Turn Rate', 'ESR Req');
set(legend1, 'Location', 'NorthWest', 'FontName', 'Times New Roman', ...
      'FontSize', 10);
xlim([clock_sec(app5_start_i,1) app6_turnrate(size(app6_turnrate, ...
      1),1)]);

```

```

if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 5 6 Turn Rate Test'
end

% Approach Phase 7 Final Approach Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(app7_start_i:app7_end_i,1),app7_esr_altline...
      (app7_start_i:app7_end_i,1),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
line(clock_sec(app7_start_i:app7_end_i,1),app7_esr_altline...
      (app7_start_i:app7_end_i,2),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 7: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app7_start_i,1) clock_sec(app7_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 7 Approach Test ALT'
end

% Approach Phase 7 Final Approach Test
figure;
plot(clock_sec,Track_Y);
hold on;
line(clock_sec(app7_start_i:app7_end_i,1),app7_esr_latline...
      (app7_start_i:app7_end_i,1),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
line(clock_sec(app7_start_i:app7_end_i,1),app7_esr_latline...
      (app7_start_i:app7_end_i,2),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
ylabel('Lateral Deviation (ft)','FontName','Times New Roman',...
      'FontSize',12);

```

```

xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 7: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Lateral Deviation','ESR Req');
set(legend1,'Location','NorthEast','FontName',...
      'Times New Roman','FontSize',10);
xlim([clock_sec(app7_start_i,1) clock_sec(app7_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 7 Approach Test Lateral'
end

% Approach Phase 7 Final Approach Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget,':g','LineWidth',1.5);
line(clock_sec(app7_start_i:app7_end_i,1),TASTarget(app7_start_i:...
    app7_end_i,1)+16.9,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(app7_start_i:app7_end_i,1),TASTarget(app7_start_i:...
    app7_end_i,1)-16.9,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Airspeed (ft/sec)','FontName','Times New
Roman','FontSize',12);
title('Approach Phase 7: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName',...
      'Times New Roman','FontSize',10);
xlim([clock_sec(app7_start_i,1) clock_sec(app7_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 7 Approach Test TAS'
end

% Approach Phase 8 Missed Approach Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(app8_start_i:app8_end_i,1),app8_esr_altline...
    (app8_start_i:app8_end_i,1),'LineStyle','--','Color','r',...
    'LineWidth',1.5);
line(clock_sec(app8_start_i:app8_end_i,1),app8_esr_altline...
    (app8_start_i:app8_end_i,2),'LineStyle','--','Color','r',...
    'LineWidth',1.5);

```

```

hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 8: Missed Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app8_start_i,1) clock_sec(app8_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 8 Missed Approach Test
ALT'
end

% Approach Phase 8 Missed Approach Test
figure;
plot(clock_sec,Track_Y);
hold on;
line(clock_sec(app8_start_i:app8_end_i,1),app8_esr_latline...
      (app8_start_i:app8_end_i,1),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
line(clock_sec(app8_start_i:app8_end_i,1),app8_esr_latline...
      (app8_start_i:app8_end_i,2),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
ylabel('Lateral Deviation (ft)','FontName','Times New Roman',...
      'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 8: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Lateral Deviation','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app8_start_i,1) clock_sec(app8_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 8 Approach Test Lateral'
end

% Approach Phase 8 Missed Approach Test
figure;
plot(clock_sec,TAS);

```



```

hold on;
plot(clock_sec,TASTarget,':g','LineWidth',1.5);
line(clock_sec(app8_start_i:app8_end_i,1),TASTarget(app8_start_i:...
    app8_end_i,1)+16.9,'LineStyle','--','Color','r','LineWidth',1.5);
line(clock_sec(app8_start_i:app8_end_i,1),TASTarget(app8_start_i:...
    app8_end_i,1)-16.9,'LineStyle','--','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Airspeed (ft/sec)','FontName','Times New
Roman','FontSize',12);
title('Approach Phase 8: Missed Approach Test','FontName',...
    'Times New Roman','FontSize',14);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
    'FontSize',10);
xlim([clock_sec(app8_start_i,1) clock_sec(app8_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 8 Missed Approach Test
TAS'
end

% Approach Phase 12 - 13 Turn Rate Test
figure;
p1 = plot(app12_turnrate(:,1),app12_turnrate(:,2));
hold on;
plot(app13_turnrate(:,1),app13_turnrate(:,2));
hold off;
l1 = line([app12_turnrate(1,1)
app12_turnrate(size(app12_turnrate,1)...
    ,1)], [app12_avg_turnrate app12_avg_turnrate], 'LineStyle', ':', ...
    'Color', 'g', 'LineWidth', 1.5);
line([app13_turnrate(1,1)
app13_turnrate(size(app13_turnrate,1),1)], ...
    [app13_avg_turnrate
app13_avg_turnrate], 'LineStyle', ':', 'Color', ...
    'g', 'LineWidth', 1.5);
line([clock_sec(app12_start_i,1) clock_sec(app13_end_i,1)], [3 3], ...
    'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);

set(get(get(p1,'Annotation'),'LegendInformation'),'IconDisplayStyle'...
    , 'off');

```

```

set(get(get(l1,'Annotation'),'LegendInformation'),'IconDisplayStyle'...
    , 'off');
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Turn Rate (deg/sec)', 'FontName', 'Times New
Roman', 'FontSize'...
    , 12);
title('Approach Phase 12 - 13: Turn Rate Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Turn Rate', 'Avg Turn Rate', 'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman', ...
    'FontSize', 10);
xlim([clock_sec(app12_start_i,1)
app13_turnrate(size(app13_turnrate,...
    1),1)]);
ylim([0 25]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 12 13 Turn Rate Test'
end

% Approach Phase 14 Straight and Level Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget, ':g', 'LineWidth', 1.5);

line(clock_sec(app14_start_i:app14_end_i,1),AltTarget(app14_start_i:...
    app14_end_i,1)+100, 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);

line(clock_sec(app14_start_i:app14_end_i,1),AltTarget(app14_start_i:...
    app14_end_i,1)-100, 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
hold off;
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Altitude (ft)', 'FontName', 'Times New Roman', 'FontSize', 12);
title('Approach Phase 14: Straight and Level Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Alt', 'Commanded Alt', 'ESR Req');
set(legend1, 'Location', 'SouthEast', 'FontName', 'Times New Roman', ...
    'FontSize', 10);
xlim([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Approach Phase 14 Straight and Level Test ALT'
end

```

```

% Approach Phase 14 Straight and Level Test
figure;
plot(clock_sec,Direction);
line([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)],...
      [app14_CourseTarget app14_CourseTarget], 'LineStyle',...
      ':', 'Color', 'g', 'LineWidth', 1.5);
line([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)],...
      [app14_CourseTarget+10 app14_CourseTarget+10],...
      'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
line([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)],...
      [app14_CourseTarget-10 app14_CourseTarget-10],...
      'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Course (deg)', 'FontName', 'Times New Roman', 'FontSize', 12);
title('Approach Phase 14: Straight and Level Test', 'FontName',...
      'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Course', 'Commanded Course', 'ESR Req');
set(legend1, 'Location', 'SouthEast', 'FontName', 'Times New Roman',...
      'FontSize', 10);
xlim([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)]);
ylim([287 309]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Approach Phase 14 Straight and Level Test Course'
end

% Approach Phase 14 Straight and Level Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget, ':g', 'LineWidth', 1.5);

line(clock_sec(app14_start_i:app14_end_i,1),TASTarget(app14_start_i:...
      app14_end_i,1)+16.9, 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);

line(clock_sec(app14_start_i:app14_end_i,1),TASTarget(app14_start_i:...
      app14_end_i,1)-16.9, 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
hold off;
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Airspeed (ft/sec)', 'FontName', 'Times New
Roman', 'FontSize', 12);

```

```

title('Approach Phase 14: Straight and Level Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app14_start_i,1) clock_sec(app14_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta ...
        'Figures\Approach Phase 14 Straight and Level Test TAS'
end

% Approach Phase 16 - 17 Turn Rate Test
figure;
p1 = plot(app16_turnrate(:,1),app16_turnrate(:,2));
hold on;
plot(app17_turnrate(:,1),app17_turnrate(:,2));
hold off;
l1 = line([app16_turnrate(1,1)
app16_turnrate(size(app16_turnrate,1)...
    ,1)], [app16_avg_turnrate app16_avg_turnrate], 'LineStyle',':',...
    'Color','g','LineWidth',1.5);
line([app17_turnrate(1,1)
app17_turnrate(size(app17_turnrate,1),1)],...
    [app17_avg_turnrate
app17_avg_turnrate], 'LineStyle',':', 'Color',...
    'g', 'LineWidth',1.5);
line([clock_sec(app16_start_i,1) clock_sec(app17_end_i,1)], [3 3],...
    'LineStyle','--', 'Color','r', 'LineWidth',1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);

set(get(get(p1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');

set(get(get(l1, 'Annotation'), 'LegendInformation'), 'IconDisplayStyle'...
    , 'off');
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Turn Rate (deg/sec)', 'FontName', 'Times New
Roman', 'FontSize'...
    , 12);
title('Approach Phase 16 - 17: Turn Rate Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Turn Rate','Avg Turn Rate','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app16_start_i,1)
app17_turnrate(size(app17_turnrate,...

```

```

        1),1)]];
ylim([0 25]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 16 17 Turn Rate Test'
end

% Approach Phase 18 Final Approach Test
figure;
plot(clock_sec,Alt);
hold on;
plot(clock_sec,AltTarget,':g','LineWidth',1.5);
line(clock_sec(app18_start_i:app18_end_i,1),app18_esr_altline...
      (app18_start_i:app18_end_i,1),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
line(clock_sec(app18_start_i:app18_end_i,1),app18_esr_altline...
      (app18_start_i:app18_end_i,2),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Altitude (ft)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 18: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Alt','Commanded Alt','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app18_start_i,1) clock_sec(app18_end_i-5,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 18 Approach Test ALT'
end

% Approach Phase 18 Final Approach Test
figure;
plot(clock_sec,Track_Y);
hold on;
line(clock_sec(app18_start_i:app18_end_i,1),app18_esr_latline...
      (app18_start_i:app18_end_i,1),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
line(clock_sec(app18_start_i:app18_end_i,1),app18_esr_latline...
      (app18_start_i:app18_end_i,2),'LineStyle','--','Color','r',...
      'LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);

```

```

ylabel('Lateral Deviation (ft)','FontName','Times New Roman',...
      'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
title('Approach Phase 8: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft Lateral Deviation','ESR Req');
set(legend1,'Location','NorthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app18_start_i,1) clock_sec(app18_end_i-5,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 18 Approach Test Lateral'
end

% Approach Phase 18 Final Approach Test
figure;
plot(clock_sec,TAS);
hold on;
plot(clock_sec,TASTarget,'g','LineWidth',1.5);
line(clock_sec(app18_start_i:app18_end_i,1),TASTarget(app18_start_i...
      :app18_end_i,1)+16.9,'LineStyle','--
','Color','r','LineWidth',1.5);
line(clock_sec(app18_start_i:app18_end_i,1),TASTarget(app18_start_i...
      :app18_end_i,1)-16.9,'LineStyle','--
','Color','r','LineWidth',1.5);
hold off;
set(gca,'XGrid','on');
set(gca,'YGrid','on');
set(gca,'FontName','Times New Roman');
set(gca,'FontSize',12);
xlabel('Time (sec)','FontName','Times New Roman','FontSize',12);
ylabel('Airspeed (ft/sec)','FontName','Times New
Roman','FontSize',12);
title('Approach Phase 18: Approach Test','FontName',...
      'Times New Roman','FontSize',14);
legend1 = legend('Aircraft TAS','Commanded TAS','ESR Req');
set(legend1,'Location','SouthEast','FontName','Times New Roman',...
      'FontSize',10);
xlim([clock_sec(app18_start_i,1) clock_sec(app18_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase 18 Approach Test TAS'
end

% Approach Phase RNP Test
figure;
plot(app_track_y(:,2),app_track_y(:,1));
line([clock_sec(app1_start_i,1) clock_sec(app18_end_i,1)],...
      [app_max_track_y app_max_track_y],'LineStyle',':','Color',...
      'g','LineWidth',1.5);
line([clock_sec(app1_start_i,1) clock_sec(app18_end_i,1)],...

```

```

        [0.125 0.125], 'LineStyle', '--', 'Color', 'r', 'LineWidth', 1.5);
set(gca, 'XGrid', 'on');
set(gca, 'YGrid', 'on');
set(gca, 'FontName', 'Times New Roman');
set(gca, 'FontSize', 12);
xlabel('Time (sec)', 'FontName', 'Times New Roman', 'FontSize', 12);
ylabel('Lateral Deviation (NM)', 'FontName', 'Times New Roman', ...
    'FontSize', 12);
title('Approach Phase: Lateral Deviation Test', 'FontName', ...
    'Times New Roman', 'FontSize', 14);
legend1 = legend('Aircraft Lateral Deviation', '95% Containment', ...
    'ESR Req');
set(legend1, 'Location', 'NorthEast', 'FontName', 'Times New Roman', ...
    'FontSize', 10);
xlim([clock_sec(app1_start_i,1) clock_sec(app18_end_i,1)]);
if TFsavefig == 1;
    print -f -dmeta 'Figures\Approach Phase Lateral Deviation Test'
end
end

if TFq == 0;
    quit_input = menu('Plot another flight phase?', 'Yes', 'No');
    if quit_input == 1;
        TFq = 0;
    else
        TFq = 1;
    end
end
end
end

```

A.2 TAS Target Function

```
function [TASTarget] = TASTarget(Static,Dynamic,OAT,IASTarget,n);
TASTarget = zeros(n,1);
for i=1:1:n;
    rho_i = (Static(i,1) + Dynamic(i,1))/(287.05*OAT(i,1));
    TASTarget(i,1) = IASTarget(i,1)*sqrt(1.225/rho_i);
end
```


A.3 Flight Phase Analysis Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Equivalent Safe Response Model                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% NAVAIR Airworthiness Airspace Integration Project
% John Southwell
% August 2010

% Flight Phase Analysis Function

% This function determines the following for a given flight segment:
% Deviations in altitude, ground track, true airspeed, and lateral track
% over time
% Maximum deviations in altitude, ground track, bank angle, and true
% airspeed during the given flight segment
% Average vertical rate and average turn rate
% Maximum turn rate
% Maximum bank angle

function [alt_dev,hdg_dev,bank_dev,TAS_dev,track_y,max_alt_dev,...
    max_hdg_dev,max_bank_dev,max_TAS_dev,avg_climbrate,avg_turnrate,...
    max_turnrate,max_bankangle,turnrate,climbrate]...
    = flight_phase_analysis(clock_sec,track_y,Track_Y,Alt,AltTarget,...
    Direction,GroundTrackTarget,Roll,BankTarget,TAS,TASTarget,clock,...
    start_i,end_i)

k = size(track_y,1);
if k == 1;
    j = 0;
else
    j = 1;
end
l = 1;
for i=start_i:1:end_i;
    track_y(k+j,1) = abs(Track_Y(i,1))*0.000164578834;
    track_y(k+j,2) = clock_sec(i,1);
    alt_dev(i,1) = Alt(i,1) - AltTarget(i,1);
    hdg_dev(i,1) = (acos(sin(Direction(i,1)*pi/180)*sin(...
        GroundTrackTarget*pi/180)+cos(Direction(i,1)*pi/180)*cos(...
        GroundTrackTarget*pi/180)))*180/pi;
    if Alt(i,1) > 20;
        TAS_dev(i,1) = TAS(i,1) - TASTarget(i,1);
    else
        TAS_dev(i,1) = 0;
```

```

    end
    bank_angle(1,1) = Roll(i,1);
    j = j + 1;
    l = l + 1;
end
bank_dev = 0;
for i=start_i:1:end_i;
    if clock_sec(i,1) >= clock_sec(start_i,1) + 0;
        bank_start_i = i;
        break;
    end
end
for i=bank_start_i:1:end_i;
    bank_dev(i,1) = Roll(i,1) - BankTarget(i,1);
end
max_alt_dev = max(abs(alt_dev));
max_hdg_dev = max(abs(hdg_dev));
max_bank_dev = max(abs(bank_dev));
max_TAS_dev = max(abs(TAS_dev));
max_bankangle = max(bank_angle);
max_turnrate = 0;
turnrate = 0;
j = 1;
for i=start_i:1:end_i-10;
    if abs(Direction(i,1) - GroundTrackTarget) > 10;
        tr = real((acos(sin(Direction(i+10,1)*pi/180)*sin(...
            Direction(i,1)*pi/180)+cos(Direction(i+10,1)*pi/180)*cos(...
            Direction(i,1)*pi/180)))*180/pi)/(clock(i+10,1) - ...
            clock(i,1))*1000);
        turnrate(j,2) = tr;
        turnrate(j,1) = clock(i,1)/1000;
        j = j + 1;
        if tr > max_turnrate;
            max_turnrate = tr;
        end
    end
end
if size(turnrate) > 1;
    avg_turnrate = mean(turnrate(:,2));
    turnrate(:,3) = avg_turnrate;
else
    avg_turnrate = 0;
end
climbrate = 0;
j = 1;
for i=start_i:1:end_i-20;
    if (abs(Alt(i,1) - AltTarget(i,1)) > 40 && abs(AltTarget(i+1,1) - ...
        Alt(i+1,1)) < abs(AltTarget(i,1) - Alt(i,1)));

```

```

        vr = (Alt(i+20,1) - Alt(i,1))/(clock(i+20,1) -
clock(i,1))*1000*60;
        climbrate(j,2) = vr;
        climbrate(j,1) = clock(i,1)/1000;
        j = j + 1;
    end
end
if size(climbrate) > 1;
    avg_climbrate = mean(climbrate(:,2));
    climbrate(:,3) = avg_climbrate;
else
    avg_climbrate = 0;
end

```

A.4 ESR climb Function

```
function [esr_climbrate,esr_descentrate] = ESRclimb(avg_climbrate)

if abs(avg_climbrate) > 500;
    esr_climbrate = 'Pass';
    esr_descentrate = 'Pass';
else
    esr_climbrate = 'Fail';
    esr_descentrate = 'Fail';
end
```

A.5 ESR Straight Level Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Equivalent Safe Response Model                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% NAVAIR Airworthiness Airspace Integration Project
% John Southwell
% August 2010

% ESR Straight and Level Function

% This function determines if the ESR requirements for straight and level
% flight have been met.  These requirements include: altitude deviation,
% ground track deviation, and airspeed deviation

function [esr_alt,esr_hdg,esr_TAS,esr_straightlevel] =
ESRstraightlevel(...
    max_alt_dev,max_hdg_dev,max_TAS_dev)

if abs(max_alt_dev) > 100;
    esr_alt = 'Fail';
else
    esr_alt = 'Pass';
end

if abs(max_hdg_dev) > 10;
    esr_hdg = 'Fail';
else
    esr_hdg = 'Pass';
end

if abs(max_TAS_dev) > 16.9;
    esr_TAS = 'Fail';
else
    esr_TAS = 'Pass';
end

TF1 = strcmp('Pass',esr_alt);
TF2 = strcmp('Pass',esr_hdg);
TF3 = strcmp('Pass',esr_TAS);

if (TF1 && TF2 && TF3) == 1;
    esr_straightlevel = 'Pass';
else
    esr_straightlevel = 'Fail';
end
```

A.6 ESR Turn Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%                               Equivalent Safe Response Model                               %  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
% NAVAIR Airworthiness Airspace Integration Project  
% John Southwell  
% August 2010  
  
% ESR Turn Rate Function  
  
% This function determines if the average turn rate obtained during a  
% given flight phase meets the ESR requirements  
  
function [esr_turnrate] = ESRturn(avg_turnrate)  
  
if abs(avg_turnrate) < 3;  
    esr_turnrate = 'Fail';  
else  
    esr_turnrate = 'Pass';  
end
```

A.7 ESR Final Approach Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%                               Equivalent Safe Response Model                               %  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
% NAVAIR Airworthiness Airspace Integration Project  
% John Southwell  
% August 2010  
  
% ESR Final Approach Function  
  
% This function determines if the ESR requirements for the final approach  
% phase have been met.  These requirements include: altitude deviation  
% (glideslope deviation), lateral deviation, ground track deviation,  
% airspeed deviation, and bank angle deviation  
  
function [esr_alt,esr_TAS,esr_lat,esr_approach,gs_dev_max,lat_dev_max]...  
    = ESRFinalApproach(alt_dev,max_TAS_dev,Track_X,Track_Y,start_i,end_i)  
  
for i=start_i:1:end_i;  
    if 73.5 + Track_X(i,1)*tan(0.525*pi/180) > 100;  
        if alt_dev(i,1) > 100;  
            esr_alt = 'Fail';  
            break;  
        else  
            esr_alt = 'Pass';  
        end  
    else  
        if alt_dev(i,1) > 73.5 + Track_X(i,1)*tan(0.525*pi/180);  
            esr_alt = 'Fail';  
            break;  
        else  
            esr_alt = 'Pass';  
        end  
    end  
end  
  
for i=start_i:1:end_i;  
    gs_dev(i,1) = (atan(alt_dev(i,1)/(Track_X(i,1)+8016)))*180/pi;  
    lat_dev(i,1) = (atan(abs(Track_Y(i,1))/(Track_X(i,1)+8016)))*180/pi;  
end  
  
gs_dev_max = max(gs_dev(:,1));  
lat_dev_max = max(lat_dev(:,1));  
  
for i=start_i:1:end_i;
```

```

        if Track_Y(i,1) > 262 + Track_X(i,1)*tan(1.875*pi/180);
            esr_lat = 'Fail';
            break;
        else
            esr_lat = 'Pass';
        end
    end

    if abs(max_TAS_dev) > 16.9;
        esr_TAS = 'Fail';
    else
        esr_TAS = 'Pass';
    end

    TF1 = strcmp('Pass',esr_alt);
    TF3 = strcmp('Pass',esr_TAS);
    TF5 = strcmp('Pass',esr_lat);

    if (TF1 && TF3 && TF5) == 1;
        esr_approach = 'Pass';
    else
        esr_approach = 'Fail';
    end
end

```


A.8 ESR Missed Approach Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                               Equivalent Safe Response Model                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% NAVAIR Airworthiness Airspace Integration Project
% John Southwell
% August 2010

% ESR Missed Approach Function

% This function determines if the ESR requirements for the missed approach
% phase have been met. These requirements include: altitude deviation
% (glideslope deviation), lateral deviation, ground track deviation,
% airspeed deviation, and bank angle deviation. This function differs
% from
% the final approach function in that the allowed altitude deviations and
% lateral deviations increase along the flight path.

function [esr_alt,esr_TAS,esr_lat,esr_approach,gs_dev_max,lat_dev_max]...
    = ESRMisApproach(alt_dev,max_TAS_dev,Track_X,Track_Y,start_i,end_i)

for i=start_i:1:end_i;
    if 73.5 + (Track_X(start_i,1) - Track_X(i,1))*tan(0.525*pi/180) > 100;
        if alt_dev(i,1) > 100;
            esr_alt = 'Fail';
            break;
        else
            esr_alt = 'Pass';
        end
    else
        if alt_dev(i,1) > 73.5 + (Track_X(start_i,1) - Track_X(i,1))*...
            tan(0.525*pi/180);
            esr_alt = 'Fail';
            break;
        else
            esr_alt = 'Pass';
        end
    end
end

for i=start_i:1:end_i;
    gs_dev(i,1) = (atan(alt_dev(i,1)/(Track_X(i,1)+8016)))*180/pi;
    lat_dev(i,1) = (atan(abs(Track_Y(i,1))/(Track_X(i,1)+8016)))*180/pi;
end
```

```

gs_dev_max = max(gs_dev(:,1));
lat_dev_max = max(lat_dev(:,1));

for i=start_i:1:end_i;
    if Track_Y(i,1) > 262 + (Track_X(start_i,1) - Track_X(i,1))*...
        tan(1.875*pi/180);
        esr_lat = 'Fail';
        break;
    else
        esr_lat = 'Pass';
    end
end

if abs(max_TAS_dev) > 16.9;
    esr_TAS = 'Fail';
else
    esr_TAS = 'Pass';
end

TF1 = strcmp('Pass',esr_alt);
TF3 = strcmp('Pass',esr_TAS);
TF5 = strcmp('Pass',esr_lat);

if (TF1 && TF3 && TF5) == 1;
    esr_approach = 'Pass';
else
    esr_approach = 'Fail';
end

```

Appendix B ArcWulf ESR Results

B.1 Calm Conditions Results

Aircraft: ArcWulf

Test Date: 8 / 23 / 2010

Test Weather Conditions: Calm

Final ESR Grade: Pass

Flight Phase	Pass/Fail
Departure	Pass
En Route	Pass
Arrival	Pass
Approach	Pass

Flight Phase	Test	Pass/Fail	Actual	ESR Requirement
Departure 2	Climb	Pass	714 ft/min	≥ 500 ft/min
Departure 5	Descent	Pass	-678 ft/min	≤ -500 ft/min
Departure 8	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	≤ 100 ft
	Course Deviation	Pass	1.9 deg	≤ 10 deg
	Airspeed Deviation	Pass	1.1 ft/sec	≤ 16.9 ft/sec
Departure 9	Turn	Pass	10.4 deg/sec	≥ 3 deg/sec
Departure 10	Turn	Pass	13.9 deg/sec	≥ 3 deg/sec
Departure 11	Turn	Pass	15.1 deg/sec	≥ 3 deg/sec
Departure 12	Turn	Pass	13.8 deg/sec	≥ 3 deg/sec
Departure	RNP Level	Pass	0.026 NM	≤ 0.125 NM
En Route 1	Climb	Pass	712 ft/min	≥ 500 ft/min
En Route 4	Descent	Pass	-721 ft/min	≤ -500 ft/min
En Route 7	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	≤ 100 ft
	Course Deviation	Pass	2.0 deg	≤ 10 deg
	Airspeed Deviation	Pass	0.4 ft/sec	≤ 16.9 ft/sec

En Route	RNP Level	Pass	0.030 NM	<=0.25 NM
Arrival 1	Climb	Pass	734 ft/min	>=500 ft/min
Arrival 4	Descent	Pass	-697 ft/min	<=-500 ft/min
Arrival 7	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	2.2 deg	<=10 deg
	Airspeed Deviation	Pass	0 ft/sec	<=16.9 ft/sec
Arrival 8	Turn	Pass	10.7 deg/sec	>=3 deg/sec
Arrival 9	Turn	Pass	14.3 deg/sec	>=3 deg/sec
Arrival 10	Turn	Pass	13.6 deg/sec	>=3 deg/sec
Arrival 11	Turn	Pass	14.4 deg/sec	>=3 deg/sec
Arrival	RNP Level	Pass	0.029 NM	<=0.25 NM
Approach 1	Descent	Pass	-689 ft/min	<=-500 ft/min
Approach 5	Turn	Pass	10.8 deg/sec	>=3 deg/sec
Approach 6	Turn	Pass	13.1 deg/sec	>=3 deg/sec
Approach 7	Final Approach	Pass		
	Altitude Deviation	Pass	17 ft	<=100 ft
	Glide Slope Deviation	Pass	0.009 deg	<=0.525 deg
	Lateral Deviation	Pass	0.023 deg	<=1.875 deg
	Airspeed Deviation	Pass	0.9 ft/sec	<=16.9 ft/sec
Approach 8	Missed Approach	Pass		
	Altitude Deviation	Pass	17 ft	<=100 ft
	Climb Angle Deviation	Pass	0.113 deg	<=0.525 deg
	Lateral Deviation	Pass	0.023 deg	<=1.875 deg
	Airspeed Deviation	Pass	1.0 ft/sec	<=16.9 ft/sec
Approach 12	Turn	Pass	12.5 deg/sec	>=3 deg/sec
Approach 13	Turn	Pass	12.4 deg/sec	>=3 deg/sec
Approach 14	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	2.0 deg	<=10 deg
	Airspeed Deviation	Pass	0.4 ft/sec	<=16.9 ft/sec

Approach 16	Turn	Pass	13.5 deg/sec	>=3 deg/sec
Approach 17	Turn	Pass	17.6 deg/sec	>=3 deg/sec
Approach 18	Final Approach	Pass		
	Altitude Deviation	Pass	17 ft	<=100 ft
	Glide Slope Deviation	Pass	0.008 deg	<=0.525 deg
	Lateral Deviation	Pass	0.021 deg	<=1.875 deg
	Airspeed Deviation	Pass	1.1 ft/sec	<=16.9 ft/sec
Approach	RNP Level	Pass	0.012 NM	<=0.125 NM

Figure B.1.1: Departure Climb Rate Test Altitude vs. Time, Calm

Figure B.1.2: Departure Climb Rate Test Vertical Rate vs. Time, Calm

Figure B.1.3: Departure Descent Rate Test Altitude vs. Time, Calm

Figure B.1.4: Departure Descent Rate Test Vertical Rate vs. Time, Calm

Figure B.1.5: Departure Straight and Level Test Altitude vs. Time, Calm

Figure B.1.6: Departure Straight and Level Test Course vs. Time, Calm

Figure B.1.7: Departure Straight and Level Test Airspeed vs. Time, Calm

Figure B.1.8: Departure Turn Rate Test Turn Rate vs. Time, Calm

Figure B.1.9: Departure RNP Test Lateral Deviation vs. Time, Calm

Figure B.1.10: En Route Climb Rate Test Altitude vs. Time, Calm

Figure B.1.11: En Route Climb Rate Test Vertical Rate vs. Time, Calm

Figure B.1.12: En Route Descent Rate Test Altitude vs. Time, Calm

Figure B.1.13: En Route Descent Rate Test Vertical Rate vs. Time, Calm

Figure B.1.14: En Route Straight and Level Test Altitude vs. Time, Calm

Figure B.1.15: En Route Straight and Level Test Course vs. Time, Calm

Figure B.1.16: En Route Straight and Level Test Airspeed vs. Time, Calm

Figure B.1.17: En Route RNP Test Lateral Deviation vs. Time, Calm

Figure B.1.18: Arrival Climb Rate Test Altitude vs. Time, Calm

Figure B.1.19: Arrival Climb Rate Test Vertical Rate vs. Time, Calm

Figure B.1.20: Arrival Descent Rate Test Altitude vs. Time, Calm

Figure B.1.21: Arrival Descent Rate Test Vertical Rate vs. Time, Calm

Figure B.1.22: Arrival Straight and Level Test Altitude vs. Time, Calm

Figure B.1.23: Arrival Straight and Level Test Course vs. Time, Calm

Figure B.1.24: Arrival Straight and Level Test Airspeed vs. Time, Calm

Figure B.1.25: Arrival Turn Rate Test Turn Rate vs. Time, Calm

Figure B.1.26: Arrival RNP Test Lateral Deviation vs. Time, Calm

Figure B.1.27: Approach Descent Rate Test Altitude vs. Time, Calm

Figure B.1.28: Approach Descent Rate Test Vertical Rate vs. Time, Calm

Figure B.1.29: Approach Turn Rate Test Turn Rate vs. Time, Calm

Figure B.1.30: Approach Final Approach Test Altitude vs. Time, Calm

Figure B.1.31: Approach Final Approach Test Lateral Deviation vs. Time, Calm

Figure B.1.32: Approach Final Approach Test Airspeed vs. Time, Calm

Figure B.1.33: Approach Missed Approach Test Altitude vs. Time, Calm

Figure B.1.34: Approach Missed Approach Test Lateral Deviation vs. Time, Calm

Figure B.1.35: Approach Missed Approach Test Airspeed vs. Time, Calm

Figure B.1.36: Approach Turn Rate Test Turn Rate vs. Time, Calm

Figure B.1.37: Approach Straight and Level Test Altitude vs. Time, Calm

Figure B.1.38: Approach Straight and Level Test Course vs. Time, Calm

Figure B.1.39: Approach Straight and Level Test Airspeed vs. Time, Calm

Figure B.1.40: Approach Turn Rate Test Turn Rate vs. Time, Calm

Figure B.1.41: Approach Final Approach Test Altitude vs. Time, Calm

Figure B.1.42: Approach Final Approach Test Lateral Deviation vs. Time, Calm

Figure B.1.43: Approach Final Approach Test Airspeed vs. Time, Calm

Figure B.1.44: Approach RNP Test Lateral Deviation vs. Time, Calm

B.2 Light Turbulence Condition

Aircraft: ArcWulf

Test Date: 8 / 13 / 2010

Test Weather Conditions: Light turbulence

Final ESR Grade: Pass

Flight Phase	Pass/Fail
Departure	Pass
En Route	Pass
Arrival	Pass
Approach	Pass

Flight Phase	Test	Pass/Fail	Actual	ESR Requirement
Departure 2	Climb	Pass	702 ft/min	≥ 500 ft/min
Departure 5	Descent	Pass	-629 ft/min	≤ -500 ft/min
Departure 8	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	≤ 100 ft
	Course Deviation	Pass	5.4 deg	≤ 10 deg
	Airspeed Deviation	Pass	3.0 ft/sec	≤ 16.9 ft/sec
Departure 9	Turn	Pass	10.0 deg/sec	≥ 3 deg/sec
Departure 10	Turn	Pass	15.0 deg/sec	≥ 3 deg/sec
Departure 11	Turn	Pass	15.4 deg/sec	≥ 3 deg/sec
Departure 12	Turn	Pass	14.6 deg/sec	≥ 3 deg/sec
Departure	RNP Level	Pass	0.026 NM	≤ 0.125 NM
En Route 1	Climb	Pass	701 ft/min	≥ 500 ft/min
En Route 4	Descent	Pass	-701 ft/min	≤ -500 ft/min
En Route 7	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	≤ 100 ft
	Course Deviation	Pass	4.8 deg	≤ 10 deg
	Airspeed Deviation	Pass	2.4 ft/sec	≤ 16.9 ft/sec
En Route	RNP Level	Pass	0.030 NM	≤ 0.25 NM
Arrival 1	Climb	Pass	733 ft/min	≥ 500 ft/min

Arrival 4	Descent	Pass	-612 ft/min	<=-500 ft/min
Arrival 7	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	<=100 ft
	Course Deviation	Pass	4.1 deg	<=10 deg
	Airspeed Deviation	Pass	3 ft/sec	<=16.9 ft/sec
Arrival 8	Turn	Pass	11.2 deg/sec	>=3 deg/sec
Arrival 9	Turn	Pass	14.2 deg/sec	>=3 deg/sec
Arrival 10	Turn	Pass	12.8 deg/sec	>=3 deg/sec
Arrival 11	Turn	Pass	13.6 deg/sec	>=3 deg/sec
Arrival	RNP Level	Pass	0.031 NM	<=0.25 NM
Approach 1	Descent	Pass	-765 ft/min	<=-500 ft/min
Approach 5	Turn	Pass	10.5 deg/sec	>=3 deg/sec
Approach 6	Turn	Pass	13.4 deg/sec	>=3 deg/sec
Approach 7	Final Approach	Pass		
	Altitude Deviation	Pass	20 ft	<=100 ft
	Glide Slope Deviation	Pass	0.012 deg	<=0.525 deg
	Lateral Deviation	Pass	0.096 deg	<=1.875 deg
	Airspeed Deviation	Pass	8.9 ft/sec	<=16.9 ft/sec
Approach 8	Missed Approach	Pass		
	Altitude Deviation	Pass	20 ft	<=100 ft
	Climb Angle Deviation	Pass	0.126 deg	<=0.525 deg
	Lateral Deviation	Pass	0.053 deg	<=1.875 deg
	Airspeed Deviation	Pass	8.5 ft/sec	<=16.9 ft/sec
Approach 12	Turn	Pass	13.1 deg/sec	>=3 deg/sec
Approach 13	Turn	Pass	12.4 deg/sec	>=3 deg/sec
Approach 14	Straight and Level	Pass		
	Altitude Deviation	Pass	3 ft	<=100 ft
	Course Deviation	Pass	7.3 deg	<=10 deg
	Airspeed Deviation	Pass	5.5 ft/sec	<=16.9 ft/sec
Approach 16	Turn	Pass	14.1 deg/sec	>=3 deg/sec
Approach 17	Turn	Pass	16.9 deg/sec	>=3 deg/sec

Approach 18	Final Approach	Pass		
	Altitude Deviation	Pass	25 ft	<=100 ft
	Glide Slope Deviation	Pass	0.116 deg	<=0.525 deg
	Lateral Deviation	Pass	0.082 deg	<=1.875 deg
	Airspeed Deviation	Pass	15.0 ft/sec	<=16.9 ft/sec
Approach	RNP Level	Pass	0.012 NM	<=0.125 NM

Figure B.2.1: Departure Climb Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.2: Departure Climb Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.3: Departure Descent Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.4: Departure Descent Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.5: Departure Straight and Level Test Altitude vs. Time, Light Turbulence

Figure B.2.6: Departure Straight and Level Test Course vs. Time, Light Turbulence

Figure B.2.7: Departure Straight and Level Test Airspeed vs. Time, Light Turbulence

Figure B.2.8: Departure Turn Rate Test Turn Rate vs. Time, Light Turbulence

Figure B.2.9: Departure RNP Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.10: En Route Climb Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.11: En Route Climb Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.12: En Route Descent Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.13: En Route Descent Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.14: En Route Straight and Level Test Altitude vs. Time, Light Turbulence

Figure B.2.15: En Route Straight and Level Test Course vs. Time, Light Turbulence

Figure B.2.16: En Route Straight and Level Test Airspeed vs. Time, Light Turbulence

Figure B.2.17: En Route RNP Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.18: Arrival Climb Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.19: Arrival Climb Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.20: Arrival Descent Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.21: Arrival Descent Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.22: Arrival Straight and Level Test Altitude vs. Time, Light Turbulence

Figure B.2.23: Arrival Straight and Level Test Course vs. Time, Light Turbulence

Figure B.2.24: Arrival Straight and Level Test Airspeed vs. Time, Light Turbulence

Figure B.2.25: Arrival Turn Rate Test Turn Rate vs. Time, Light Turbulence

Figure B.2.26: Arrival RNP Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.27: Approach Descent Rate Test Altitude vs. Time, Light Turbulence

Figure B.2.28: Approach Descent Rate Test Vertical Rate vs. Time, Light Turbulence

Figure B.2.29: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence

Figure B.2.30: Approach Final Approach Test Altitude vs. Time, Light Turbulence

Figure B.2.31: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.32: Approach Final Approach Test Airspeed vs. Time, Light Turbulence

Figure B.2.33: Approach Missed Approach Test Altitude vs. Time, Light Turbulence

Figure B.2.34: Approach Missed Approach Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.35: Approach Missed Approach Test Airspeed vs. Time, Light Turbulence

Figure B.2.36: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence

Figure B.2.37: Approach Straight and Level Test Altitude vs. Time, Light Turbulence

Figure B.2.38: Approach Straight and Level Test Course vs. Time, Light Turbulence

Figure B.2.39: Approach Straight and Level Test Airspeed vs. Time, Light Turbulence

Figure B.2.40: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence

Figure B.2.41: Approach Final Approach Test Altitude vs. Time, Light Turbulence

Figure B.2.42: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence

Figure B.2.43: Approach Final Approach Test Airspeed vs. Time, Light Turbulence

Figure B.2.44: Approach RNP Test Lateral Deviation vs. Time, Light Turbulence

B.3 14 knot wind from the East Condition

Aircraft: ArcWulf

Test Date: 9 / 1 / 2010

Test Weather Conditions: East 14 Knot Wind

Final ESR Grade: Pass

Flight Phase	Pass/Fail
Departure	Pass
En Route	Pass
Arrival	Pass
Approach	Pass

Flight Phase	Test	Pass/Fail	Actual	ESR Requirement
Departure 2	Climb	Pass	709 ft/min	>=500 ft/min
Departure 5	Descent	Pass	-654 ft/min	<=-500 ft/min
Departure 8	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	1.5 deg	<=10 deg
	Airspeed Deviation	Pass	0.4 ft/sec	<=16.9 ft/sec
Departure 9	Turn	Pass	8.3 deg/sec	>=3 deg/sec
Departure 10	Turn	Pass	8.1 deg/sec	>=3 deg/sec
Departure 11	Turn	Pass	13.2 deg/sec	>=3 deg/sec
Departure 12	Turn	Pass	7.4 deg/sec	>=3 deg/sec
Departure	RNP Level	Pass	0.041 NM	<=0.125 NM
En Route 1	Climb	Pass	713 ft/min	>=500 ft/min
En Route 4	Descent	Pass	-648 ft/min	<=-500 ft/min
En Route 7	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	2.6 deg	<=10 deg
	Airspeed Deviation	Pass	1.1 ft/sec	<=16.9 ft/sec
En Route	RNP Level	Pass	0.044 NM	<=0.25 NM
Arrival 1	Climb	Pass	722 ft/min	>=500 ft/min

Arrival 4	Descent	Pass	-644 ft/min	<=-500 ft/min
Arrival 7	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	2.0 deg	<=10 deg
	Airspeed Deviation	Pass	0 ft/sec	<=16.9 ft/sec
Arrival 8	Turn	Pass	5.4 deg/sec	>=3 deg/sec
Arrival 9	Turn	Pass	11.1 deg/sec	>=3 deg/sec
Arrival 10	Turn	Pass	8.5 deg/sec	>=3 deg/sec
Arrival 11	Turn	Pass	10.5 deg/sec	>=3 deg/sec
Arrival	RNP Level	Pass	0.045 NM	<=0.25 NM
Approach 1	Descent	Pass	-646 ft/min	<=-500 ft/min
Approach 5	Turn	Pass	5.0 deg/sec	>=3 deg/sec
Approach 6	Turn	Pass	5.2 deg/sec	>=3 deg/sec
Approach 7	Final Approach	Pass		
	Altitude Deviation	Pass	19 ft	<=100 ft
	Glide Slope Deviation	Pass	0.008 deg	<=0.525 deg
	Lateral Deviation	Pass	0.159 deg	<=1.875 deg
	Airspeed Deviation	Pass	0.4 ft/sec	<=16.9 ft/sec
Approach 8	Missed Approach	Pass		
	Altitude Deviation	Pass	10 ft	<=100 ft
	Climb Angle Deviation	Pass	0.069 deg	<=0.525 deg
	Lateral Deviation	Pass	0.008 deg	<=1.875 deg
	Airspeed Deviation	Pass	0.4 ft/sec	<=16.9 ft/sec
Approach 12	Turn	Pass	8.7 deg/sec	>=3 deg/sec
Approach 13	Turn	Pass	10.9 deg/sec	>=3 deg/sec
Approach 14	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	<=100 ft
	Course Deviation	Pass	1.8 deg	<=10 deg
	Airspeed Deviation	Pass	0.9 ft/sec	<=16.9 ft/sec
Approach 16	Turn	Pass	6.2 deg/sec	>=3 deg/sec
Approach 17	Turn	Pass	15.1 deg/sec	>=3 deg/sec

Approach 18	Final Approach	Pass		
	Altitude Deviation	Pass	10 ft	<=100 ft
	Glide Slope Deviation	Pass	0.003 deg	<=0.525 deg
	Lateral Deviation	Pass	0.011 deg	<=1.875 deg
	Airspeed Deviation	Pass	0.5 ft/sec	<=16.9 ft/sec
Approach	RNP Level	Pass	0.035 NM	<=0.125 NM

Figure B.3.1: Departure Climb Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.2: Departure Climb Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.3: Departure Descent Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.4: Departure Descent Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.5: Departure Straight and Level Test Altitude vs. Time, 14kt E Wind

Figure B.3.6: Departure Straight and Level Test Course vs. Time, 14kt E Wind

Figure B.3.7: Departure Straight and Level Test Airspeed vs. Time, 14kt E Wind

Figure B.3.8: Departure Turn Rate Test Turn Rate vs. Time, 14kt E Wind

Figure B.3.9: Departure RNP Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.10: En Route Climb Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.11: En Route Climb Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.12: En Route Descent Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.13: En Route Descent Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.14: En Route Straight and Level Test Altitude vs. Time, 14kt E Wind

Figure B.3.15: En Route Straight and Level Test Course vs. Time, 14kt E Wind

Figure B.3.16: En Route Straight and Level Test Airspeed vs. Time, 14kt E Wind

Figure B.3.17: En Route RNP Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.18: Arrival Climb Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.19: Arrival Climb Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.20: Arrival Descent Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.21: Arrival Descent Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.22: Arrival Straight and Level Test Altitude vs. Time, 14kt E Wind

Figure B.3.23: Arrival Straight and Level Test Course vs. Time, 14kt E Wind

Figure B.3.24: Arrival Straight and Level Test Airspeed vs. Time, 14kt E Wind

Figure B.3.25: Arrival Turn Rate Test Turn Rate vs. Time, 14kt E Wind

Figure B.3.26: Arrival RNP Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.27: Approach Descent Rate Test Altitude vs. Time, 14kt E Wind

Figure B.3.28: Approach Descent Rate Test Vertical Rate vs. Time, 14kt E Wind

Figure B.3.29: Approach Turn Rate Test Turn Rate vs. Time, 14kt E Wind

Figure B.3.30: Approach Final Approach Test Altitude vs. Time, 14kt E Wind

Figure B.3.31: Approach Final Approach Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.32: Approach Final Approach Test Airspeed vs. Time, 14kt E Wind

Figure B.3.33: Approach Missed Approach Test Altitude vs. Time, 14kt E Wind

Figure B.3.34: Approach Missed Approach Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.35: Approach Missed Approach Test Airspeed vs. Time, 14kt E Wind

Figure B.3.36: Approach Turn Rate Test Turn Rate vs. Time, 14kt E Wind

Figure B.3.37: Approach Straight and Level Test Altitude vs. Time, 14kt E Wind

Figure B.3.38: Approach Straight and Level Test Course vs. Time, 14kt E Wind

Figure B.3.39: Approach Straight and Level Test Airspeed vs. Time, 14kt E Wind

Figure B.3.40: Approach Turn Rate Test Turn Rate vs. Time, 14kt E Wind

Figure B.3.41: Approach Final Approach Test Altitude vs. Time, 14kt E Wind

Figure B.3.42: Approach Final Approach Test Lateral Deviation vs. Time, 14kt E Wind

Figure B.3.43: Approach Final Approach Test Airspeed vs. Time, 14kt E Wind

Figure B.3.44: Approach RNP Test Lateral Deviation vs. Time, 14kt E Wind

B.4 ArcWulf model with increased weight/inertia in light turbulence condition

Aircraft: ArcWulf

Test Date: 9 / 20 / 2010

Test Weather Conditions: Light turbulence

Final ESR Grade: Pass

Flight Phase	Pass/Fail
Departure	Pass
En Route	Pass
Arrival	Pass
Approach	Pass

Flight Phase	Test	Pass/Fail	Actual	ESR Requirement
Departure 2	Climb	Pass	895 ft/min	≥ 500 ft/min
Departure 5	Descent	Pass	-621 ft/min	≤ -500 ft/min
Departure 8	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	≤ 100 ft
	Course Deviation	Pass	1.4 deg	≤ 10 deg
	Airspeed Deviation	Pass	2.1 ft/sec	≤ 16.9 ft/sec
Departure 9	Turn	Pass	7.8 deg/sec	≥ 3 deg/sec
Departure 10	Turn	Pass	11.6 deg/sec	≥ 3 deg/sec
Departure 11	Turn	Pass	12.3 deg/sec	≥ 3 deg/sec
Departure 12	Turn	Pass	11.8 deg/sec	≥ 3 deg/sec
Departure	RNP Level	Pass	0.042 NM	≤ 0.125 NM
En Route 1	Climb	Pass	894 ft/min	≥ 500 ft/min
En Route 4	Descent	Pass	-633 ft/min	≤ -500 ft/min
En Route 7	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	≤ 100 ft
	Course Deviation	Pass	1.6 deg	≤ 10 deg
	Airspeed Deviation	Pass	2.2 ft/sec	≤ 16.9 ft/sec
En Route	RNP Level	Pass	0.050 NM	≤ 0.25 NM

Arrival 1	Climb	Pass	928 ft/min	>=500 ft/min
Arrival 4	Descent	Pass	-597 ft/min	<=-500 ft/min
Arrival 7	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	<=100 ft
	Course Deviation	Pass	0.9 deg	<=10 deg
	Airspeed Deviation	Pass	3 ft/sec	<=16.9 ft/sec
Arrival 8	Turn	Pass	8.8 deg/sec	>=3 deg/sec
Arrival 9	Turn	Pass	11.8 deg/sec	>=3 deg/sec
Arrival 10	Turn	Pass	11.7 deg/sec	>=3 deg/sec
Arrival 11	Turn	Pass	11.7 deg/sec	>=3 deg/sec
Arrival	RNP Level	Pass	0.046 NM	<=0.25 NM
Approach 1	Descent	Pass	-687 ft/min	<=-500 ft/min
Approach 5	Turn	Pass	9.5 deg/sec	>=3 deg/sec
Approach 6	Turn	Pass	10.9 deg/sec	>=3 deg/sec
Approach 7	Final Approach	Pass		
	Altitude Deviation	Pass	22 ft	<=100 ft
	Glide Slope Deviation	Pass	0.013 deg	<=0.525 deg
	Lateral Deviation	Pass	0.023 deg	<=1.875 deg
	Airspeed Deviation	Pass	7.8 ft/sec	<=16.9 ft/sec
Approach 8	Missed Approach	Pass		
	Altitude Deviation	Pass	22 ft	<=100 ft
	Climb Angle Deviation	Pass	0.130 deg	<=0.525 deg
	Lateral Deviation	Pass	0.009 deg	<=1.875 deg
	Airspeed Deviation	Pass	12.1 ft/sec	<=16.9 ft/sec
Approach 12	Turn	Pass	10.5 deg/sec	>=3 deg/sec
Approach 13	Turn	Pass	9.6 deg/sec	>=3 deg/sec
Approach 14	Straight and Level	Pass		
	Altitude Deviation	Pass	4 ft	<=100 ft
	Course Deviation	Pass	1.4 deg	<=10 deg
	Airspeed Deviation	Pass	7.8 ft/sec	<=16.9 ft/sec

Approach 16	Turn	Pass	10.7 deg/sec	≥ 3 deg/sec
Approach 17	Turn	Pass	14.4 deg/sec	≥ 3 deg/sec
Approach 18	Final Approach	Pass		
	Altitude Deviation	Pass	23 ft	≤ 100 ft
	Glide Slope Deviation	Pass	0.014 deg	≤ 0.525 deg
	Lateral Deviation	Pass	0.029 deg	≤ 1.875 deg
	Airspeed Deviation	Pass	10.8 ft/sec	≤ 16.9 ft/sec
Approach	RNP Level	Pass	0.018 NM	≤ 0.125 NM

Figure B.4.1: Departure Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.2: Departure Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.3: Departure Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.4: Departure Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.5: Departure Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.6: Departure Straight and Level Test Course vs. Time, Light Turbulence with increased weight

Figure B.4.7: Departure Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.8: Departure Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight

Figure B.4.9: Departure RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.10: En Route Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.11: En Route Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.12: En Route Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.13: En Route Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.14: En Route Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.15: En Route Straight and Level Test Course vs. Time, Light Turbulence with increased weight

Figure B.4.16: En Route Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.17: En Route RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.18: Arrival Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.19: Arrival Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.20: Arrival Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.21: Arrival Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.22: Arrival Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.23: Arrival Straight and Level Test Course vs. Time, Light Turbulence with increased weight

Figure B.4.24: Arrival Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.25: Arrival Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight

Figure B.4.26: Arrival RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.27: Approach Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.28: Approach Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight

Figure B.4.29: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight

Figure B.4.30: Approach Final Approach Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.31: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.32: Approach Final Approach Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.33: Approach Missed Approach Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.34: Approach Missed Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.35: Approach Missed Approach Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.36: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight

Figure B.4.37: Approach Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.38: Approach Straight and Level Test Course vs. Time, Light Turbulence with increased weight

Figure B.4.39: Approach Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.40: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight

Figure B.4.41: Approach Final Approach Test Altitude vs. Time, Light Turbulence with increased weight

Figure B.4.42: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight

Figure B.4.43: Approach Final Approach Test Airspeed vs. Time, Light Turbulence with increased weight

Figure B.4.44: Approach RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight

B.5 ArcWulf model with increased weight/inertia and autopilot tuning in light turbulence condition

Aircraft: ArcWulf

Test Date: 9 / 16 / 2010

Test Weather Conditions: Light turbulence

Final ESR Grade: Pass

Flight Phase	Pass/Fail
Departure	Pass
En Route	Pass
Arrival	Pass
Approach	Pass

Flight Phase	Test	Pass/Fail	Actual	ESR Requirement
Departure 2	Climb	Pass	911 ft/min	≥ 500 ft/min
Departure 5	Descent	Pass	-717 ft/min	≤ -500 ft/min
Departure 8	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	≤ 100 ft
	Course Deviation	Pass	1.7 deg	≤ 10 deg
	Airspeed Deviation	Pass	2.5 ft/sec	≤ 16.9 ft/sec
Departure 9	Turn	Pass	6.1 deg/sec	≥ 3 deg/sec
Departure 10	Turn	Pass	5.9 deg/sec	≥ 3 deg/sec
Departure 11	Turn	Pass	10.6 deg/sec	≥ 3 deg/sec
Departure 12	Turn	Pass	5.4 deg/sec	≥ 3 deg/sec
Departure	RNP Level	Pass	0.106 NM	≤ 0.125 NM
En Route 1	Climb	Pass	916 ft/min	≥ 500 ft/min
En Route 4	Descent	Pass	-685 ft/min	≤ -500 ft/min
En Route 7	Straight and Level	Pass		
	Altitude Deviation	Pass	1 ft	≤ 100 ft
	Course Deviation	Pass	1.2 deg	≤ 10 deg
	Airspeed Deviation	Pass	1.9 ft/sec	≤ 16.9 ft/sec
En Route	RNP Level	Pass	0.115 NM	≤ 0.25 NM

Arrival 1	Climb	Pass	926 ft/min	>=500 ft/min
Arrival 4	Descent	Pass	-878 ft/min	<=-500 ft/min
Arrival 7	Straight and Level	Pass		
	Altitude Deviation	Pass	2 ft	<=100 ft
	Course Deviation	Pass	1.1 deg	<=10 deg
	Airspeed Deviation	Pass	2 ft/sec	<=16.9 ft/sec
Arrival 8	Turn	Pass	4.6 deg/sec	>=3 deg/sec
Arrival 9	Turn	Pass	7.6 deg/sec	>=3 deg/sec
Arrival 10	Turn	Pass	6.3 deg/sec	>=3 deg/sec
Arrival 11	Turn	Pass	8.0 deg/sec	>=3 deg/sec
Arrival	RNP Level	Pass	0.152 NM	<=0.25 NM
Approach 1	Descent	Pass	-628 ft/min	<=-500 ft/min
Approach 5	Turn	Pass	5.1 deg/sec	>=3 deg/sec
Approach 6	Turn	Pass	5.8 deg/sec	>=3 deg/sec
Approach 7	Final Approach	Pass		
	Altitude Deviation	Pass	22 ft	<=100 ft
	Glide Slope Deviation	Pass	0.017 deg	<=0.525 deg
	Lateral Deviation	Pass	0.084 deg	<=1.875 deg
	Airspeed Deviation	Pass	9.1 ft/sec	<=16.9 ft/sec
Approach 8	Missed Approach	Pass		
	Altitude Deviation	Pass	23 ft	<=100 ft
	Climb Angle Deviation	Pass	0.159 deg	<=0.525 deg
	Lateral Deviation	Pass	0.022 deg	<=1.875 deg
	Airspeed Deviation	Pass	10.6 ft/sec	<=16.9 ft/sec
Approach 12	Turn	Pass	5.9 deg/sec	>=3 deg/sec
Approach 13	Turn	Pass	5.5 deg/sec	>=3 deg/sec
Approach 14	Straight and Level	Pass		
	Altitude Deviation	Pass	3 ft	<=100 ft
	Course Deviation	Pass	6.3 deg	<=10 deg
	Airspeed Deviation	Pass	5.2 ft/sec	<=16.9 ft/sec

Approach 16	Turn	Pass	6.0 deg/sec	≥ 3 deg/sec
Approach 17	Turn	Pass	13.3 deg/sec	≥ 3 deg/sec
Approach 18	Final Approach	Pass		
	Altitude Deviation	Pass	23 ft	≤ 100 ft
	Glide Slope Deviation	Pass	0.012 deg	≤ 0.525 deg
	Lateral Deviation	Pass	0.053 deg	≤ 1.875 deg
	Airspeed Deviation	Pass	10.8 ft/sec	≤ 16.9 ft/sec
Approach	RNP Level	Pass	0.042 NM	≤ 0.125 NM

Figure B.5.1: Departure Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.2: Departure Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.3: Departure Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.4: Departure Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.5: Departure Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.6: Departure Straight and Level Test Course vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.7: Departure Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.8: Departure Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.9: Departure RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.10: En Route Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.11: En Route Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.12: En Route Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.13: En Route Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.14: En Route Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.15: En Route Straight and Level Test Course vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.16: En Route Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.17: En Route RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.18: Arrival Climb Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.19: Arrival Climb Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.20: Arrival Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.21: Arrival Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.22: Arrival Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.23: Arrival Straight and Level Test Course vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.24: Arrival Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.25: Arrival Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.26: Arrival RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.27: Approach Descent Rate Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.28: Approach Descent Rate Test Vertical Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.29: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.30: Approach Final Approach Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.31: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.32: Approach Final Approach Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.33: Approach Missed Approach Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.34: Approach Missed Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.35: Approach Missed Approach Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.36: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.37: Approach Straight and Level Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.38: Approach Straight and Level Test Course vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.39: Approach Straight and Level Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.40: Approach Turn Rate Test Turn Rate vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.41: Approach Final Approach Test Altitude vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.42: Approach Final Approach Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.43: Approach Final Approach Test Airspeed vs. Time, Light Turbulence with increased weight and autopilot tuning

Figure B.5.44: Approach RNP Test Lateral Deviation vs. Time, Light Turbulence with increased weight and autopilot tuning

B.6 Comparison of Original and Modified ArcWulf ESR Results

Flight Phase	Test	Original	Weight and Tuning	Change
Departure 2	Climb	702	911	30%
Departure 5	Descent	-629	-717	14%
Departure 8	Straight and Level			
	Altitude Deviation	2	2	0%
	Course Deviation	5.4	1.7	69%
	Airspeed Deviation	3	2.5	17%
Departure 9	Turn	10	6.1	39%
Departure 10	Turn	15	5.9	61%
Departure 11	Turn	15.4	10.6	31%
Departure 12	Turn	14.6	5.4	63%
Departure	RNP Level	0.026	0.106	308%
En Route 1	Climb	701	916	31%
En Route 4	Descent	-701	-685	2%
En Route 7	Straight and Level			
	Altitude Deviation	2	1	50%
	Course Deviation	4.8	1.2	75%
	Airspeed Deviation	2.4	1.9	21%
En Route	RNP Level	0.03	0.115	283%
Arrival 1	Climb	733	926	26%
Arrival 4	Descent	-612	-878	43%
Arrival 7	Straight and Level			
	Altitude Deviation	2	2	0%
	Course Deviation	4.1	1.1	73%
	Airspeed Deviation	3	2	33%
Arrival 8	Turn	11.2	4.6	59%
Arrival 9	Turn	14.2	7.6	46%
Arrival 10	Turn	12.8	6.3	51%
Arrival 11	Turn	13.6	8	41%
Arrival	RNP Level	0.031	0.152	390%
Approach 1	Descent	-765	-628	18%
Approach 5	Turn	10.5	5.1	51%

Approach 6	Turn	13.4	5.8	57%
Approach 7	Final Apporach			
	Altitude Deviation	20	22	10%
	Glide Slope Deviation	0.012	0.017	42%
	Lateral Deviation	0.096	0.084	13%
	Airspeed Deviation	8.9	9.1	2%
Approach 8	Missed Apporach			
	Altitude Deviation	20	23	15%
	Climb Angle Deviation	0.0126	0.159	1162%
	Lateral Deviation	0.053	0.022	58%
	Airspeed Deviation	8.5	10.6	25%
Approach 12	Turn	13.1	5.9	55%
Approach 13	Turn	12.4	5.5	56%
Approach 14	Straight and Level			
	Altitude Deviation	3	3	0%
	Course Deviation	7.3	6.3	14%
	Airspeed Deviation	5.5	5.2	5%
Approach 16	Turn	14.1	6	57%
Approach 17	Turn	16.9	13.3	21%
Approach 18	Final Apporach			
	Altitude Deviation	25	23	8%
	Glide Slope Deviation	0.116	0.012	90%
	Lateral Deviation	0.082	0.053	35%
	Airspeed Deviation	15	10.8	28%
Approach	RNP Level	0.012	0.042	250%