A COMPUTER MODEL TO ESTIMATE BENEFITS OF DATA LINK MANDATES AND REDUCED SEPARATIONS IN THE NORTH ATLANTIC ORGANIZED TRACK SYSTEM

Aswin Kumar Gunnam

Thesis submitted to the Faculty of the Virginia Polytechnic Institute
And State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Civil Engineering

Antonio A. Trani, Chairman
Antoine G. Hobeika
Montasir M. Abbas

November 13, 2012
Blacksburg, Virginia

Keywords: Airspace, North Atlantic, Organized Tracks, Reduced separations, CPDLC, ADS-C, data link.

A Computer Model to Estimate Benefits of Data Link Mandates and Reduced Separations across North Atlantic Organized Track System

Aswin Kumar Gunnam

ABSTRACT

The International Civil Aviation Organization (ICAO) proposed to introduce new operational strategies across the North Atlantic (NAT) airspace. This includes Minimum Navigation Performance Specifications (MNPS) airspace to increase the capacity and efficiency of the North Atlantic Organized Track System (NAT OTS). A numerical integration and simulation model called North Atlantic Simulation and Modeling (NATSAM) is developed to study the effects of these new strategies on NAT system performance. The model is capable of investigating the effects of implementing different operational policies and strategies proposed by ICAO such as Reduced Lateral Separation Minimum (RLatSM), NAT Region Data link mandate (DLM), Reduced Longitudinal Separation Minimum (RLongSM), cruise-climb profiles, variable Mach number profiles, step-climbs and other operational concepts to be studied by the ICAO.

NATSAM models the individual flight performance using the Base of Aircraft Data (BADA) 3.9 model to calculate the flight profiles and fuel burn. The model employs simple heuristics to execute flight track assignment in the organized track system for each scenario. Detailed outputs and also aggregated outputs are provided by the model from which various key performance indicators (KPI) can be derived to assess the performance of the system.

ACKNOWLEDGEMENTS

I would like to first thank Dr. Antonio Trani for giving me the opportunity to conduct this research and for his critical insights and critique. This work would not have been possible without his guidance. He is not only an inspiring person but also a great human being. I would also like to thank my advisory committee Dr. Montasir Abbas and Dr. Antoine Hobeika for their constant support and valuable inputs.

I would like to extend a special thanks to Norma Campos (FAA) for the continuous support, reviews and constructive comments in the course of the project. I wish to extend my gratitude to Thea Graham, John Guilding and David Chin for their support towards this research effort.

I also want to thank Maria Rye, a graduate student in AOE, for developing some of the components which were used in the model. Finally I would like to thank all my colleagues in the Air Transportation Systems Laboratory and my friends for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1. Background

The North Atlantic (NAT) oceanic airspace is one of the most congested airspaces in the world with high volume and high density air traffic. It is the main link between North America – Europe, Caribbean – Europe and also for South America and Europe. This is also a major link for aircraft flying between North America and the Middle East countries. The NAT airspace is a very complex system with responsibilities shared between nine ICAO member states: Canada, the United States, the United Kingdom, Denmark, France, Iceland, Ireland, Norway, and Portugal. The traffic is controlled by seven Oceanic Centers as shown in Figure 1

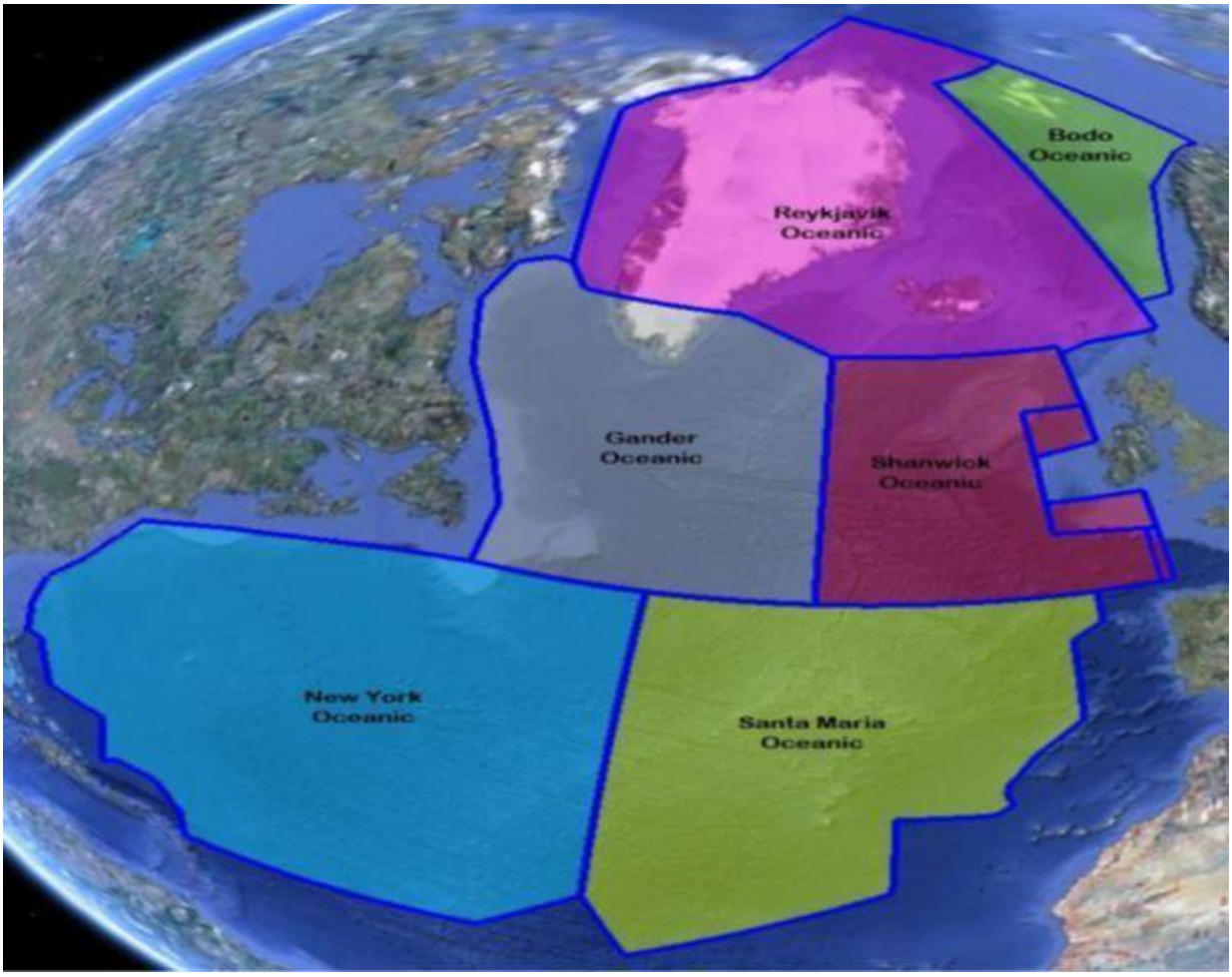


Figure 1: North Atlantic Flight Information Regions (NAT FIR) Source: ICAO document NAT SPG 48 Implementation Plan for the Trial Application of RLatSM

A total of 350,000 flights cross the North Atlantic each year (1) and there are over hundred airlines operating an average of 1300 total flights every day in this region. The composition of NAT traffic is primarily commercial. However there are some International General Aviation (IGA) aircraft operating at higher altitudes in the NAT airspace and there are also military aircraft operations. The total passenger traffic to/from the United States via Atlantic region in the year 2011 was 55.6 million. This traffic is expected to grow to 64.5 million by the year 2015 and 78.9 million by the year 2020 (2). Though transoceanic flights comprise only 4% of total U.S. air carrier operations they consume 26% of total fuel, generate 20 % of total passenger revenue and 49% of total cargo revenue (3). On the other hand the international prices of jet fuel had increased by 50% from year 2005 (1.81 US $/gal) to year 2011 (2.71 US $/gal). These prices are forecasted to be US $ 3.07 per gal in year 2015 and US $ 2.93 per gal in year 2020 (2). Considering the growth of traffic into the future ICAO and member states recognize the need to increase the capacity and efficiency of the NAT airspace. A previous study (4) concludes that NAT operators perceive the inability to fly preferred routes and altitudes as a major operational inefficiency. This survey also reveals that operators see data link as an enabler of improved operational efficiency and also they consider the most reduced separations as the most valuable near term application of this technology (4).

1.2. Current Airspace

1.2.1. Communications, Navigation and Surveillance

Very high frequency (VHF) voice communications are not available for most of the NAT airspace. Pilots have to depend on high frequency (HF) voice communications to communicate with the oceanic centers and the disadvantages associated with HF communications are susceptibility to disruptions, atmospheric effects, ambiguity in accents, frequency congestion and third party relay between pilots and controllers (5). Virtually there is no radar coverage in the NAT Oceanic airspace. Surveillance is accomplished by position reports by pilots transmitted every 10° of longitude (approximately one hour of flight time). Formerly inertial navigation was the prime mode of navigation but currently most of the aircraft switched to global navigation satellite system (GNSS) navigation which is more accurate and reliable.

1.2.2. Organized Tracks and Random Airspace

The airspace over North Atlantic is highly organized and is divided in two sections: (a) Organized Track System (OTS) and (b) Random airspace. OTS consists of well-defined tracks which are published daily by Gander Oceanic Center for eastbound flights and by Shanwick Oceanic Center for westbound flights. The vertical boundaries of OTS are from Flight Level FL 310 (31000 ft.) to Flight Level FL 390 (39000 ft.) both inclusive. The eastbound tracks are aligned with the North Atlantic Jet Stream to maximize the tail wind on the aircraft whereas westbound tracks are located away from the Jet Stream to minimize the headwind. Typically five to seven tracks are published each day for each direction and out of these usually the core tracks are highly wind optimal as shown in Figure 2. In other words the aircraft on the core tracks burn less fuel because of good tail wind for easterly flights and less head wind for westerly flights. Typically, on a given day, approximately 1300 flights travel across the North Atlantic and more than half of them operate in the NAT OTS. The remaining flights operate on non-structured routes called the Random routes. These random routes are uniquely designed based on the flight plan and are outside and away from the OTS by at least 1°. The OTS is valid only during specific times of the day for each direction. All flights flying outside this OTS validity time are considered as random flights. The flights that operate during OTS validity time but fly outside the OTS itself are also considered random flights.

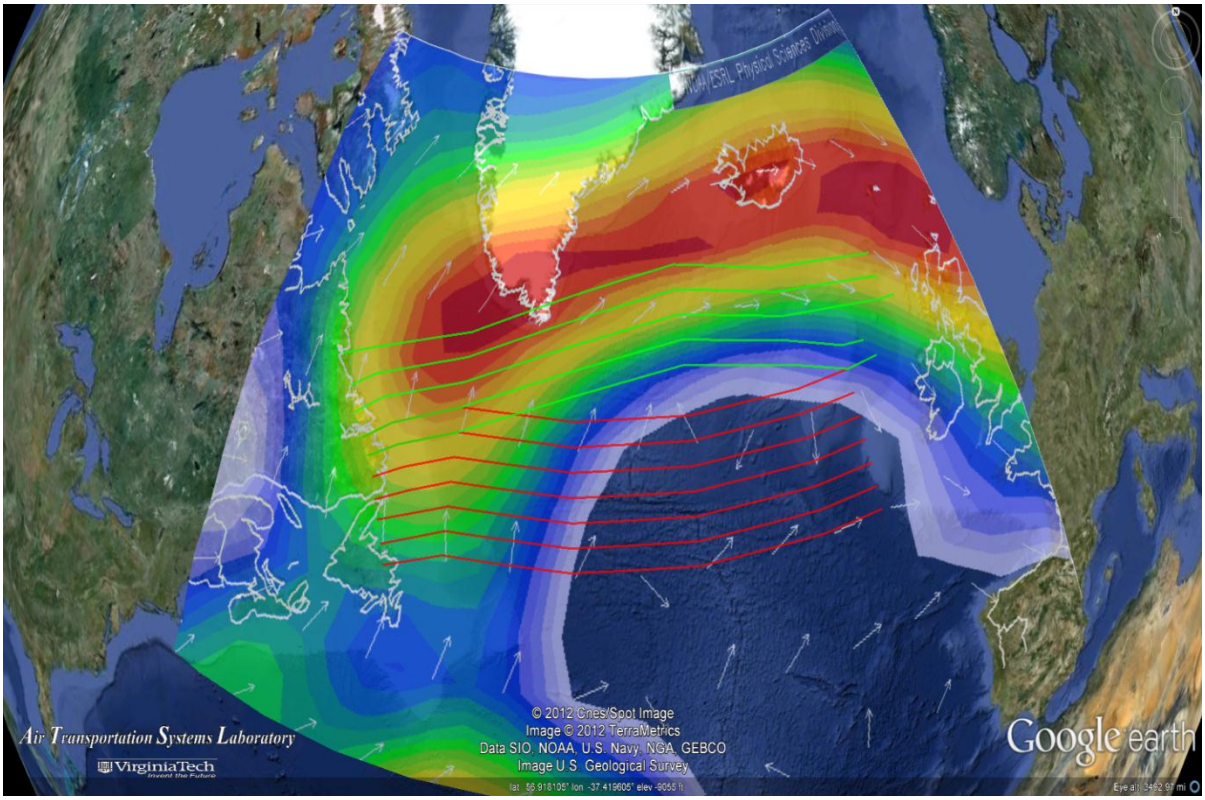


Figure 2: Wind intensity image and NAT Tracks overlaid over Google Earth. Tracks in green are Eastbound and in red are Westbound. Source: NOAA NCAR Reanalysis website (6)

1.2.3. NAT Traffic Patterns

As a result of passenger demand, the difference in time zone and landing restrictions at the destination airports most of the NAT traffic is characterized by two distinct traffic flows during a 24-hour period. The majority of the westbound traffic leaves Europe in the late morning and reaches North American East Coast in the afternoon-local time given the time difference. The majority of the eastbound traffic leaves North America in the evening and reaches Europe in the morning-local time. The peak of eastbound traffic flow crosses 30W longitude at 0400 Coordinated Universal Time (UTC) and therefore the eastbound OTS is valid only from 0100 UTC to 0800 UTC so that the peak is contained in the centre. Similarly the peak of westbound flow crosses 30W at 1500 UTC and therefore the westbound OTS is valid only from 1130 UTC to 1900 UTC. A temporal frequency distribution of departure times of flights operating in the NAT for a 24-hour duration is shown in Figure 3. In the figure the departures from 0 hours to around 5 hours are predominantly eastbound flights departing the North American airports at midnight. The departures from 5 hours to 19 hours are predominantly westbound flights departing

European airports in the early morning and the departures from 19 hours to 24 hours are again eastbound flights. Note that the directions described represent majority of the flow and there would be few cargo, general aviation and military flights which do not necessarily follow the commercial flight schedules and fly in the opposite directions.

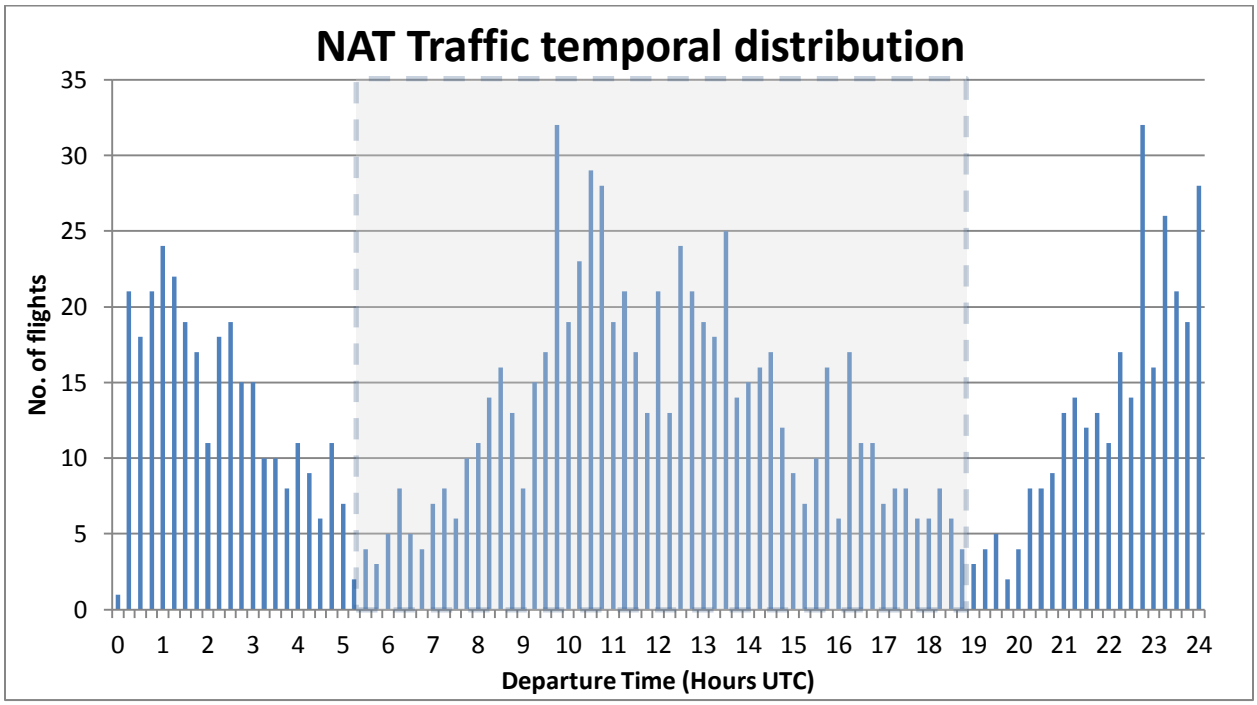


Figure 3: Temporal frequency distribution of flights entering NAT boundary.

Usually before filing a flight plan, flight dispatchers select either a track or a random route using some kind of flight planning tools. Depending on the forecasted wind conditions at the time of flight they optimize the altitude, speed, flight path and select a NAT track or random route. Flights compete for the core tracks which are the most wind efficient tracks and there is no guarantee at the time of NAT entry that the flights will be assigned the exact track-flight level combination they requested in the flight plan.

1.2.4. NAT Wind

An introduction of NAT winds and the Jet Stream is essential to comprehend the benefits mechanism of the OTS. The winds over the ocean are strong particularly at high altitudes including the common flight levels used by NAT flights. The North Atlantic Jet Stream is a significant atmospheric phenomenon occurring over the North Atlantic Ocean. It is a fast flowing narrow air current that travels eastwards from the North America to Europe. The design of NAT OTS tracks mainly depends on the location and strength of the Jet Stream and the design principle is to optimize the routes for flights while taking the

Jet Stream into consideration. The Jet Stream is highly energetic and dynamic with frequent changes from day to day. Therefore OTS is designed each day and eastbound tracks are placed as close as possible to the Jet Stream while the westbound tracks are placed as far as possible. The Jet Stream can influence the travel times of NAT flights by as much as 60 min (7). The location, direction and strength of the Jet Stream are shown in Figure 4 for a day in January.

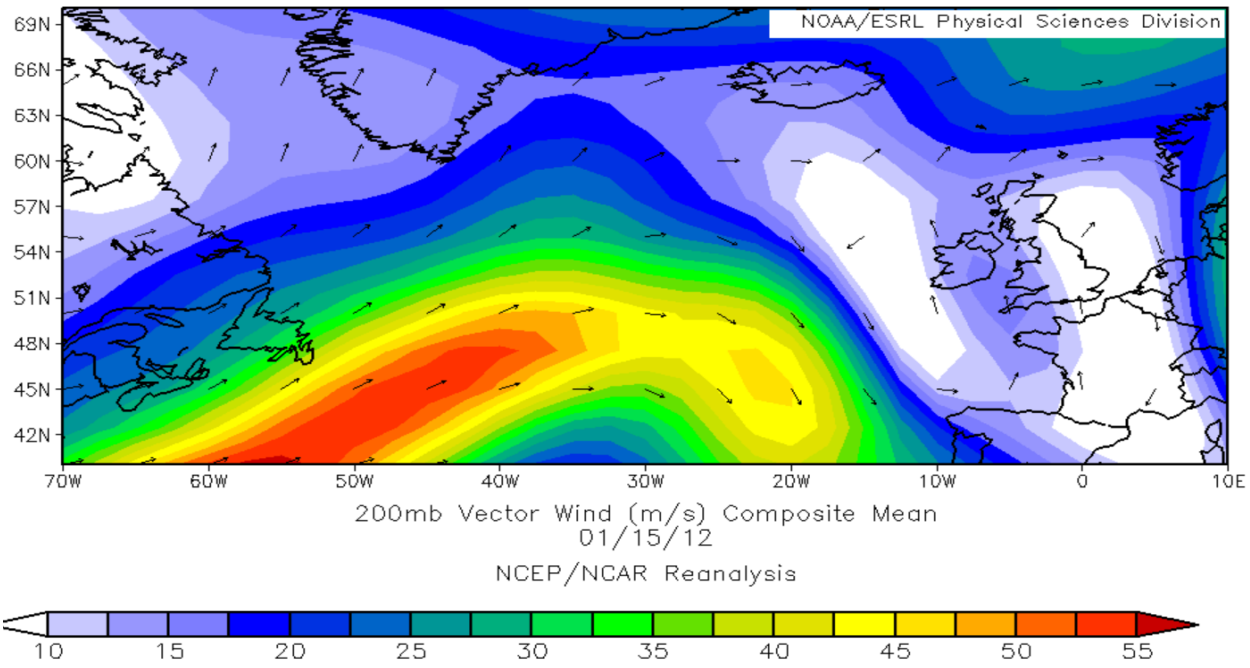


Figure 4: North Atlantic Jet Stream at 39000 ft. Source: NOAA NCAR Reanalysis website (6)

1.3. New Concepts of Operation

To cater for the requests of airlines for more efficient flight profiles, to increase the capacity of the NAT system and most importantly to increase the safety levels in NAT airspace, ICAO intends to introduce new concepts of operations (CONOPS). These concepts require aircraft to be equipped with certain advanced avionics. The current NAT fleet has a mix of equipped and non-equipped aircraft and hence to accommodate both types of aircraft, these concepts would be introduced gradually in phases. The CONOPS to be implemented in NAT are discussed below.

1.3.1. NAT Region Data Link Mandate (DLM)

The NAT systems planning group (SPG) at their 43 meeting (2007) has realized that large height deviations (LHD) and gross navigational errors (lateral) were the reason for not meeting the target level of safety in the NAT. NAT SPG 44 (2008) concluded that Automatic Dependent Surveillance (ADS-C)

conformance monitoring could mitigate the occurrence of navigational errors and in 2009 NAT SPG 45 identified the need to mandate the usage of data link in the NAT airspace. In the later meetings NAT SPG defined the lateral and vertical limits of the exclusionary airspace where DLM would be applicable. Only fully equipped aircraft would be allowed to operate in the exclusionary airspace. The definition of different equipage levels is given below:

- Fully Equipped (hence referred to as equipped)
    - Equipped with FANS-1/A ADS-C and CPDLC certified according to requirements in RTCA DO-258A/EUROCAE ED-100A or equivalent.
    - Pre-requisite systems:
        - SATCOM
        - GNSS\GPS
        - FMC upgrade or FMC that will support FANS-1/A ADS-C and CPDLC
- Partially Equipped (hence referred to as non-equipped)
    - Lacks one or several of the systems specified above
- Unequipped (hence referred to as non-equipped)
    - Lacks all of the systems specified above

Partially equipped and unequipped aircraft are considered to be non-compliant with DLM requirements and they need to be retrofitted to meet the requirements of the mandate.

ADS-C is an advanced avionics system and serves as a surveillance tool by sending frequent aircraft position reports to the air traffic management facilities. Also when reports do not match with the flight plan, ADS-C shall provide controllers with alerts in the form of contracts. Other alerts are also issued in the events of:

- Lateral deviation event (LDE) with a lateral deviation threshold of 9.3 km (5 NM) or less.
- Level range deviation event (LRDE) with a vertical deviation threshold of 90 m (300 ft.) or less.
- Waypoint change event (WCE) at compulsory reporting points.

CPDLC provides efficient communication in data form (typically text messages) which are reliable, fast and are less prone to misinterpretations by pilots and controllers. Usage of CPDLC also off loads the majority of communicatis using VHF or HF channels and eliminates the dependence on these channels.

Implementation plans for NAT DLM and extents of exclusionary airspace are given below:

- **Phase 1 - From 7 February 2013**
  - In vertical plane from 36000 ft. to 39000 ft. both inclusive
  - In horizontal plane, no more than two tracks within the NAT OTS designated as core tracks and identified in NAT track message
  - Aircraft need to be equipped to operate in mandated airspace
- **Phase 2 - From 5 February 2015**, in specified portions of the NAT Minimum Navigation Specification (MNPS) Airspace
  - In vertical plane from 36000 ft. to 39000 ft. both inclusive
  - In horizontal plane, across all the OTS tracks and identified in NAT track message

During the DLM implementation phase only the equipped aircraft would be allowed to operate in exclusionary airspace and all non-equipped aircraft have to operate below or 1° laterally away from the exclusionary airspace.

1.3.2. Reduced Lateral Separation Minimum (RLatSM)

Airlines and other customers of NAT are requesting more efficient flight profiles and routes in order to reduce the operating cost and show a return on their investment on advanced avionics (8). Introducing reduced lateral separation can provide efficient profiles and routes and also has the potential to increase the capacity of airspace. RLatSM can be implemented in following configurations:

1) New tracks introduced exactly in between the existing 1° spaced tracks, thus reducing the spacing to ½° as shown in Figure 6. This creates more tracks (one less than twice the actual) and increases the capacity but occupies the same airspace as the current configuration.

2) By maintaining the current number of tracks but reducing the spacing to ½° as shown in Figure 7. This configuration has the same capacity of the current configuration but it frees up a large portion of airspace to random and opposite direction flights.

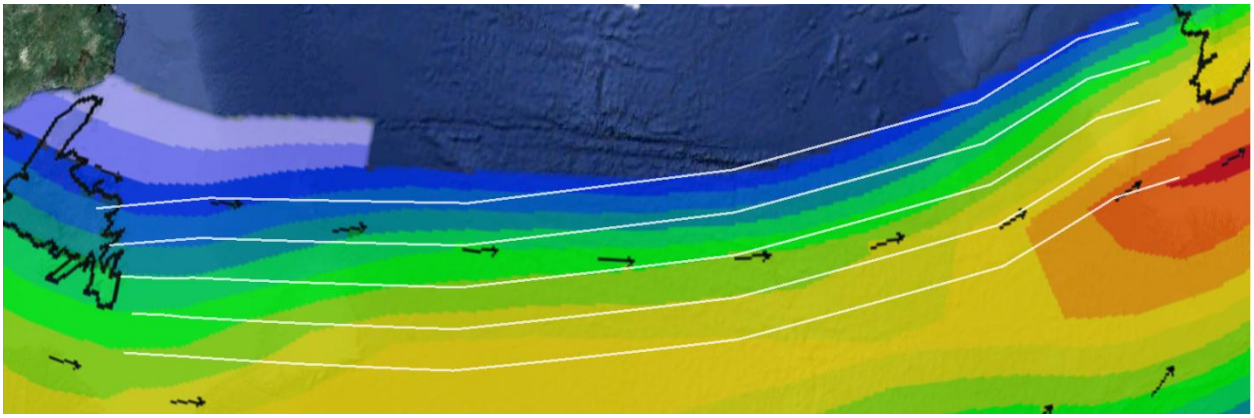


Figure 5: Current track system in NAT. Eastbound tracks are shown above. Source: NOAA NCAR Reanalysis website (6)

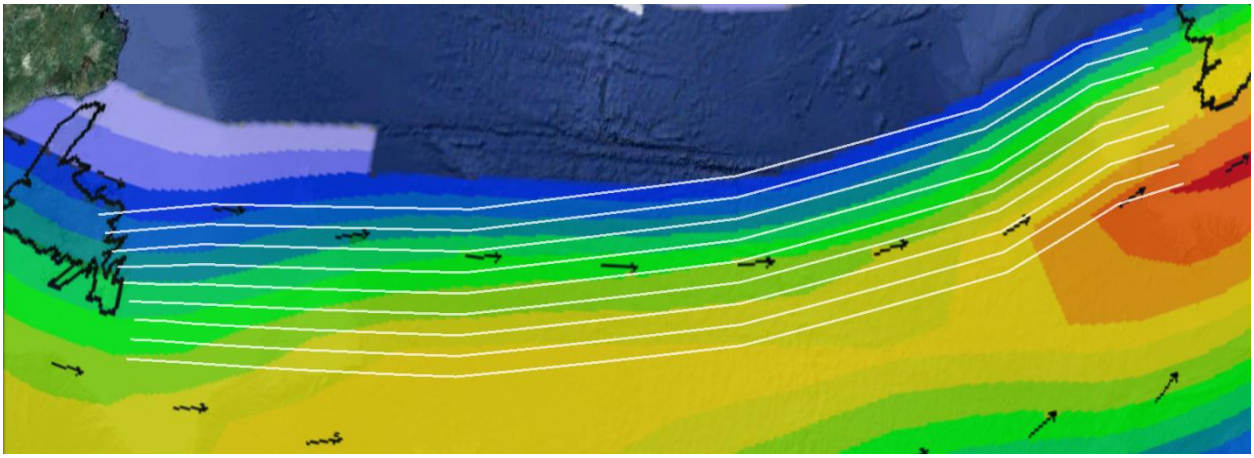


Figure 6: Increased number of tracks occupying same amount of airspace as the actual system. Source: NOAA NCAR Reanalysis website (6)

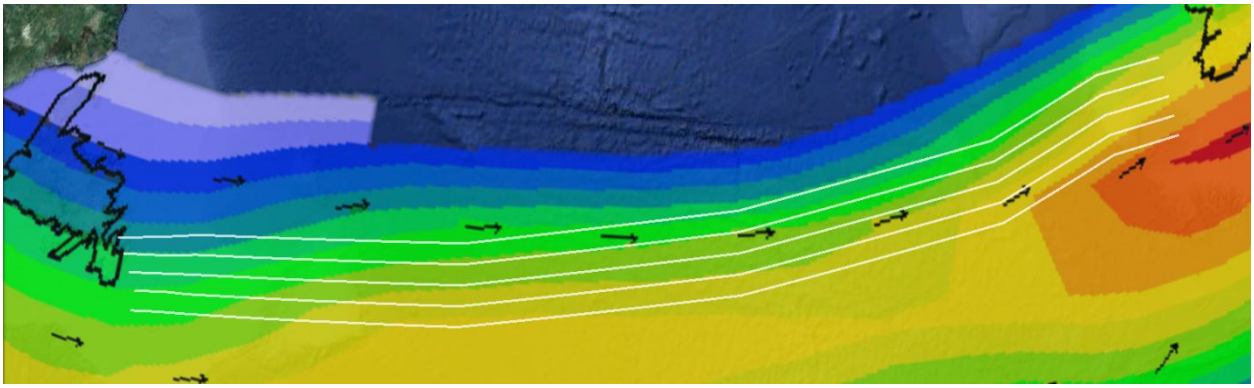


Figure 7: Same number of tracks occupying less airspace than the actual system. Source: NOAA NCAR Reanalysis website (6)

Upon careful evaluation of the two configurations stated above and several other configurations, ICAO decided to implement the second configuration i.e., maintain the same number of tracks but reduce the spacing to ½° and free up airspace to random and opposite direction flights. This decision is in-line with the free-flight concept envisioned by many agencies like ICAO, FAA and Eurocontrol.

Figure 8 is a cross section of Figure 5 showing how the track/flight levels are placed with respect to the winds in NAT. The intensity of wind shown is a composite mean from 310°E to 0°E and the latitudes of each track are the weighted mean of all waypoints for that track. Figure 9 is a cross section of Figure 7 and it shows the how tracks can be placed in more favouring winds by reducing the separation. Note that in reduced separations, represented by Figure 7 and Figure 9, some additional airspace is available for random flights which is otherwise occupied by regular tracks in 1° separations. The benefits of implementing RLatSM are providing more efficient flight profiles and more random airspace. Due to the proposed reduced lateral separation, in case of easterly tracks, more number of tracks can be placed closer to the core of the Jet Stream as shown in Figure 9. In case of westerly tracks more number of tracks can be placed away from the Jet Stream. Flights have more flexibility in choosing random tracks in case of RLatSM because of the availability of more free airspace as shown in Figure 7 and Figure 9.

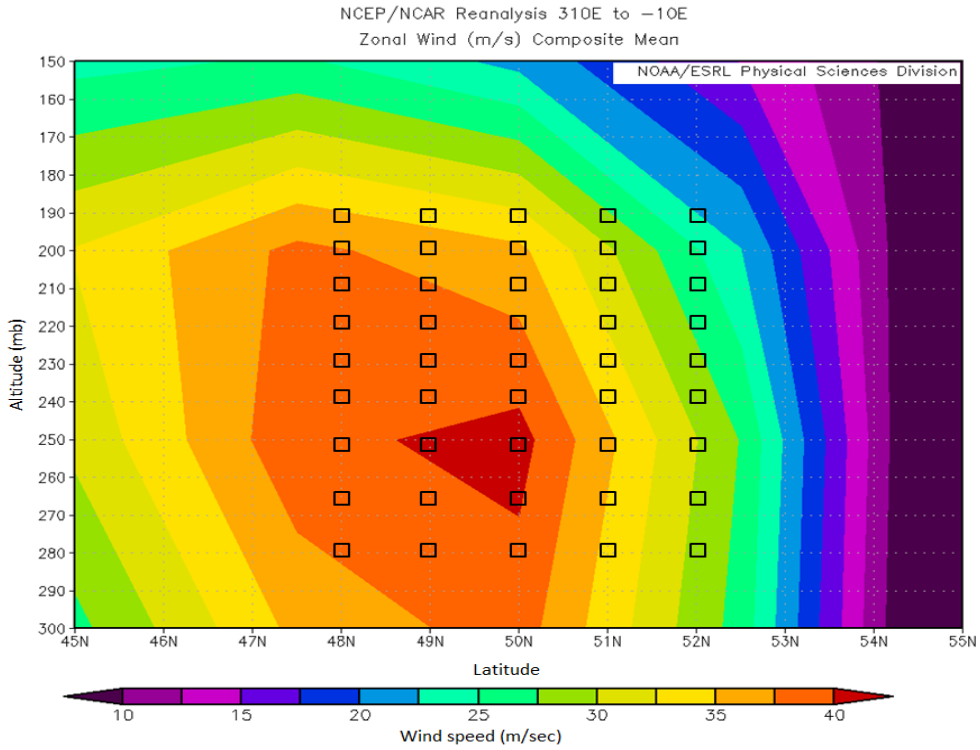


Figure 8: Easterly tracks/flight level's location and cross section of wind intensity in NAT.

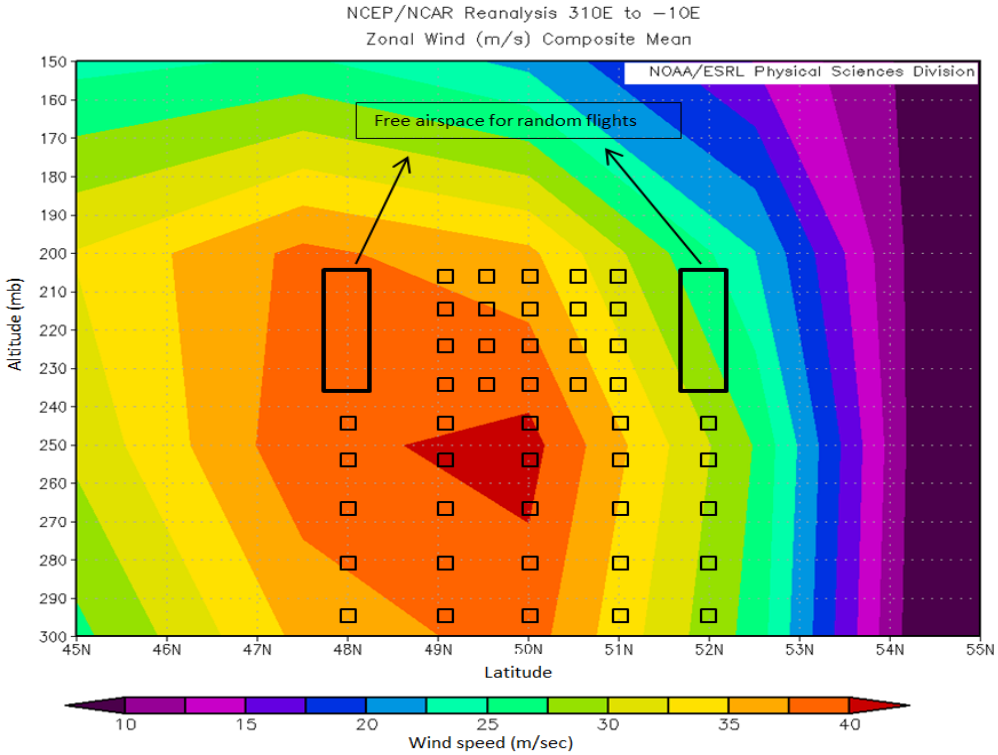


Figure 9: Placing tracks optimally in RLatSM. Source: NOAA NCAR Re-analysis website

The requirements for the aircraft to operate in RLatSM tracks are appropriate RNP approval (usually RNP-4), ADS-C and CPDLC. These requirements are same as the DLM and therefore only equipped aircraft are allowed to operate on these ½° spaced tracks.

The implementation plan and extent of RLatSM are defined below as per ICAO SPG 48 draft implementation plan of RLatSM in NAT Region.

- Phase 1 – 2015 – introduce 25 NM lateral separation by implementing ½° spacing between the two core tracks within the vertical limits applicable to the airspace associated with the NAT Region Data Link Mandate; only aircraft with the appropriate RNP approval, ADS-C and CPDLC would be permitted to operate on the ½° spaced tracks
- Phase 2 – To Be Determined – introduce 25 NM lateral separation by implementing ½° spacing through the entire NAT Organized Track System (OTS), within the vertical limits applicable to the airspace associated with the NAT Region Data Link Mandate; only aircraft with the appropriate RNP approval, ADS-C and CPDLC would be permitted to operate on the ½° spaced tracks.
- Phase 3 – To Be Determined – introduce 25 NM lateral separation throughout the entire NAT Region, including converging and intersecting track situations, within the vertical limits applicable to the airspace associated with the NAT Region Data Link Mandate. The application of the reduced separation standard between targets of opportunity should be permissible in any part of the NAT Region outside the OTS (mixed mode operations).

For the purpose of this study only Phases 1 and 2 are considered. Phase 2 is assumed to be implemented in the year 2017 (also assumed by ICAO and FAA).

1.3.3. Reduced Longitudinal Separation Minimum (RLongSM)

NAT customers have indicated that the ability to execute step climbs enables more fuel efficient flight profiles (9). The current longitudinal separation in NAT airspace is 10 minutes in-trail using a constant Mach number technique and the maximum number of flights that can be injected into a track/flight level combination is 6/hour (although in practice fewer flights are assigned due to variations in speeds of successive aircraft pairs). Starting in March 2011 NAVCANADA and NATS UK have already started 5 minute in trail separations on a trial basis for only eligible aircraft. The implementation of RLongSM has two benefits:

1 Since the in-trail separation would be reduced to 5 minutes the maximum number of aircraft that can be injected in any given track/flight level combination increases from 6 per hour to 11 per hour. The capacity nearly doubled and therefore higher density of traffic can operate at optimum altitudes.

2 The probability of performing a step-climb in the NAT would be increased due to the increased targets of opportunity.

Both the above benefits derive reduced fuel consumption and reduced emissions. The requirements for RLongSM are the same as those of RLatSM which in turn are same as that of DLM.

RLongSM trail is under progress and after thorough analysis of the trail results a decision would be taken on making RLongSM operational in NAT. For this study the implementation dates of RLatSM have been assumed to be the same as those for RLongSM.

CHAPTER 2 LITERATURE REVIEW

The objective of this study is to estimate the benefits of implementing new Concepts of Operations (CONOPS) in the NAT region. The primary benefits to be estimated are the fuel savings and travel time savings for all flights operating in the NAT airspace. Therefore understanding aircraft fuel burn mechanism and aircraft performance characteristics are of paramount important. In this section we discuss similar studies aimed at quantifying the benefits and system performance in the NAT region.

2.1 BADA 3.9

The Aircraft Performance Model (APM) employed in the model is the Eurocontrol Base of Aircraft Data (BADA) version 3.9 (10). This model is intended for use in aircraft trajectory simulations, fuel calculations and environmental studies. BADA is widely used directly in many aviation modeling tools such as TAAM, EDMS, AEDT, AEST, NIRS, NST and indirectly in tools like INM. This is a widely accepted model among the aviation industry and most of the state aviation agencies. The BADA 3.9 model is estimated to deliver results with an error of below 5% (11) for the overall fuel flow and below 3% (12) for the cruise fuel flow. The important features of the BADA model are discussed here.

BADA is the most exhaustive APM publicly available in the industry. This is a collection of data files in ASCII format which contain the operational performance parameters, airlines procedures, aerodynamic and aircraft related parameters and pre-calculated performance summary tables for individual aircraft. Each aircraft type has a set of four text files: (1) Operational performance file (OPF), (2) Airline procedures file (APF), (3) Performance table file (PTF) and (4) Performance table data (PTD). The operational performance parameters and airlines procedures parameters which form the core of BADA are derived from operational performance models and airlines procedures models respectively. A brief introduction to these models, text files and the respective parameters is given below.

2.1.1 Operations Performance Model and OPF

The International standard atmosphere (ISA) model provides information on pressure, density, speed of sound and temperature as a function of altitude. Aircraft aerodynamic behavior varies with atmospheric conditions. BADA adopted the ICAO ISA model described in the ICAO manual of standard atmosphere: ICAO Document No. 7488 in all calculations related to atmosphere.

Aircraft fuel consumption in the BADA model is derived from a Total Energy Model (TEM) based on the basic aerodynamic equations of motion. The Total-Energy Model equates the rate of work done by forces acting on the aircraft to the rate of increase in potential and kinetic energy. This can be written in equation [1]:

$$(Thr - D) * V_{TAS} = mg_0 \frac{dh}{dt} + mV_{TAS} \frac{dV_{TAS}}{dt} \quad \text{----------} \quad [1]$$

Where:

Thr = thrust acting parallel to the aircraft velocity vector [Newtons];

D = aerodynamic drag [Newtons];

m = aircraft mass [kilograms];

h = geodetic altitude [m];

g_0 = gravitational acceleration [9.80665 m/s2];

V_{TAS} = true airspeed [m/s];

Most of the operational performance parameters are derived from the total energy model and these parameters are listed in the operations performance file (OPF). Each aircraft type has an OPF which is an ASCII text file with 7 sections listed in Table 1: Parameters in the OPF file

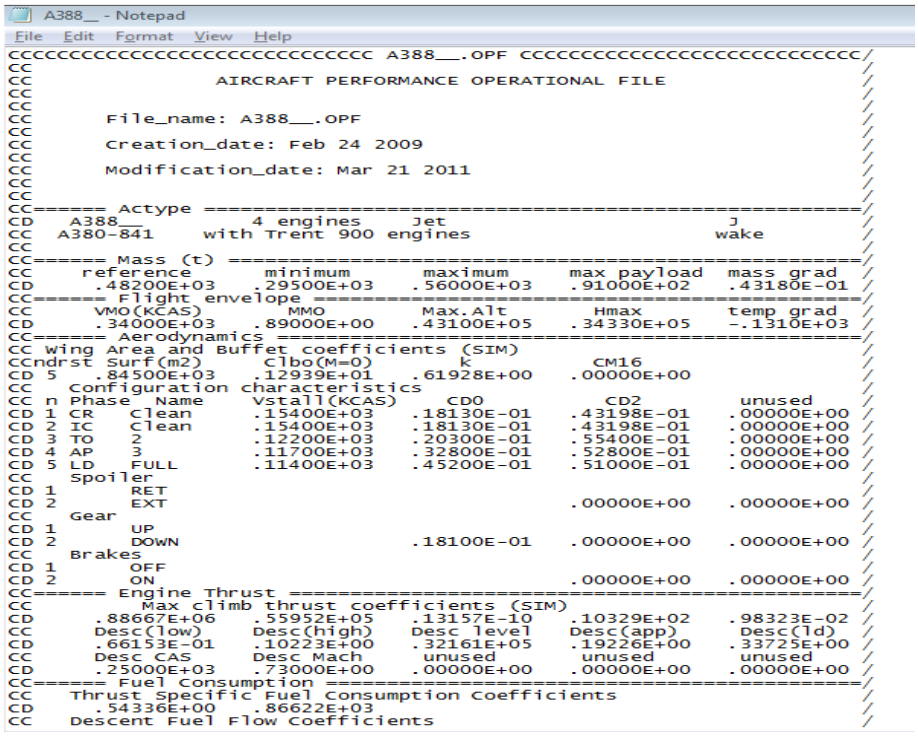
. The OPF contains the core parameters used in modeling a flight.

Section	**Parameters listed in the section**
Actype (Aircraft type)	Number of engines, engine type and wake category
Mass	Min mass, max mass, reference mass, max payload mass and mass gradient
Flight envelope	Max operating speed, max operational Mach number,max operating altitude, max altitude at MTOW and temp gradient
Aerodynamics	Parasitic and induced drag coefficients for cruise, approach and landing, reference wing surface area, stall speeds for takeoff, climb, cruise, approach and landing, buffeting gradient and buffet onset lift coefficient
Engine thrust	Different types of climb, descent, approach, landing and temperature thrust coefficients, transition altitude, descent speed and descent Mach number
Fuel flow	Different types of thrust specific fuel coefficients, descent fuel flow coefficients and cruise fuel flow correction coefficient

Ground movements	Takeoff and landing lengths, wingspan and aircraft length

Table 1: Parameters in the OPF file

A screen capture of the OPF forAirbus A380-800 is shown in Figure 10.



```
A388__ - Notepad
File  Edit  Format  View  Help
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC A388__.OPF CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC/
CC                                                                          /
CC                  AIRCRAFT PERFORMANCE OPERATIONAL FILE                   /
CC                                                                          /
CC                                                                          /
CC       File_name: A388__.OPF                                              /
CC                                                                          /
CC       Creation_date: Feb 24 2009                                         /
CC                                                                          /
CC       Modification_date: Mar 21 2011                                     /
CC                                                                          /
CC                                                                          /
CC====== Actype ============================================================/
CD   A388__          4 engines    Jet                          J            /
CC  A380-841     with Trent 900 engines                      wake           /
CC                                                                          /
CC====== Mass (t) ==========================================================/
CC     reference       minimum       maximum     max payload  mass grad    /
CD      .48200E+03    .29500E+03    .56000E+03    .91000E+02    .43180E-01 /
CC====== Flight envelope ===================================================/
CC      VMO(KCAS)        MMO         Max.Alt        Hmax        temp grad    /
CD      .34000E+03    .89000E+00    .43100E+05    .34330E+05    -.1310E+03 /
CC====== Aerodynamics ======================================================/
CC Wing Area and Buffet coefficients (SIM)                                  /
CCndrst Surf(m2)      Clbo(M=0)         k           CM16                    /
CD 5    .84500E+03    .12939E+01    .61928E+00    .00000E+00                /
CC   Configuration characteristics                                          /
CC n Phase  Name     Vstall(KCAS)     CD0           CD2          unused     /
CD 1 CR    Clean      .15400E+03    .18130E-01    .43198E-01    .00000E+00 /
CD 2 IC    Clean      .15400E+03    .18130E-01    .43198E-01    .00000E+00 /
CD 3 TO    2          .12200E+03    .20300E-01    .55400E-01    .00000E+00 /
CD 4 AP    3          .11700E+03    .32800E-01    .52800E-01    .00000E+00 /
CD 5 LD    FULL       .11400E+03    .45200E-01    .51000E-01    .00000E+00 /
CC   Spoiler                                                                /
CD 1       RET                                                              /
CD 2       EXT                                   .00000E+00    .00000E+00 /
CC   Gear                                                                   /
CD 1       UP                                                               /
CD 2       DOWN                     .18100E-01    .00000E+00    .00000E+00 /
CC   Brakes                                                                 /
CD 1       OFF                                                              /
CD 2       ON                                    .00000E+00    .00000E+00 /
CC====== Engine Thrust =====================================================/
CC          Max climb thrust coefficients (SIM)                             /
CD      .88667E+06    .55952E+05    .13157E-10    .10329E+02    .98323E-02 /
CC      Desc(low)     Desc(high)    Desc level    Desc(app)     Desc(ld)   /
CD      .66153E-01    .10223E+00    .32161E+05    .19226E+00    .33725E+00 /
CC      Desc CAS      Desc Mach      unused        unused        unused     /
CD      .25000E+03    .73000E+00    .00000E+00    .00000E+00    .00000E+00 /
CC====== Fuel Consumption ==================================================/
CC    Thrust Specific Fuel Consumption Coefficients                         /
CD      .54336E+00    .86622E+03                                            /
CC    Descent Fuel Flow Coefficients                                        /
```

Figure 10: Operational Performance File for Airbus A380-800 in BADA 3.9

If the operating procedures of a flight are known, in other words boundary conditions of a flight phase such as the speed schedules, operating altitudes, takeoff mass and flight configurations are known, then fuel consumption can be calculated for any phase of flight using the aerodynamic equations and parameters listed in the OPF. For example if the origin airport altitude, takeoff mass, Top of Climb (TOC) and the climb speed profile are known then a full climb profile can be calculated, however these variables differ from flight to flight as a function of airspace, airlines, local conditions and regulations.

2.1.2 Airline Procedure File (APF)

To overcome the variability in operating procedures described in Section 2.1.1 BADA provides a set of standardized operating procedures for climb, cruise and descent phases for each aircraft type to facilitate simulation of nominal aircraft operations. These standard procedures are defined based on the

aircraft manufacturer reference performance data and operational data. The user can modify some of these standard procedures to suit more specific simulation needs if the procedures under study are different. The standard procedures are listed in the airline procedure file which contains speed schedules for climb, cruise and descent phases of flight for low, average and high takeoff masses. A sample of the APF for the Airbus A380-800 is shown in Figure 11.

A388_ - Notepad

File Edit Format View Help

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC A388__.APF CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC/
CC                                                                                             /
CC                  AIRLINES PROCEDURES FILE                                                   /
CC                                                                                             /
CC       File_name: A388__.APF                                                                 /
CC                                                                                             /
CC       Creation_date: Feb 24 2009                                                            /
CC                                                                                             /
CC       Modification_date: Mar 05 2009                                                        /
CC                                                                                             /
CC                                                                                             /
CC                                                                                             /
CC    LO= 295.00 to ---.--   /  AV= ---.-- to ---.--   /  HI= ---.-- to 560.00                 /
CC                                                                                             /
CC=============================================================================================/
CC   COM CO     Company name ------climb-------   --cruise--   -----descent------  --approach-  model- /
CC                      mass  lo  hi               lo  hi          hi  lo          (unused)         /
CC     version engines   ma   cas cas mc xxxx xx   cas cas mc   mc cas cas xxxx xx  xxx xxx xxx  opf___ /
CC===:=======:=======::==::===:===:==:====:==::===:===:==::==:===:===:====:==::===:===:===::======:/
CD   *** **     Default Company                                                                /
CD     841               LO   320 320 85           250 320 85   85 300 300            0   0   0  A388__ /
CD     841               AV   320 320 85           250 320 85   85 300 300            0   0   0  A388__ /
CD     841               HI   320 320 85           250 320 85   85 300 300            0   0   0  A388__ /
CC===:=======:=======::==::===:===:==:====:==::===:===:==::==:===:===:====:==::===:===:===::======:/
CC//////////////////////////////////////////THE END//////////////////////////////////////////
```

Figure 11: Airline Performance File for Airbus A380-800 in BADA 3.9

2.1.3 Performance Table File (PTF)

The performance table file specifies the climb, cruise and descent performance at different flight levels. The PTF lists the true air speed, rate of climb/descent, fuel flow for climb and descent. It only lists true air speed and cruise fuel flow for all possible flight levels. The data present in the PTF is generated using the OPF and APF files. A sample PTF is shown in Figure 12. The values in the PTF can be used to calculate the fuel consumption for all phases of flight without actually integrating the equations of motion over time. However, the results will only apply to three aircraft conditions prescribed.

A388__ - Notepad

File Edit Format View Help

```
BADA PERFORMANCE FILE                                        Mar 30 2011

AC/Type: A388__
                                Source OPF File:             Mar 21 2011
                                Source APF file:             Mar 05 2009

 Speeds:   CAS(LO/HI)  Mach    Mass Levels [kg]         Temperature:  ISA
 climb   - 250/320     0.85    low      -  354000
 cruise  - 250/320     0.85    nominal  -  482000       Max Alt. [ft]:  43100
 descent - 250/300     0.85    high     -  560000
```

FL	CRUISE				CLIMB					DESCENT		
	TAS [kts]	fuel [kg/min] lo	nom	hi	TAS [kts]	ROCD [fpm] lo	nom	hi	fuel [kg/min] nom	TAS [kts]	ROCD [fpm] nom	fuel [kg/min] nom
0					164	2288	1804	1569	572.8	153	825	191.2
5					165	2275	1790	1554	568.3	154	839	189.7
10					166	2262	1776	1539	563.8	160	851	189.1
15					172	2359	1850	1604	562.1	172	943	108.0
20					174	2345	1835	1588	557.7	204	960	110.4
30	230	132.7	177.6	211.7	197	2720	2122	1842	559.7	230	970	61.6
40	233	133.0	178.2	212.4	231	3187	2472	2147	567.0	233	987	60.7
60	272	149.4	185.7	213.3	272	3696	2671	2222	565.6	272	1147	59.0
80	280	150.2	186.9	214.8	280	3578	2570	2127	547.1	280	1184	57.3
100	289	151.1	188.2	216.4	368	3588	2643	2237	564.6	345	1609	55.5
120	297	152.0	189.5	218.0	379	3410	2496	2101	545.2	356	1650	53.8
140	390	217.1	242.3	261.4	390	3224	2342	1959	525.5	366	1691	52.1
160	401	218.0	243.6	263.0	401	3030	2183	1811	505.7	377	1732	50.4

Figure 12: Performance Table File for Airbus A380-800 in BADA 3.9

2.1.4 Performance Table Data (PTD)

In addition to the data provided in the PTF, more detailed data on high, medium and low mass climbs and data on medium mass descents are given in the PTD file. This is to enable the users to calculate the fuel consumption quickly with less computational effort. However the results obtained from using these files are also of low fidelity. An excerpt from the PTD file is shown in Figure 13.

A388__ - Notepad

File Edit Format View Help

```
BADA PERFORMANCE FILE RESULTS
=============================
=============================

Low mass CLIMBS
===============

 FL[-] T[K] p[Pa] rho[kg/m3] a[m/s] TAS[kt] CAS[kt]    M[-] mass[kg] Thrust[N] Drag[N] Fuel[kgm] ESF[-] ROC[fpm] TDC[N]   PWC[-]
     0 288 101325   1.225      340   140.92   140.92    0.21 354000    886670    240710   560.2    0.98    2288    570639    0.88
     5 287  99508   1.207      340   141.94   140.92    0.21 354000    878749    240739   555.7    0.97    2275    563616    0.88
    10 286  97717   1.190      339   142.97   140.92    0.22 354000    870835    240769   551.3    0.97    2262    556598    0.88
    15 285  95952   1.172      339   149.12   145.92    0.23 354000    862926    231464   549.6    0.97    2359    557831    0.88
    20 284  94213   1.155      338   150.21   145.92    0.23 354000    855023    231493   545.2    0.97    2345    550823    0.88
    30 282  90812   1.121      337   173.29   165.92    0.26 354000    839234    206552   547.2    0.96    2720    558909    0.88
-----------------------------------------------------------------------------------------------------------------------------------

Medium mass CLIMBS
==================

 FL[-] T[K] p[Pa] rho[kg/m3] a[m/s] TAS[kt] CAS[kt]    M[-] mass[kg] Thrust[N] Drag[N] Fuel[kgm] ESF[-] ROC[fpm] TDC[N]   PWC[-]
     0 288 101325   1.225      340   163.60   163.60    0.25 482000    886670    329731   572.8    0.97    1804    532349    0.96
     5 287  99508   1.207      340   164.78   163.60    0.25 482000    878749    329785   568.3    0.97    1790    524727    0.96
    10 286  97717   1.190      339   165.97   163.60    0.25 482000    870835    329841   563.8    0.97    1776    517109    0.96
    15 285  95952   1.172      339   172.28   168.60    0.26 482000    862926    318631   562.1    0.96    1850    520263    0.96
    20 284  94213   1.155      338   173.54   168.60    0.26 482000    855023    318686   557.7    0.96    1835    512657    0.96
    30 282  90812   1.121      337   196.93   188.60    0.30 482000    839234    286681   559.7    0.95    2122    528157    0.96

-----------------------------------------------------------------------------------------------------------------------------------

High mass CLIMBS
================

 FL[-] T[K] p[Pa] rho[kg/m3] a[m/s] TAS[kt] CAS[kt]    M[-] mass[kg] Thrust[N] Drag[N] Fuel[kgm] ESF[-] ROC[fpm] TDC[N]   PWC[-]
     0 288 101325   1.225      340   175.95   175.95    0.27 560000    886670    384105   579.6    0.96    1569    502565    1.00
     5 287  99508   1.207      340   177.22   175.95    0.27 560000    878749    384178   575.2    0.96    1554    494571    1.00
    10 286  97717   1.190      339   178.50   175.95    0.27 560000    870835    384252   570.7    0.96    1539    486582    1.00
    15 285  95952   1.172      339   184.89   180.95    0.28 560000    862926    372007   569.0    0.96    1604    490918    1.00
    20 284  94213   1.155      338   186.23   180.95    0.28 560000    855023    372082   564.5    0.96    1588    482941    1.00
    30 282  90812   1.121      337   209.79   200.95    0.32 560000    839234    336169   566.4    0.95    1842    503065    1.00
-----------------------------------------------------------------------------------------------------------------------------------

Medium mass DESCENTS
====================

 FL[-] T[K] p[Pa] rho[kg/m3] a[m/s] TAS[kt] CAS[kt]    M[-] mass[kg] Thrust[N] Drag[N] Fuel[kgm] ESF[-] ROD[fpm] TDC[N] gammaTAS[deg]
     0 288 101325   1.225      340   153.20   153.20    0.23 482000    299029    557943   191.2    0.97     825  -258913     -3.05
     5 287  99508   1.207      340   154.31   153.20    0.23 482000    296358    557979   189.7    0.97     839  -261621     -3.08
    10 286  97717   1.190      339   160.50   158.20    0.24 482000    293689    549452   189.1    0.97     851  -255763     -3.00
    15 285  95952   1.172      339   171.88   168.20    0.26 482000    165906    431686   108.0    0.96     943  -265780     -3.11
    20 284  94213   1.155      338   203.96   198.20    0.31 482000    164387    395803   110.4    0.95     960  -231416     -2.66
    30 282  90812   1.121      337   229.62   220.00    0.35 482000     55518    265858    61.6    0.94     970  -210340     -2.39

TDC stands for (Thrust - Drag) * Cred
```

Figure 13: Performance Table Data file for Airbus A380-800 in BADA 3.9.

2.2 Aircraft Performance

Knowledge of the aircraft performance and its aerodynamic characteristics is required to model the fuel consumption and travel times. Any flight consists of three phases namely climb, cruise and descent with most of the flights consisting these three phases more than once. In all of these phases the aircraft is subjected to weight, lift, thrust and drag as described in the following sections. The weight includes empty operating weight, payload and weight of fuel. As the aircraft spends time in the air it burns fuel and the weight is decreased over time. Thus the lift required to sustain flight is also reduced.

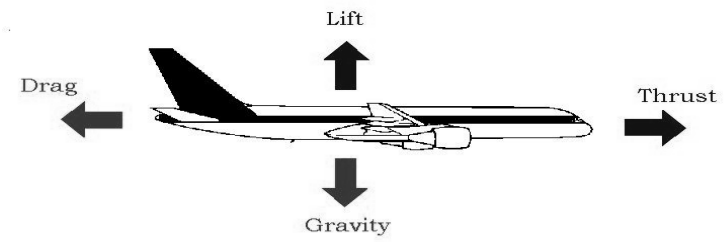


Figure 14: Forces acting on aircraft

2.2.1 Lift

Lift is required to counteract the gravitational force acting on the aircraft and keep it flying. Lift is created because of the difference in pressure around the wings of aircraft and it is defined by the formula

$$L = C_l * \frac{\rho V^2}{2} * S \quad \text{----------} \quad [2]$$

where C_l is the coefficient of lift, ρ is the density of air, V is the velocity and S is the wing area.

2.2.2 Drag

Drag is the resistance force in a fluid (in this case air) that opposes the motion of the aircraft. It is generated by the interaction of airframe body with the surrounding air and the difference in velocities between them. Drag is defined by the equation [3].

$$D = C_d * \frac{\rho V^2}{2} * A \quad \text{----------} \quad [3]$$

where C_d is the coefficient of drag and A is the reference area. The drag coefficient is expressed in terms of C_l and is given by equation [4].

$$C_d = C_{d0} + {C_l}^2 / \pi ARe \quad \text{----------} \quad [4]$$

where AR is the aspect ratio of wing and *e* is the wing efficiency factor.

2.2.3 Thrust

Thrust is the mechanical force generated by the aircraft engines to overcome the drag generated by the aircraft and is a function of the airspeed and density of the surrounding air. If the thrust at any given time is known then fuel consumption can be calculated using the thrust specific fuel consumption (TSFC) which in turn is a function of true air speed and is given by the formula

$$TSFC = C_{f1} * (1 + V_{TAS} / C_{f2}) \ (10) \quad \text{----------} \quad [5]$$

where C_{f1} and C_{f2} are constants.

2.2.4 Climb Performance

Using the coefficients provided in the BADA model takeoff thrust and maximum climb takeoff can be calculated including the fuel consumption. The maximum climb thrust for a jet engine aircraft at standard atmosphere conditions is calculated using the equation [6]

$$(Thr_{\max climb})_{ISA} = C_{Tc,1} * \left(1 - {Hp}/{C_{Tc,2}} + C_{Tc,3} * Hp^2\right) (10) \quad \text{---------} \quad [6]$$

where Hp is the geopotential altitude and $C_{Tc,x}$ are the climb thrust coefficients provided in the OPF file. This thrust includes thrust from all engines. The Thrust Specific Fuel Consumption (TSFC) for a jet engine can be calculated using the formula

$$TSFC = C_{f1} * (1 + \frac{VTAS}{C_{f2}}) (10) \quad \text{----------} \quad [7]$$

Fuel consumption can be calculating the thrust and TSFC.

2.2.5 Cruise Performance

The BADA model is impressively accurate in the cruise phase of flight. In a level and steady cruise thrust is equal to the drag on the aircraft (10). Therefore fuel consumption in cruise can be calculated using TSFC if drag is known, which can be calculated using the formulae described in Section 2.2.2

2.2.6 Descent Performance

The descent thrust is given as a ratio of the maximum climb thrust and it depends on the configuration (descent or landing) and altitude (high or low). It is given by the formula

$$Thr_{des,high} = C_{Tdes,xyz} * Thr_{\max climb} (10) \quad \text{----------} \quad [8]$$

Where $C_{Tdes,xyz}$ is the correction factor which is different for each xyz = [high altitude descent, low altitude descent, approach, landing]. These correction factors are provided in the OPF file.

2.3. Neural Network Fuel Consumption Model

This is a model developed by Trani, et al. 2004 at air transportation systems laboratory of Virginia tech using the Mat lab's neural networking toolbox. This model was developed to estimate the aircraft fuel consumption throughout the flight envelope using the aircraft performance manual alone. It also demonstrated fairly accurate performance compared to the actual flight fuel consumption data and is

more accurate than the existing analytical BADA model which is widely used. However the neural network model covers only a very limited aircraft types whereas this study requires a larger set of aircraft data base.

2.4. NAT Operations and Separations Studies

A previous study done by Gerhardt-Falk et al. (2000) investigated the potential benefits of reduced separation (vertical, longitudinal and lateral) and quantified the fuel savings and reductions in ATC communication loadings. This study investigated the benefits in reducing the then existing 2000 ft. vertical separation to 1000 ft. and indicated $25 million in fuel burn savings in 2010. The savings in reducing longitudinal separation from 10 minutes to 5 minutes are estimated to be $2.1 million in 2010. Similarly, the savings in reducing lateral separation from 60 nm to 30 nm was estimated to be $5.9 million in the year 2010. This study also investigated a hypothetical "Free flight scenario" where every flights travels on its most optimum path and the benefits for this scenarios were estimated to be $70 million in year 2010. The framework of this study was not data link equipage centric because at the time of this study data link avionics like CPDLC and ADS-C (B) were in nascent stages. Today data link usage is wide spread and its usage is expected to grow and therefore the new CONOPS are tightly integrated with data link equipage. Moreover, the operations would be segregated in vertical as well as lateral domains based on equipage, which alters the nature and magnitude of potential benefits.

The North Atlantic Simulation (NATSIM) is a model developed by FAA as a means for the stakeholders like Air navigation service providers (ANSP) and airlines to evaluate any policy, procedural and technological changes to the system. Using this model Chung and Post (2008) studied the commercial flight activity and ANSP fee structure changes in the NAT between specific countries. NATSIM is a macroscopic model which forecasts future demand and provides long-term operational and financial estimates at the aggregate level, which helps stakeholders in decision making. Its primary outputs are flight counts, ANSP's revenues, airline costs and flight characteristics.

A study performed by Williams (2005) examined the potential benefits of reduced separations in the NAT and focuses on the sensitivity of benefits with traffic demand and equipage levels. This study also estimates the benefits which could result from replacing the weight of saved fuel with additional cargo. One major assumption made in this study was that in reduced lateral scenarios the capacity of the track system increases by adding additional tracks in between the 1° spaced tracks. Recently ICAO opted to keep the number of tracks the same and free up some of the OTS airspace to random flights as

discussed in Section 1.3.2. Another difference between current proposed CONOPS and the ones assumed in this study was mandating the usage of data link, which was conceptualized and developed in the recent years by ICAO. Due to this mandate unequipped aircraft are denied access to OTS tracks above FL 360 and could result in large penalties.

Chartrand et al. studied the benefits in using to ADS-B IN for in-trail procedures. ADS-B IN provides real time traffic information in the cockpit display and improves the situational awareness of the pilots. With increased situational awareness pilots can maneuver altitude changes in the oceanic airspace, including OTS, with ease and safety. Flights can optimize their altitude profiles by executing multiple altitude changes. The study showed that the benefits, in terms of fuel savings, increase with increased ADS-B IN equipage levels.

CHAPTER 3 Data Sources and Inputs

3.1 Data Sources

Airspace simulation is a complex task and involves large amounts of data from a variety of data sources. All the data sources used in the model are outlined below.

3.1.1 Enhanced Traffic Management System (ETMS) Data

ETMS is a product of the FAA to monitor and predict the traffic levels in the national airspace system. It also consists of historical data of all flight operations and has information on individual flights including flightID, Aircraft type, Origin, Destination, Departure date and time, arrival date and time, flight plan and frequent position reports. ETMS data covers flights flying Instrument Flight Rules (IFR) in the NAT airspace. This data was used to create the flight schedule for the simulated days.

3.1.2 NAT Tracks Data

The NAT tracks for both easterly and westerly flights are published together by Gander and Shanwick centers daily (13). This track data for the required simulation period is provided by the FAA. This data contains the identifiers, waypoints of each track and OTS validity times. This data is used to recreate the exact tracks used for the days that are simulated.

3.1.3 NCEP/NCAR Reanalysis Project Wind Data

The Earth System Research Laboratory (ESRL) of the National Oceanic & Atmospheric Administration (NOAA) provides historical atmospheric information as a part of their NCEP/NCAR reanalysis project. The wind data used in this study is obtained from NOAA-ESRL reanalysis project website (6) (14). This wind information is available for altitudes ranging from 1000 milli bar (mb) (100 ft.) to 10 mb (100000 ft.) and in 2.5X2.5 degree grids for the entire globe.

3.1.4 FAA Aerospace Forecast-2012

This forecast provides information on various aviation system activities categorized into segments (2). The jet fuel prices and traffic growth factors for the NAT segment were used in the study to calculate the benefits.

3.1.5 Airlines for America (A4A) Data

Airlines for America, a trade organization of U.S. airlines, collects and maintains data on flight takeoff mass and fuel used. The takeoff mass data was used for creating distributions of takeoff mass for all the

aircraft types used in the model whereas the fuel consumption data was used to validate the model fuel burn outputs.

3.1.6 ICAO International Atmospheric Model

This atmospheric model consists of temperature, density of air, speed of sound and pressure for altitudes up to 80 km as published by ICAO Doc-7488 and is the standard adopted by ICAO for all modeling purposes. NATSAM models adopted this standard atmospheric model for all atmosphere related calculations.

3.2 Inputs

The different inputs required by NATSAM model are the specific formats in which they should be presented are described in the following sections. Some of these inputs must be preprocessed from the input data sources while some of these can be direct inputs.

3.2.1 NAT Flight Schedule

This file should contain all the flights departed in the time period of simulation and consist of flight information such as Origin, Destination, Aircraft Type, Flight ID, Equipage (optional) and Departure time for the period of simulation. Data from sources like ETMS and PDARS can be processed in Matlab to obtain the input file. The model is also capable of taking raw ETMS data in text format as input. In this study ETMS data was used to create the NAT flight schedule.

3.2.2 Airport Co-ordinates Database.

A database of all the airports present in the NAT flight schedule along with their ICAO codes, their corresponding latitudes, longitudes and elevation is required. This is used in matching the flight's origin and destination parameters to determine the geographic location of starting and ending points of the flight.

3.2.3 Wind Data

This data should contain the meridonal wind component (Vwind) and zonal wind component (Uwind) at every point on a grid covering the whole globe. The wind values can be daily mean values or values recorded at 6 hour intervals in a day (at 0, 6, 12and 18 hours UTC). In this study wind data is obtained from the NCAR reanalysis website of NOAA-ESRL Physical Sciences Division, Boulder, Colorado (6). The grid spacing of this data is 2 ½ degrees.

3.2.4 NAT Track Structure

The NAT OTS tracks are published daily by Gander Oceanic Center and Shanwick Oceanic Center. The NAT OTS track structure for the day that is being simulated should be modified to make an input file. Additional tracks should be inserted between the 1° separated tracks to reduce the separation to ½°. The additions must be made in a way such that the enhanced track structure represents all possible track formations that could result from reduced separations during all phases as shown in Figure 15. This enhanced track structure should be given as input in Matlab data format and each track should contain the track name, latitudes and longitudes of all the way points for that track.

This enhanced track structure is created by the method described below.

1. The actual published track system which is spaced at 1° is considered and a track is inserted between every two adjacent tracks, thus reducing the separation to ½°. These inserted tracks are applicable on from FL 360 to FL 390 because these are the altitudes where RLatSM is applied.
2. A track is not inserted if any two adjacent tracks are separated by more than 1° which is the case if the tracks are split into north and south groups.
3. Most of the times NATZ is designed for flights between Europe and the Caribbean and it is located on a different axis than the rest of the tracks which are parallel to the Euro-American axis. If this is the case additional track is not inserted between NATY and NATZ tracks.

Actual track system and enhanced track system for 03/15/2008 are shown in Figure 15. The top view of the track systems is shown in the upper panel and their respective cross section when viewed from the Canadian side is shown in the bottom panel. Each square represent a track/flight level combination and squares in green are the actual tracks and squares in red are the inserted tracks.

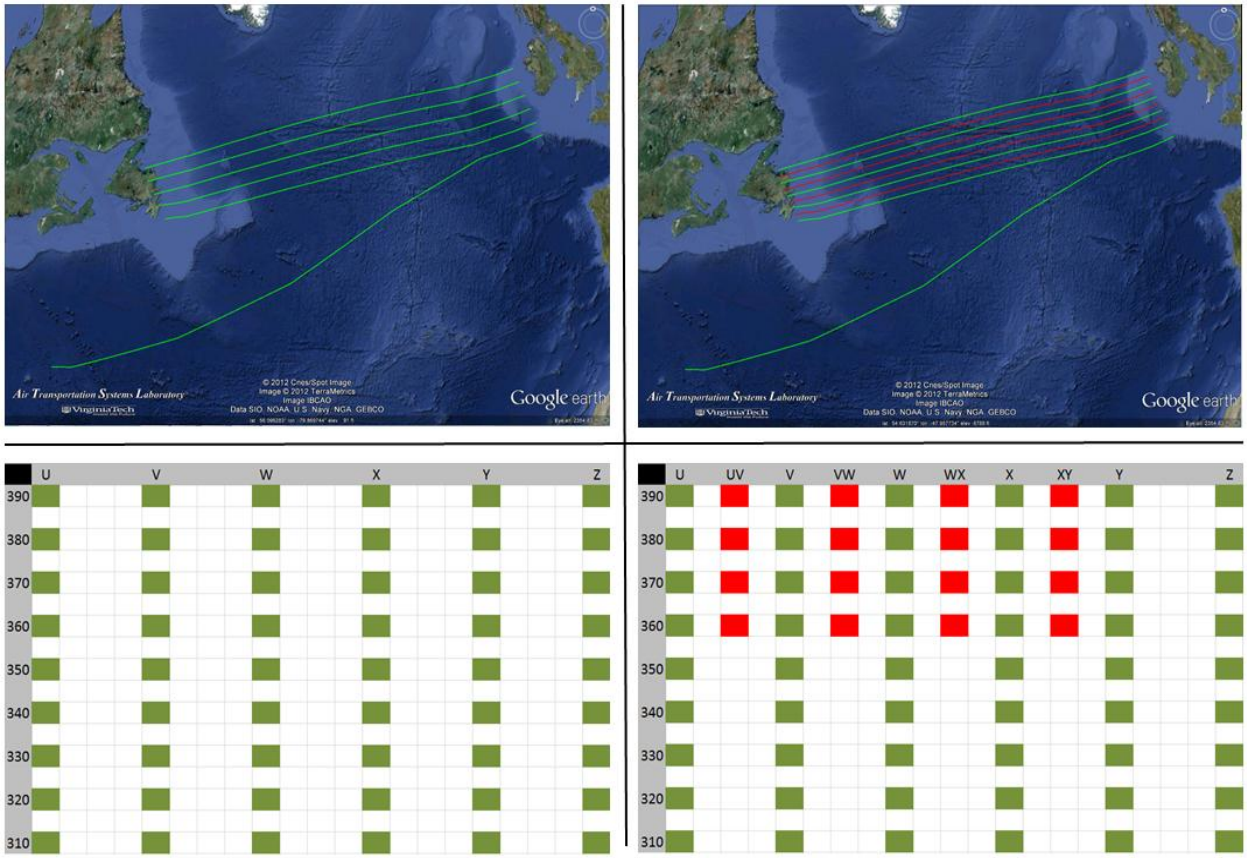


Figure 15: Actual track system (left) and Enhanced track system (right)

3.2.5 BADA 3.9 Data

The NATSAM model requires the BADA data for all the aircraft types being modeled in the simulation. This data should be organized in a Matlab data structure and can be generated from the ASCII files (OPF and APF) provided in the BADA model. This data is used in the fuel and travel time calculations. A screen capture of BADA data in matlab format is shown Figure 16.

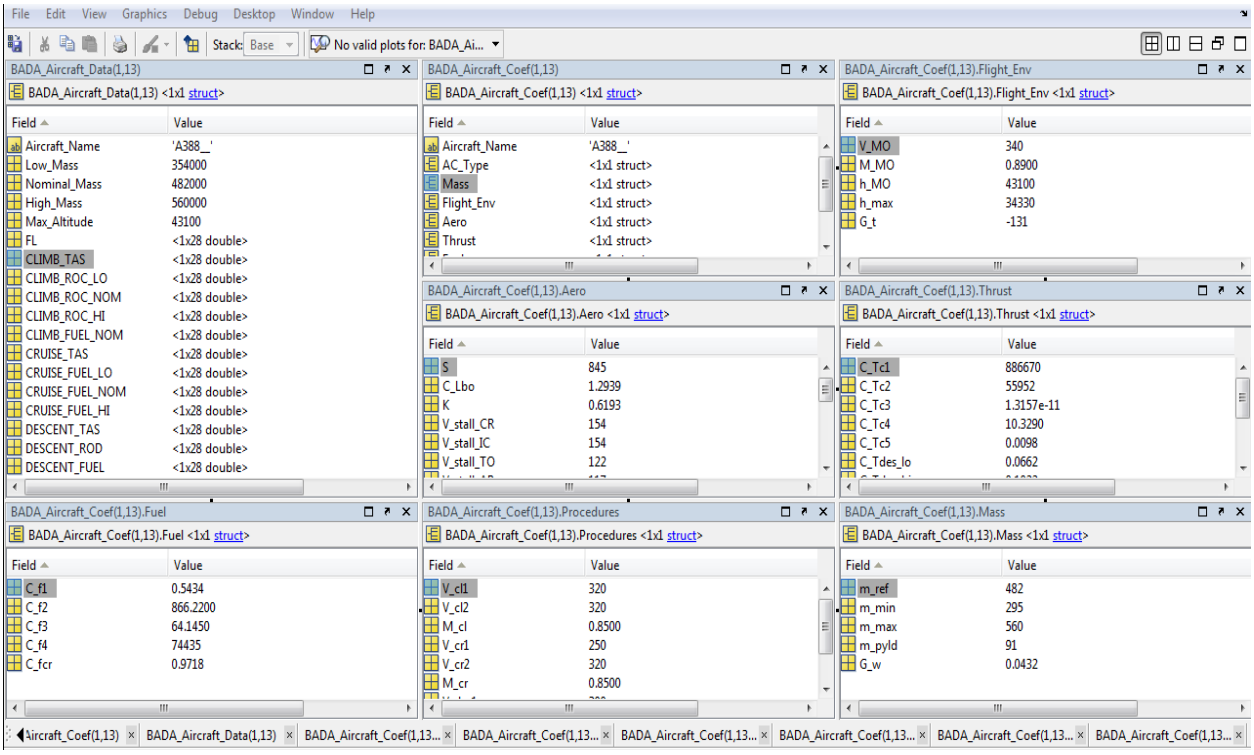


Figure 16: Screen capture of BADA data structure for Airbus A380-800

CHAPTER 4 Model Description and Validation

The objective of NATSAM model is to simulate NAT flight operations in the current state NAT system well as in future CONOPS. The model is capable of estimating various Key Performance Indicators (KPI) of the system like track utilization, cruise altitude assignment and Level of Service (LOS) for different scenarios. The model assists decision makers to understand the impacts of changes to the system. A description of the model is presented in the following sections.

4.1 Typical NAT OTS Flight in the Model.

A typical NAT OTS flight starts with filing the flight plan several hours prior to the departure. The flight dispatchers/pilots determine an optimum OTS track, cruising speed and flight level using flight planning tools and request that track/flight level combination in their flight plan where as in the model the least cost track/flight level combination from the flight cost matrix is considered as the most preffered track/flight level combination.The flight path adopted for modeling OTS flights is a hybrid of a great circle path and OTS tracks. In the model the flight climbs and cruises from its origin to the OTS entry point on a great circle path, then traverses the OTS track and exits at a known point to follow a great circle path to the destination. To account for the detours typically performed by flights in terminal airspace, a detour factor of 1.05 is applied to the great circle distances. After departure, the flight climbs to its TOC and cruise to the NAT boundary at its optimum FL which is subjected to hemispherical FL rules and weight limitations. The time at which the flight entered the OTS is recorded as NAT entry time for that particular track/flight level combination. Under current operations, most flights using the NAT OTS cruise at their assigned altitude whereas in reality only 2-4% of flights make a step climb inside the OTS. By the time the flights exit NAT OTS, they have burned a considerable amount of fuel which enables them to climb to a more optimal altitude. At this optimal altitude they cruise until they reach their Top of Descent (TOD) point and descend to their destination.

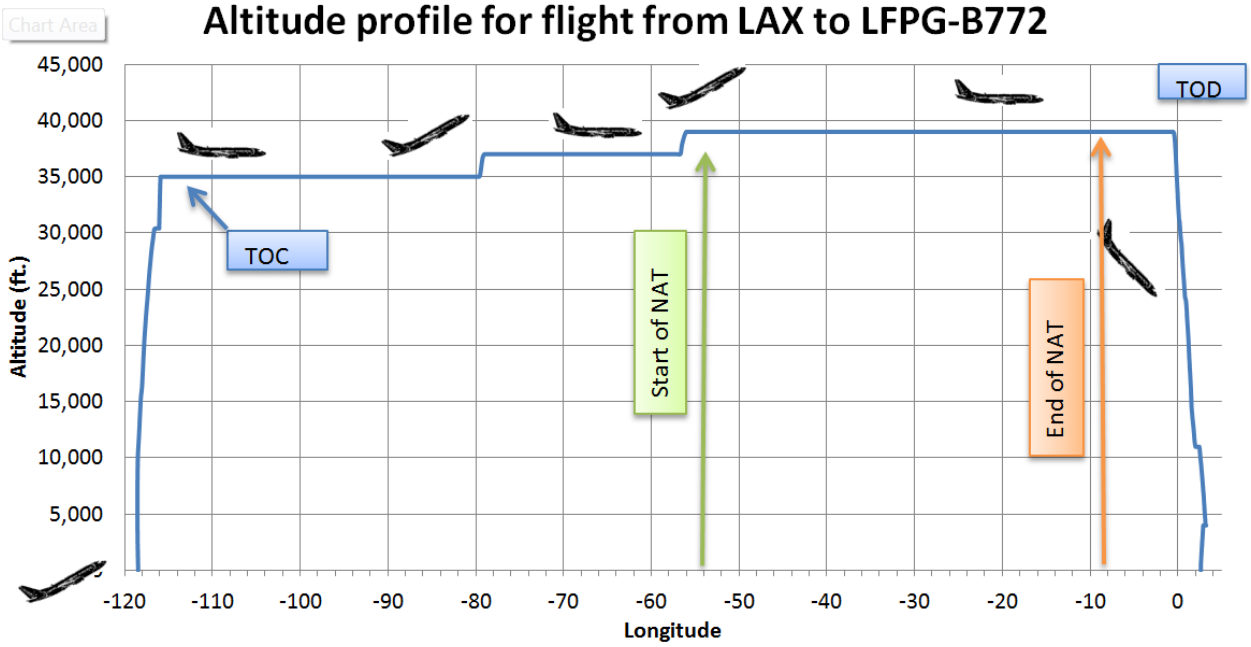


Figure 17: A typical NAT flight's altitude profile

4.2 Typical Random Flight Across NAT

All flights flying outside the OTS validity times are called random flights. Also those flights which fly during the OTS validity times but do not fly in one of the OTS tracks are called random flights. Theoretically a random flight can, and has the right to, file a flight plan that crosses or intersects the OTS but clearance to enter the OTS depends on the existing traffic at the time of entry. Controllers clear these flights below or above the OTS to avoid interference with OTS because interfering with OTS operations could mean sterilizing it for as high as 30 minutes (15). In this study for the sake of simplicity random flights are assumed to be not interfering with the OTS. In the model the climb and descent segments of these flights are similar to the NAT OTS flight discussed in 4.1. In the cruise segment, every time after cruising 1000 nm the flight's ability to climb to next possible altitude is verified. This altitude should be a valid FL and should satisfy hemispherical rules if outside NAT MNPS. If the flight is able to climb then a step climb is applied. Therefore typically 2 to 3 step climbs are made depending on the stage length of the flight. An assumption made in this study is that the % of flights performing step climbs on random routes is function of the equipage levels. Therefore a higher % of flights perform step climbs in cases where the equipage is high and vice-versa. The path followed by random flights is a great circle path from their origin to destination. A detour factor of 1.03 is applied to the great circle distance to account for the typical detours performed in the terminal airspace.

4.3 Wind Module

The wind module is a set of functions which calculates the wind component along the direction of flight using the input 3-D gridded wind data as described in 3.2.3. If the current waypoint, next waypoint and altitude are known then it calculates the wind component on the line joining these two waypoints. It is capable of calculating winds in two methods. The first method snaps to the point in the grid data which is closest to first waypoint and uses those corresponding values. This method is fairly simple and computationally fast but not very accurate. The second method uses 3 dimensional interpolation techniques and gives more accurate results but it is computationally very expensive. The difference between two methods is shown in Figure 18. User can specify the method to be used based on available computational resources and the granularity of data. If the data used is very fine grained, for example 0.25° grids or 0.5° grids, then first method can be used but if the data is coarse, 1° grid size or more, then 3-D interpolation is the best approach. The wind vectors obtained from the wind module are used by the flight cost module to calculate the ground speed of the aircraft and travel times.

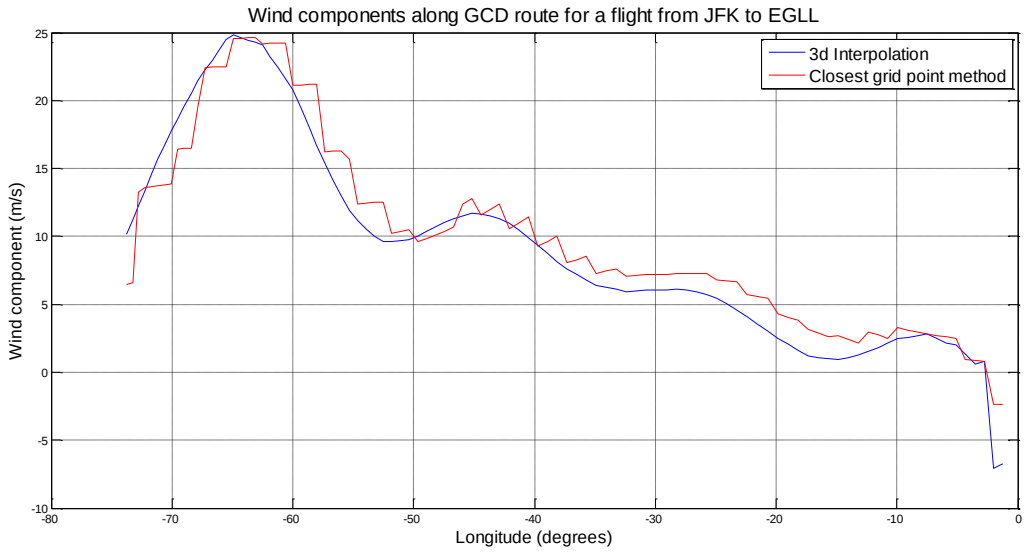


Figure 18: Winds calculated using wind module

4.4 Flight Cost Module

Any flight can cross the NAT by flying inside the OTS or outside of it on random routes. Operators compare the travel cost of both the above options and select the one with least cost. If OTS is selected then they further compare the cost of flying each track/flight level combination and select the one with least cost. Also they select two additional track/flight level combinations to file in the flight plan as

alternative preferences in case the preferred track/flight level combination is not available. Flight cost module also mimics the flight planning process followed by the airlines/operators before the actual flight. It calculates the travel cost for all the track/flight level combinations and random routes. Then it compares them with each other and selects the one with least cost as the preferred route.

Fuel cost and travel time cost for each flight are calculated for all the tracks across all flight levels (FL 310 to FL 390). For each flight there would be a fuel cost matrix and travel time cost matrix whose size is N*9. N is the total number of tracks for the given flight's direction in the enhanced track system as described in Section 3.2.4. This module not only calculates the cost for the actual 1° spaced tracks but also calculates cost for the intermediate ½° spaced tracks which probably could be a part of the track system in reduced separation scenarios as shown in Figure 19. The travel cost is calculated in the sequence of steps as described below.

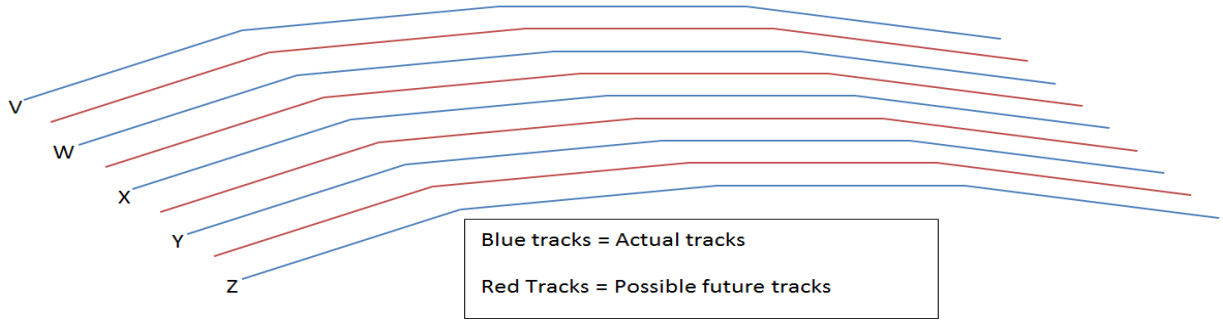


Figure 19: Enhanced Tracks system used for cost matrix generation

Note that even though cost matrix was generated for a track system with increased capacity, the simulation of NAT traffic only considers those tracks which were specified as valid tracks in the input and disregards other tracks.

4.4.1 Takeoff Mass (TOM)

The following procedure is adopted in calculating the takeoff mass:

1. The great circle distance (gcd) is calculated from the origin airport to the destination airport.
2. A detour factor of 10% is added to this gcd distance to obtain stage length.
3. Based on this stage length a nominal takeoff mass is calculated which is derived using the T100 data.
4. To randomize the takeoff mass, a random mass is added to the nominal takeoff mass and this random mass is derived from the standard takeoff mass deviation of the given aircraft type.

The standard deviations in takeoff mass for B763, B772 and A333 are obtained from operational data provided by Airlines for America (A4A) and for the rest of the aircraft types from T100 data. A random seed is used to generate the random mass to facilitate the generation of same takeoff mass distribution if desired across other cases and scenarios. The takeoff mass distribution for all aircraft types are validated with the actual operational data. At the origin-destination-aircraft type level the difference in means of model takeoff mass and the actual takeoff mass (from operational data) is well with in 5%.

4.4.2 OTS Cost Matrix Generator

For a given flight, the travel cost (fuel, time & distance) incurred in travelling each of the OTS tracks and each of the possible flight levels is calculated and saved in a matrix. This matrix is again sorted in the ascending order of fuel cost along with track and FL information. The travel costs are generated using the method discussed in 4.1 and also the mean entry time into all track/flight level combination is recorded as the flight's OTS entry time. This fuel cost and travel time cost matrices are generated for every flight in the NAT flight schedule input file. A depiction of cost matrix is shown in Figure 20. The cells in the matrix in red are flight levels which the flight is unable to climb to due to performance limitations and the greenest cell represents least fuel cost and it is considered as the preferred track-FL combination to traverse the OTS. This least cost track and FL depends on the existing wind conditions and also on the detour a flight has to make to reach that track.

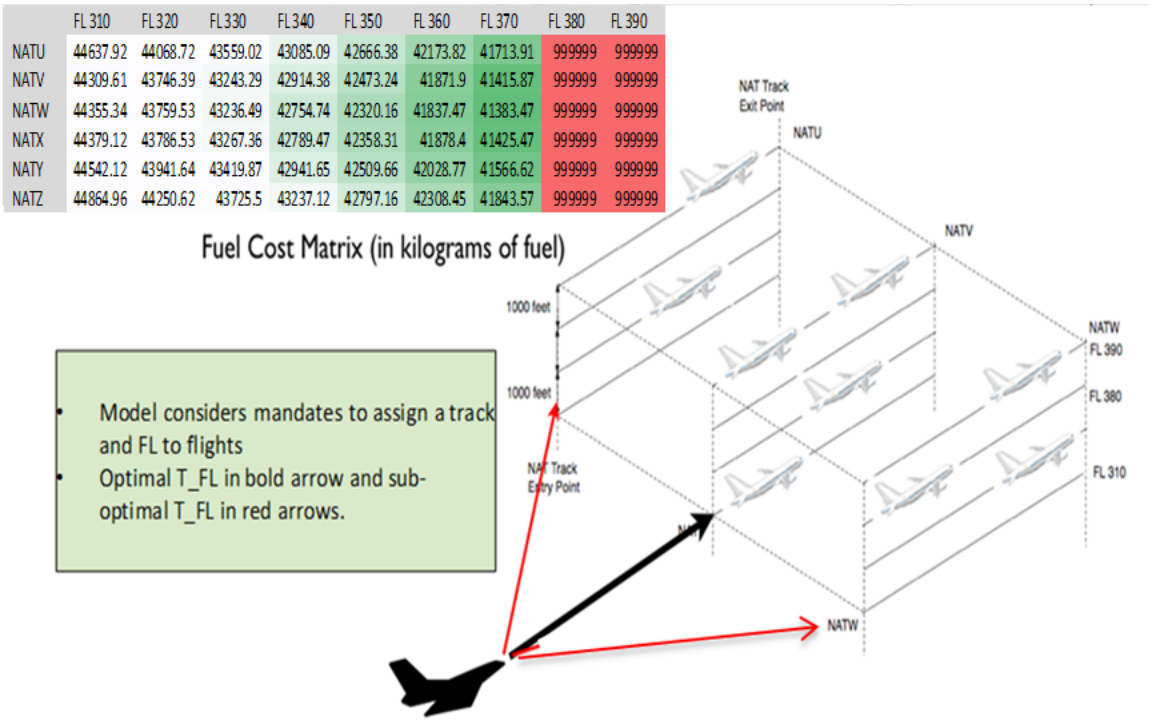


Figure 20: Sample cost matrix generated by OTS cost module

4.4.3 Random Cost Generator

The travel cost incurred in flying a random route is generated for all flights assuming that they fly a random route from their origin to destination regardless of the OTS. This random cost is calculated primarily to compare with OTS travel cost and decide whether the flight would prefer the OTS to a random route. Also this random cost enables the model to switch the flights from the OTS to random route if the OTS is not accessible due to any CONOPS or mandate, provided the random route is economical than the best available OTS route. The random flights are modeled as described in 4.2. Random flights perform step climbs even in the NAT and the number of flights performing step climb increases as the equipage increases. There fore the percent of flights that perform step climbs must be specified for each equipage level and accordingly flights perform multiple step climbs based on these values.

4.4.4 Output of Flight Cost Module:

The output of flight cost module contains fuel and travel time cost matrices, NAT OTS entry time and random travel cost. A sample output is shown in Figure 21. The left panel in the figure shows different metrics collected for a flight during its simulation like the fuel, time and distance cost matrices, entry time, wind experienced during the flight etc. and they are further broke down into different phases such

as climb, cruise, descent etc. for verification. The top right panel shows a fuel cost matrix showing fuel consumption for all track/flight level combinations. Each column in the matrix represents a track and each row a FL. Note the 9999999s in the first row indicating that the flight was unable to climb to the altitude of 39000 ft. due to operational constraints. In the bottom right panel is a fuel cost vector sorted in ascending order of fuel consumption and each row represents a fuel cost with its corresponding FL and track number. The first row (represents least cost) is considered the optimum path and the corresponding FL and track are considered as the flight planned track and FL in the assignment module.

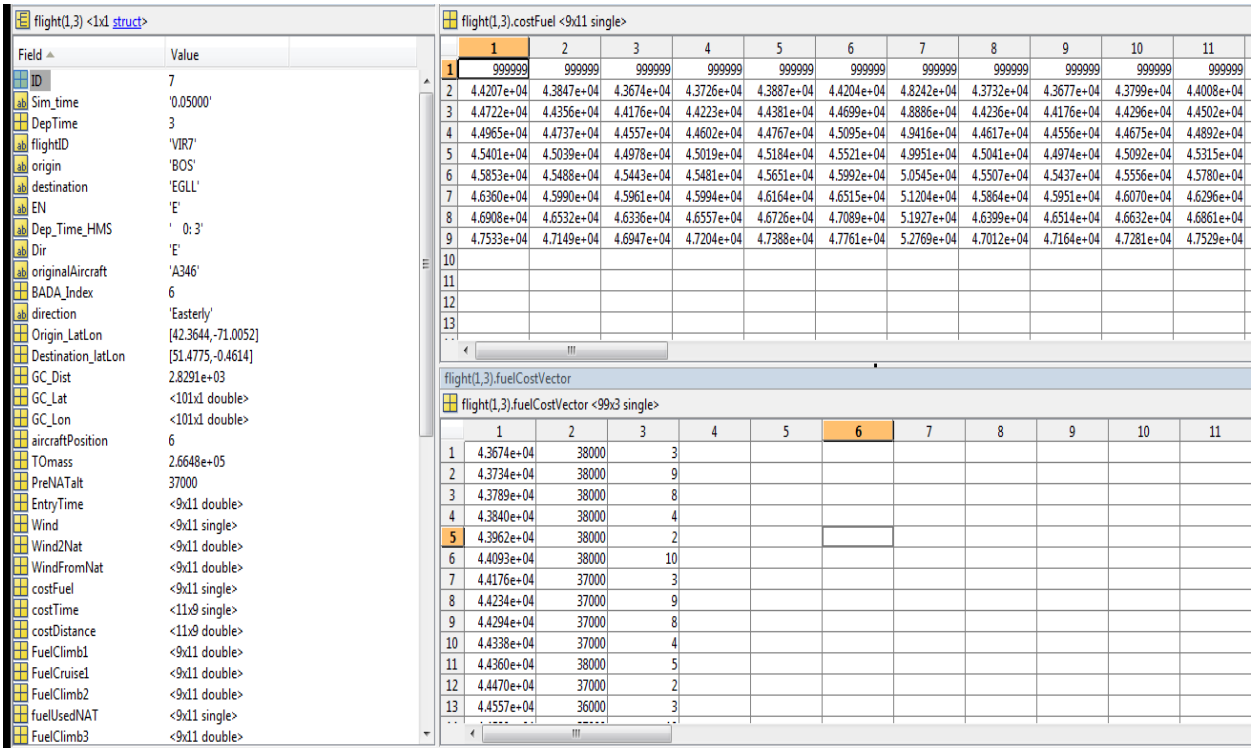


Figure 21: Output from flight cost module

4.5 Equipage Assignment

As per the ICAO, flights have to be fully equipped in order to operate inside the exclusionary airspace and therefore information on a flight's equipage is essential. If the user has data on the flight's equipage for the base line and each of the higher equipage cases (60%, 75%, 90%) then the user can specify the equipage for each flight as input in the NAT flight schedule described in Section 3.2.1. If such data is unavailable then each flight will be assigned an equipage attribute based on the rules and data provided by the user. For each aircraft type in the simulation the desired % of equipage should be specified for a given overall equipage level (60%, 75%, 90%). Then the model assigns equipage attribute to all flights

randomly based on the percentages specified. The equipage levels of aircraft types used in this study are based on a survey data conducted by CSSI, IATA and the FAA. This survey includes 44 US and international airlines which compromise 81.6 % of the total NAT MNPS operations. The rest of the operations are conducted by other airlines, general aviation and military and the equipage levels of the remaining 18.4% operations is assumed to be same as that of the surveyed airlines.

Aircraft Type	Equipage in NAT fleet		
	60% Equipage	75% Equipage	90% Equipage
A310	0%	0%	0%
A318	0%	0%	0%
A319	0%	0%	0%
A320-200	0%	0%	0%
A330-200	90%	97%	97%
A330-300	99%	100%	100%
A340-300	69%	75%	91%
A340-400	100%	100%	100%
A340-500	100%	100%	100%
A340-600	100%	100%	100%
A380-800	100%	100%	100%
B737-700	0%	0%	0%
B737-800	0%	0%	0%
B747-100	0%	0%	0%
B747-200	0%	0%	0%
B747-400	44%	55%	78%
B757-200	13%	30%	87%
B757-300	0%	0%	20%
B767-200	0%	0%	100%
B767-300	24%	65%	80%
B767-400	0%	57%	100%
B777-200	100%	100%	100%
B777-300	100%	100%	100%
MD-11	71%	92%	92%

Table 2: Equipage levels of aircraft types

4.6 Assignment Module

The assignment module simulates the operations in the NAT for a given period (42 hours in this study) and is the final step in the model. It requires all the flights sorted according to their entry times to the

NAT OTS entry points and it requires the flight costs for fuel and travel time. This module acts as oceanic air traffic control center and assigns flights to appropriate track/flight level combination and injects the flights into the OTS at safe separations. The different inputs required by the assignment module are discussed below.

4.6.1 In-Trail Separations

The default and reduced in-trail separations should be specified in minutes. The reduced separation is applied only between eligible aircraft and only in applicable track/flight level combinations. In all other cases default separation is applied. The separations (default and reduced) apply only to aircraft pairs with same cruise speeds and for aircraft pairs with dissimilar cruise speeds the separations are modified so that these pairs do not violate any separation minimums throughout the NAT segment. The separation modification criteria is explained below and shown in Figure 22.

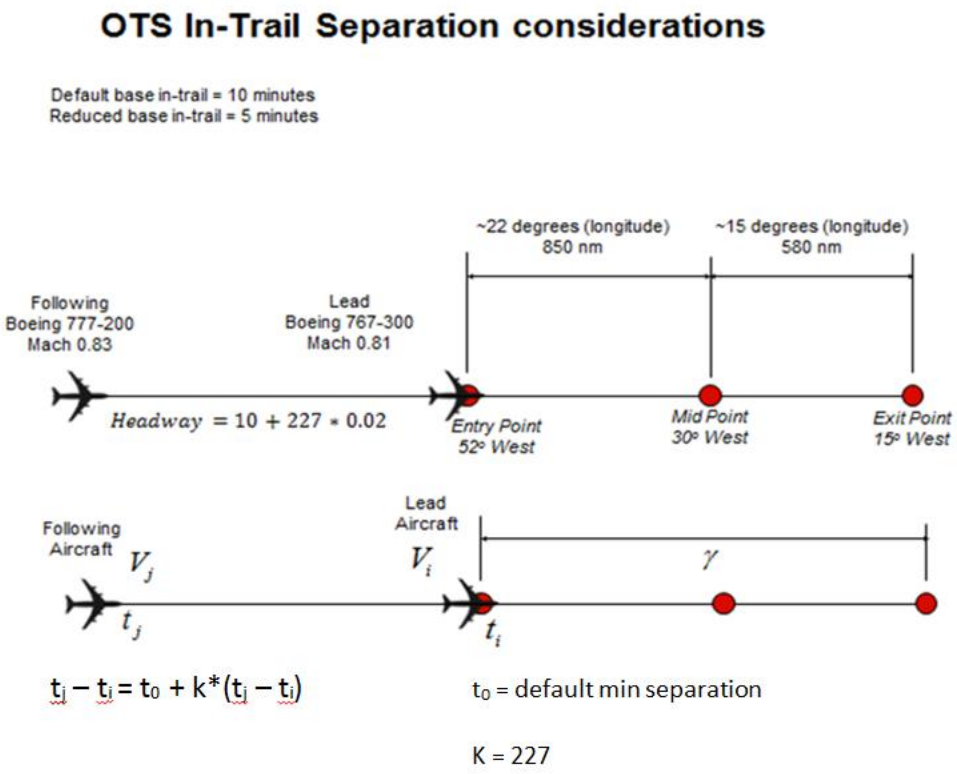


Figure 22: In-trail separation criteria

The variables t_i and t_j are the time of entry into NAT for the leader aircraft and follower aircraft respectively as shown in the figure. V_i and V_j are the cruise speeds in Mach for the leader and follower

respectively. γ is the distance between the entry and exit points of the NAT OTS and this distance usually is around 1500 nm. If $V_i \geq V_j$ then the separation which exists at the entry point gradually increases or remains same until they reach the exit point and in this case the separation is applied as is without any modification. If $V_i < V_j$ then the separation which exists at entry keeps on deceasing until they reach the exit point and in this case successive aircraft pairs are spaced such that the separation at the NAT OTS entry point is $t_0 + t_{\Delta mach}$ so that by the time they reach the exit point the separation between them is at least t_0. Here t_0 is the minimum separation and $t_{\Delta Mach}$ is $K*(V_j - V_i)$, K = 227 which is derived from empirical calculations.

4.6.2 NAT OTS Validity Times

The NAT OTS validity times are published along with the tracks by the oceanic control centers. Usually these times are for 0100 UTC to 0800 UTC for eastbound tracks and 1130 UTC to 1900 UTC for westbound tracks. These times are used to determine if a flight is entering the NAT boundary within OTS validity time or outside of it.

4.6.3 NAT Track System

The track system for all scenarios must be specified in this module. This information is used to differentiate valid track/flight level combination from the track/flight level combinations in the enhanced track system. The eastbound OTS track systems for different scenarios are shown in Figure 23 for 07/26/2008 to demonstrate how track systems for different scenarios are derived from enhanced track system for which the flight cost is calculated. Each square represent a track/flight level combination and squares in the green are accessible to both equipped and non-equipped aircraft. The squares in yellow are the tracks where DLM applies and only equipped aircraft are granted access. The squares in blue are the tracks where RLatSM is applied and only equipped aircraft are granted access. Note that each track system is a sub set of the enhanced track system.

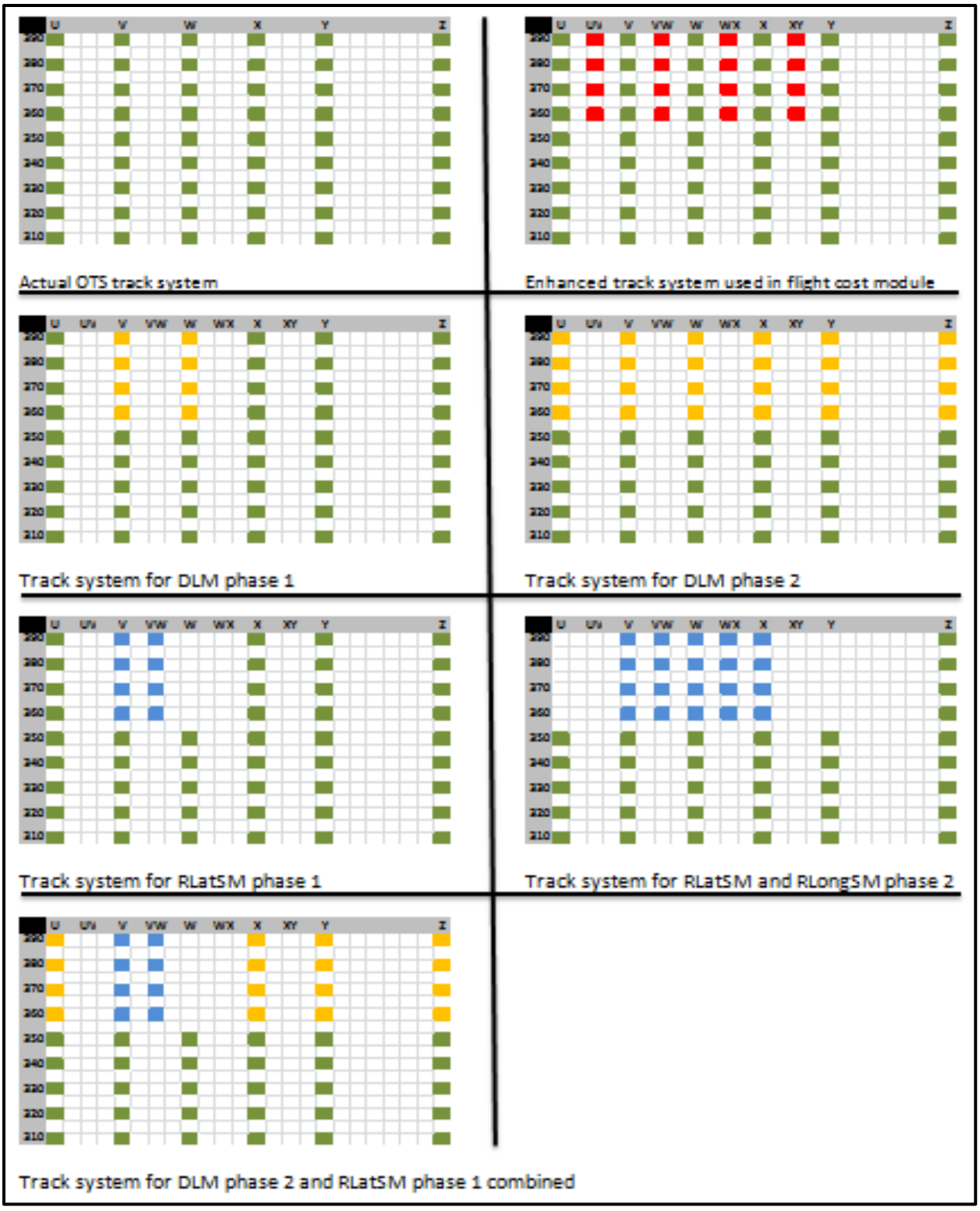


Figure 23: Easterly OTS track systems for all modeling scenarios

4.6.4 Track Assignment

The output of the flight cost module is taken as input by the assignment module. The assignment module takes one flight at a time in the order of ascending NAT entry time and assigns the best possible track/flight level combination to the OTS flights and random tracks to random flights. A flight is considered a random flight if it satisfies any one of the two conditions below and is assigned a random track (great circle path).

1. If the flight enters the NAT before or after the OTS validity times.
2. If the random track (great circle path) does not intersect the OTS tracks and fuel cost in flying that random track is less than the min (all OTS track/flight level combination fuel costs).

If it does not meet any of the above two conditions then the flight is considered as OTS flight and the flight is assigned best possible track/flight level combination depending on the availability. The least cost track/flight level combination (the first value in the sorted cost vector) is considered as the requested path and this request is validated. First the eligibility of the aircraft to traverse that track/flight level combination is verified and then the availability of that track/flight level combination is verified. The validation is done based on the requirements listed below.

- Determine whether the requested track/flight level combination exists in the track system of the scenario being simulated and if it doesn't then that request is invalid. Note that the travel cost is calculated for all the tracks present in the enhanced track system but the track system being simulated would be a subset of the enhanced system.
- If the requested track is a ½° spaced track then only equipped aircraft are allowed, else if the track is a 1° spaced track then any aircraft can enter that track.
- If the flight level requested is exclusionary, in other words if DLM applies to that FL, then only equipped aircraft are allowed to enter else all aircraft are allowed.
- Finally if the request is valid and the aircraft is eligible then it checks for availability of that track/flight level combination which is available if and only if the flight doesn't violate any immediate or projected longitudinal separations with respect to the previously injected aircraft.

If the requested track/flight level combination is valid and is available then the flight is injected into that track/flight level combination. If the request track/flight level combination is not valid or unavailable then the flight's next least cost track/flight level combination is considered as the new request and is validated again. This cycle repeats till the flight is assigned a track/flight level combination and the travel

cost corresponding to assigned track/flight level combination is read from the cost matrix as the actual cost. If none of the track/flight level combinations are valid or unavailable the flight is assigned a random track which is outside the OTS however this is a rare situation and was never observed during the study. This module keeps track of the assigned track/flight level combination, entry time, the lateral deviation in degrees from the requested track and vertical deviation in 1000's feet from the requested FL.

4.7 Validation of Output

NATSAM model creates detailed outputs (for individual flights) as well as aggregate level statistics. Figure 24 shows a sample detailed output file which contains different metrics for all flights. Note that the assigned flight level reported for the random flights is the weighted mean of all cruise FLs during the journey.

	A	B	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	flightID	Aircraft	Origin	Destination	Equipage	DepTime	NAT Entry Time	Direction	GC_Dist	TOmass	Fuel Consumed	Time Travelled	Distance Travelled	Assigned FL	BestFL	Assigned Track	Best Track	Airspace
2	[redacted]	MD11	LFPG	MEM	E	0.35	88.84	Westerly	3951.31	277492.15	75020.53	536.42	4004.84	31866.90	0.00	RND1	NA	RND
3		A346	BOS	EGLL	E	3.00	103.15	Easterly	2829.07	267555.42	43822.68	364.85	2887.50	38000.00	38000.00	NATV	NATV	OTS
4		B77W	CYUL	LFPG	E	15.00	117.26	Easterly	2982.54	228203.29	40805.81	380.03	3077.97	37000.00	37000.00	NATV	NATV	OTS
5		B744	JFK	LEMD	N	2.55	117.44	Easterly	3110.61	316726.70	59625.30	379.53	3216.62	33688.58	0.00	RND2	NA	OTS
6		B763	JFK	LIRF	N	2.00	118.01	Easterly	3707.51	180635.32	39628.96	475.90	3818.70	34000.00	34000.00	NATY	NATY	OTS

Figure 24: Sample output file

The model fuel consumption is validated with the operational data which is available for only 4 major aircraft types and is shown in Figure 25 through Figure 28. The differences in fuel consumption calculated by the model and actual fuel consumption for B772, B764, A333 and B763 are with in 5%, 5%, 6% and 8% respectively and these differences are even lower at 3000 – 3500 nm range. However the model data used for validation represents only one day where as the operational data represents several days. Due to the lack of operational data for the other aircraft types validation is carried by cross verifying the fuel consumption against the payload range diagrams published by the manufacturers.

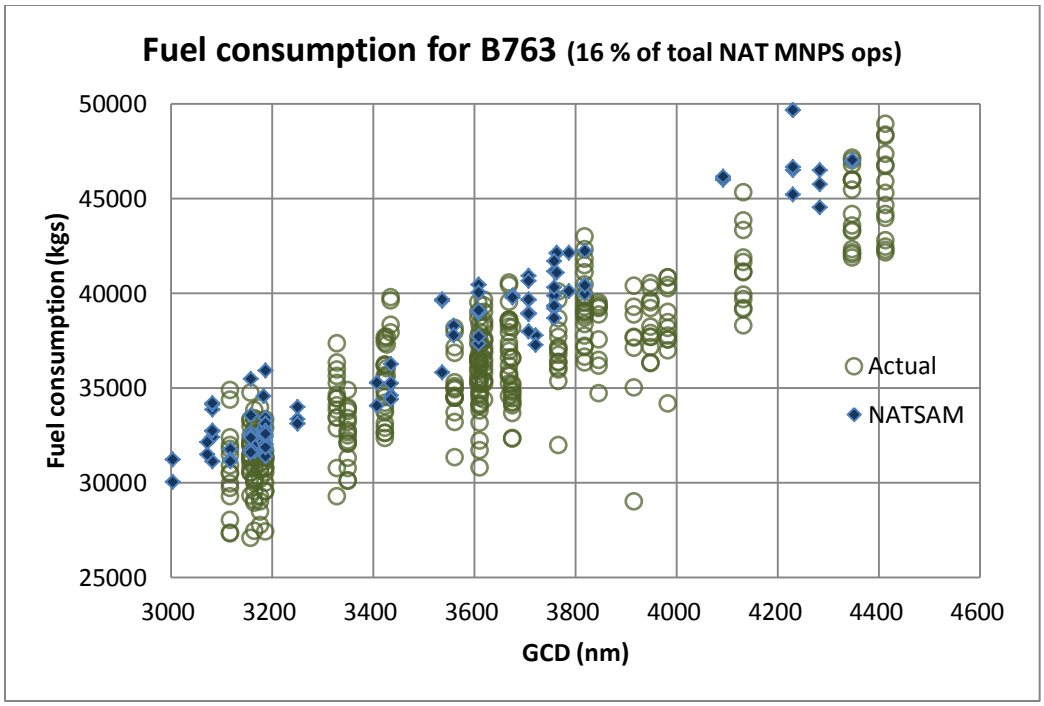


Figure 25: Fuel validation for B763

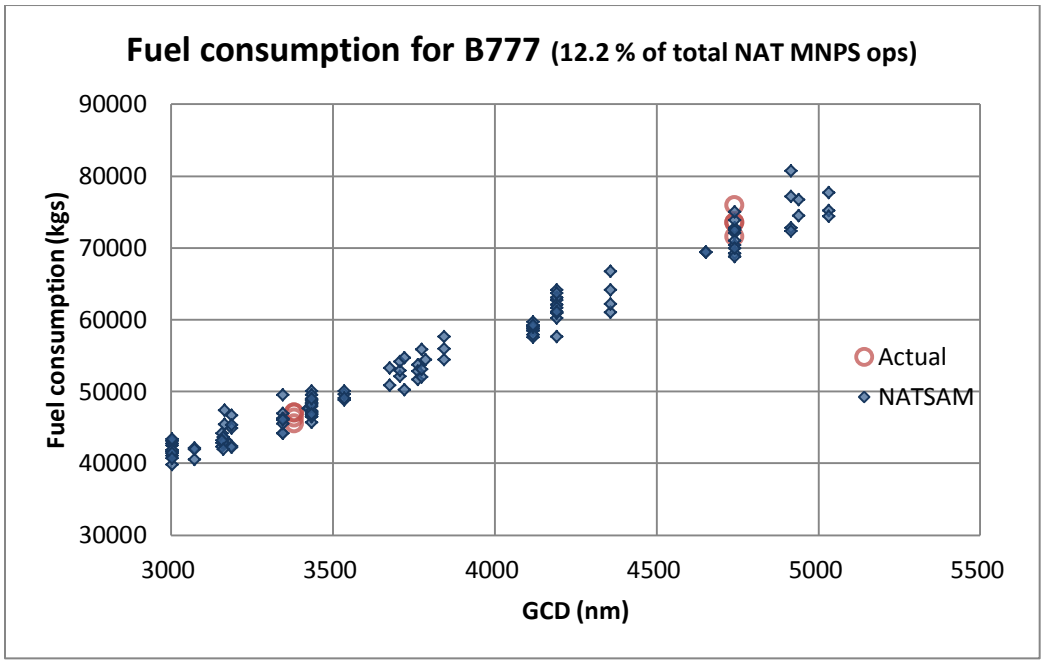


Figure 26: Fuel validation for B772

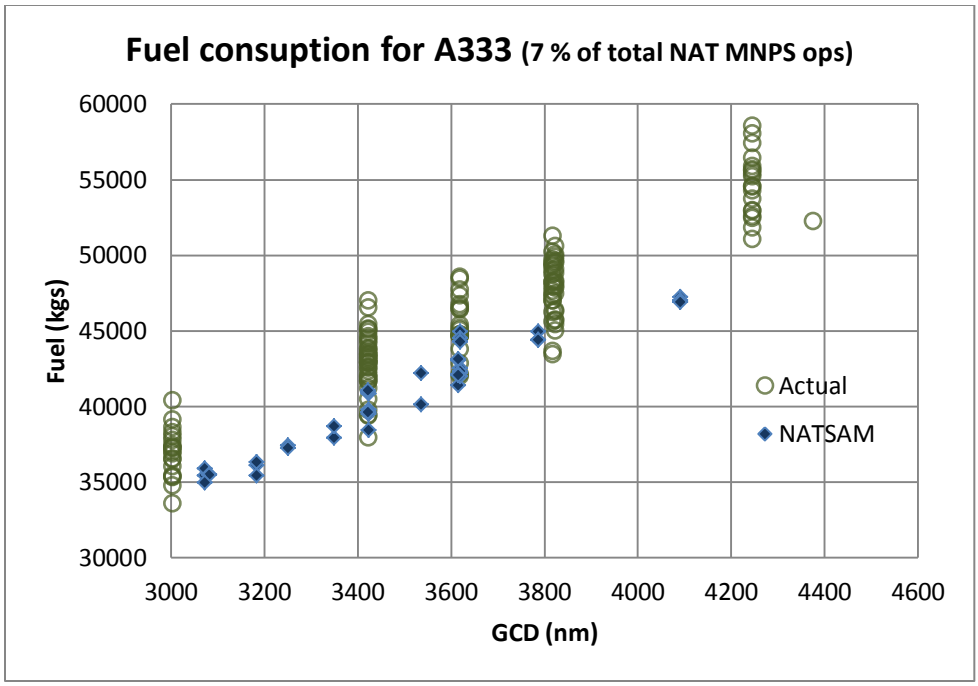


Figure 27: Fuel validation for A333

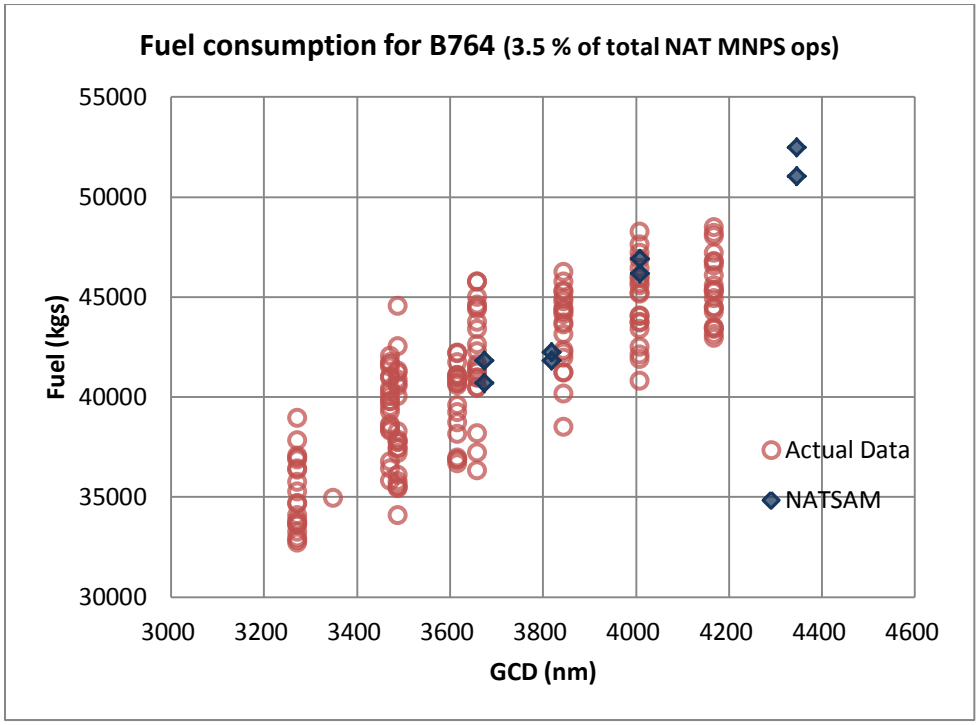


Figure 28: Fuel validation for B764

The model also produces metrics like FL distributions for every type and level of service. A plot of an aggregate level output for level of service is shown in for DLM phase 1 which has traffic demand for the

year 2013. The model predicts that only 60.6% of the flights in the NAT OTS get assigned the track-FL combination that they requested. This output is in agreement with the data provided by the UK NATS which indicates that in the year 2011, 62 % of the flights get the track-FL combination that they requested.

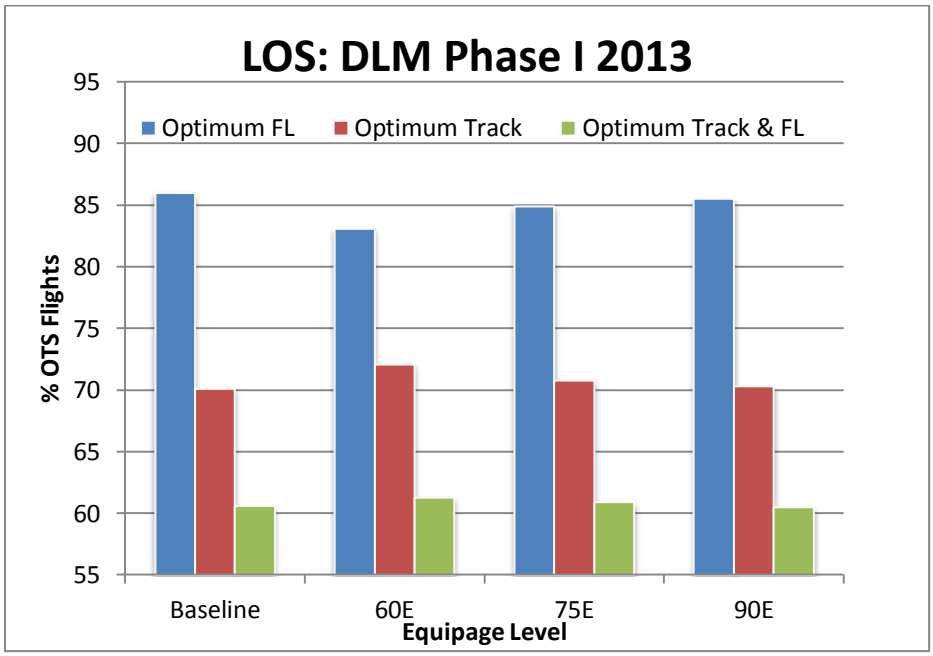


Figure 29: Level of Service for base line 2013.

The FL distribution observed in the model is validated with operational data collected for the year 2008 and validation for B763 and B772 are shown in Figure 30 and Figure 31 respectively.

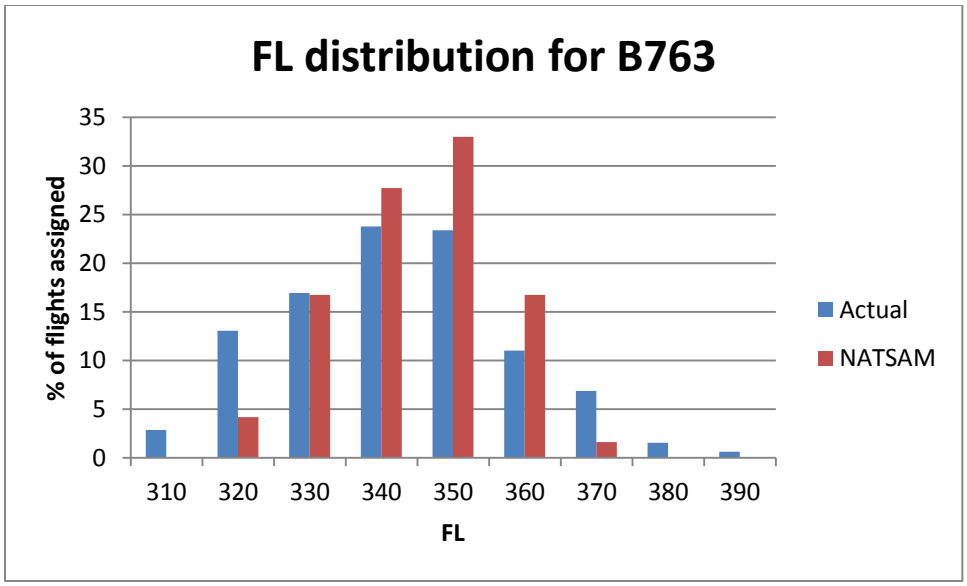


Figure 30: FL distribution for B763

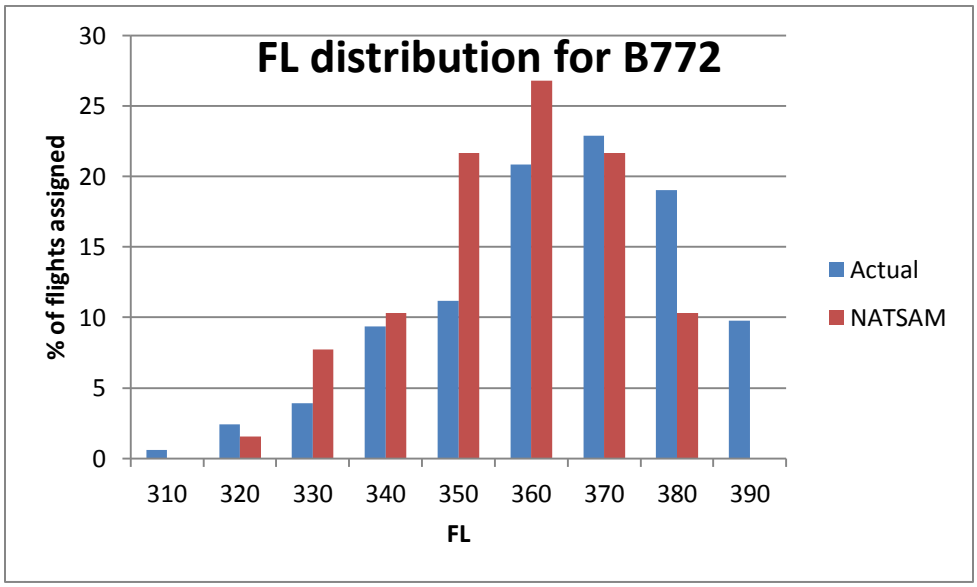


Figure 31: FL distribution for B772

The actual data obtained spans across a full year where traffic and load factors vary with seasons and reflects the variations in wind characteristics like Jetstream strength, altitude and location. On the other hand NATSAM data represents flights during a simulation period of 42 hours, where wind conditions remain same across the simulation period. Moreover the wind data used in the model was very coarse vertically and also laterally. The grid size of data in lateral domain is 2.5 degrees, in which 5 tracks could be placed at ½° spacing, and in the vertical domain wind readings are only available at every 50 mb

pressure altitudes (at 35000 ft. 50 mb translates into approximately every 5000 ft.). Due to these reasons there are differences in the FL distributions of actual data and NATSAM model. In spite of the coarseness of wind data the model draws clear distinction between each FL and replicates very well the trends followed by different aircraft. Overall NATSAM model demonstrates satisfactory performance in terms of fuel consumption, FL assignment and simulating the NAT traffic.

CHAPTER 5 Results and Conclusions

The objective of the model is to estimate the benefits or penalties experienced by all users because of introducing new CONOPS and mandates which are aimed towards increasing the safety and efficiency of the NAT airspace system. The primary benefits to the users of NAT airspace is reduction in fuel burns and travel times. The simulation outputs of the assignment module are used to derive the benefits/penalties using the approach described below.

- The mean fuel burn and travel time for all flights between a given origin and destination using a given aircraft is calculated.
- This mean fuel burn and travel time are calculated separately for each case (baseline, 60%, 75% and 90% equipped).
- The difference in fuel burn and travel time is calculated for each case with respect to baseline case and this difference is considered average benefit/penalty for a single flight between that origin and destination using that aircraft.
- This benefit is annualized by multiplying the benefit/penalty with the total number of flight count for a year.
- Total number of flights in a year between an origin and destination using an aircraft type is derived from the 2012 Traffic Flow Management System (TFMS) data.

The benefits and penalties reported in this study are the total benefits experienced by the airspace system in a given scenario and equipage level. The benefits and penalties experienced by individual flights including OTS, random, easterly and westerly are summed up to obtain global benefits.

Some of the assumptions made in the model are shown in

Passenger Value of Time (VOT)	$43.50/hour (as per ICAO) in year 2012
Fuel cost per gallon	$ 2.97 in 2012
Load factor (Atlantic market)	80.9 % in year 2012
Aircraft capacity	Derived from OAG fleets and airline data for 12 aircraft groups
All costs and benefits are estimated in year 2012 US dollars	

Table 3: Assumptions in the study

The passenger VOT was obtained from ICAO and the aircraft capacities were obtained from OAG fleets and airline data. Using the aircraft capacity and load factors the total number of passengers per flight is obtained which is used to calculate the total time savings. Again the time savings are converted into monetary benefits using the VOT.

5.1 Modeling Scenarios

Three CONOPS namely DLM, RLatSM and RLongSM were investigated in this research effort. Each of these three CONOPS were studies in different scenarios where each scenario corresponds to one of the traffic levels and each scenario has three cases where each case corresponds to a certain equipage level. The equipage levels considered were 60%, 75% and 90% and the corresponding cases are denoted as 60E, 75E and 90E respectively. The traffic levels studied were for the years, 2013 which is the baseline year, 2015and 2017 which are the horizon years. 60% equipage is the most likely equipage level anticipated by the year 2013 as the current level of equipage in NAT is approximately 55%. To accommodate the unequipped flights the CONOPS would be introduced in phases conforming to the implementation plans of ICAO as described in the Section 1.3. The complete list of modeling scenarios and cases is shown in Table 4. Each of these cases is simulated in the assignment module.

Case #	Scenario/CONOPS	Traffic Level	Case #	Scenario/CONOPS	Traffic Level	Equipage Level
	Data link Mandate Phase			**RLatSM Phase**		
BL-2013	Baseline – no mandate	2013	**BL-2015**	Baseline – no mandate	2015	N/A
2a	Phase 1 – two core tracks from FL 360 to FL 390	2013	**2b**	Phase 1 –½ degree spacing between two core tracks from FL 360 to FL 390; Exclusionary airspace applies only in these tracks and FLs; from 2015	2015	60%
3a		2013	**3b**		2015	75%
4a		2013	**4b**		2015	90%
BL-2015	Baseline – no mandate	2015	**BL-2017**	Baseline – no mandate	2017	N/A
5a	Phase 2 – all OTS tracks from FL 360 to FL 390	2015	**5b**	Phase 2 – ½ degree spacing in all OTS tracks; Exclusionary airspace applies only in these tracks and FLs; from 2017	2017	60%
6a		2015	**6b**		2017	75%
7a		2015	**7b**		2017	90%
	Data link Mandate Phase			**DLM Phase 1, RLatSM Phase 2 & RLongSM together**		

BL-2017	Baseline – no mandate	2017	**BL-2015**	Baseline – no mandate	2015	N/A
8a	Phase 2 – all OTS tracks from FL 360 to FL 390; from 2017	2017	**2c**	Phase 1 – 5 minutes in-trail separation in all OTS tracks from FL 360 to FL 390; – ½ degree spacing between **two core tracks** from FL360 to FL390; Exclusionary airspace in all tracks from FL 360 to FL 390; from 2015	2015	60%
9a		2017	**3c**		2015	75%
10a		2017	**4c**		2015	90%
	DLM Phase 1 & RLatSM Phase 2 together			**DLM Phase 2, RLatSM Phase 2 & RLongSM together**		
BL-2015	Baseline – no mandate	2015	**BL-2017**	Baseline – no mandate	2017	N/A
2ab	Baseline – no mandate DLM Phase 2 and RLatSM Phase 1 together ½ degree spacing between two core tracks; Exclusionary airspace in all OTS tracks from FL 360 to FL 390	2015	**5c**	Phase 2 – 5 minutes in-trail separation in all OTS tracks from FL 360 to FL 390; – ½ degree spacing between **all OTS tracks** from FL360 to FL390; Exclusionary airspace in all tracks from FL 360 to FL 390; from 2017	2017	60%
3ab		2015	**6c**		2017	75%
4ab		2015	**7c**		2017	90%

Table 4: Modeling scenarios

5.2 DLM Phase 1

The operational changes that come into effect in DLM phase 1 are the FLs from FL 360 to FL 390 in the two core tracks are made exclusionary and are available only to the equipped aircraft. All the remaining tracks and FLs are available to all aircraft. Three cases 2a, 3a and 4a with equipage levels 60%, 75% and 90% respectively are studied in this scenario with 2013 traffic levels.

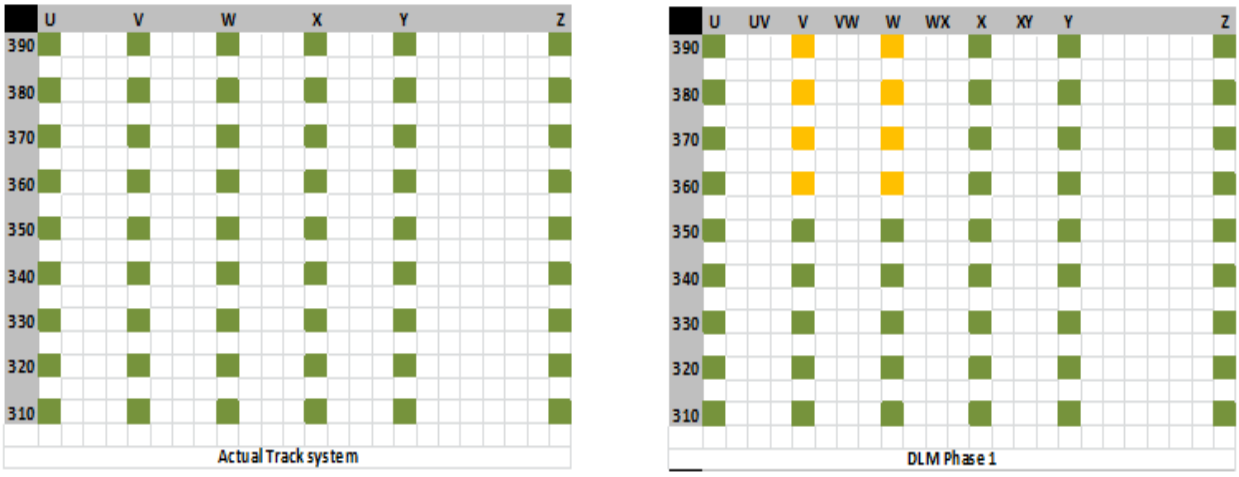


Figure 32: DLM Phase 1 Track system

The benefits experienced by operators during DLM phase 1 is shown in Figure 33.

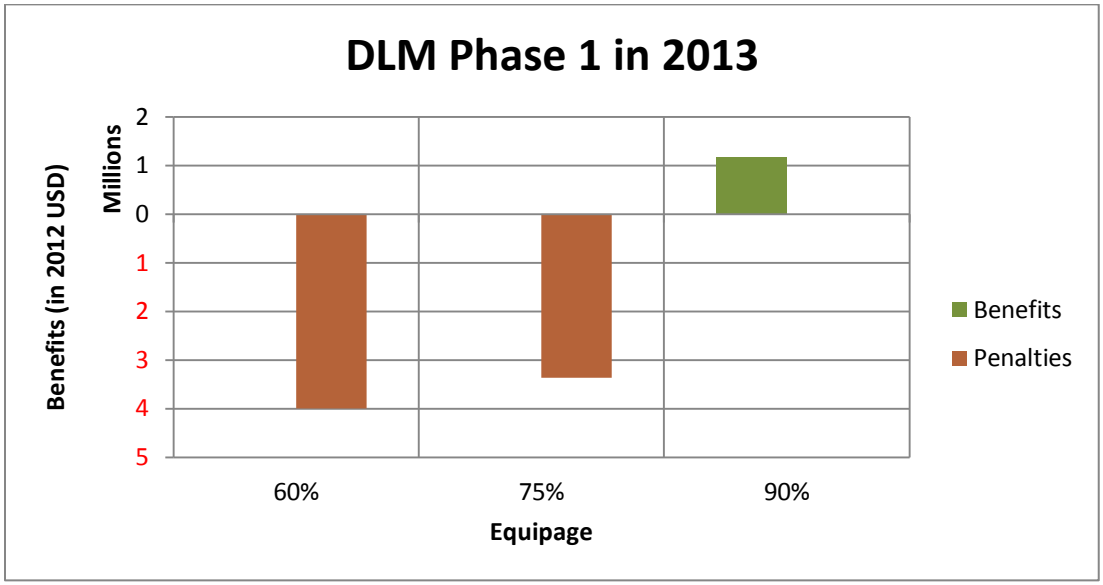


Figure 33: Benefits in DLM Phase 1

The benefits in DLM phase 1 are -$3.4 million, -$2.8 million and $1 million at 60%, 75% and 90% equipage levels respectively. In 60E and 75E cases those unequipped aircraft whose optimum paths are part of exclusionary airspace are now forced to either deviate laterally to another track or deviate vertically to another flight level below FL 360. Some of the equipped flights experience benefits because they are able to fly more optimum paths which are otherwise not available due to congestion but still the penalties experienced by unequipped aircraft are far higher than the benefits. In 90E case only 10% of the unequipped flights are restricted access to exclusionary airspace most of them being older aircraft. Due to the high equipage, random equipped flights are able to fly optimum flight profiles by performing step climbs and realize benefits.

5.3 DLM Phase 2

In DLM phase 2 the exclusionary airspace extends from the two core tracks in phase 1 to all the OTS tracks from FL 360 to FL 390. Cases 5a, 6a and 7a with equipage levels of 60%, 75% and 90% respectively with 2015 traffic levels and cases 8a, 9a, 10a with equipage levels of 60%, 75% and 90% respectively with 2017 traffic levels are studied in this scenario.

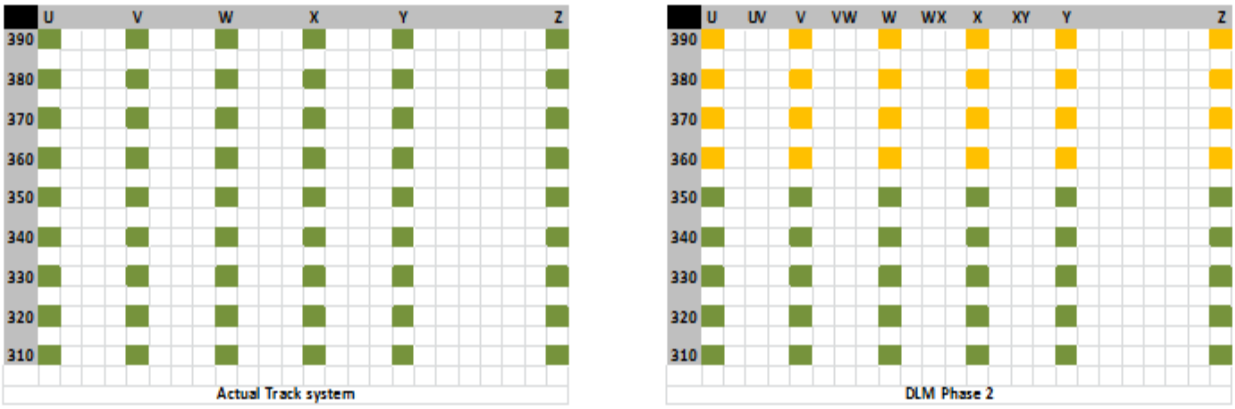


Figure 34: DLM Phase 2 Track system

The benefits/penalties experienced by operators during DLM phase 2 in 2015 is shown in Figure 35 and in the benefits for 2017 is shown in Figure 36.

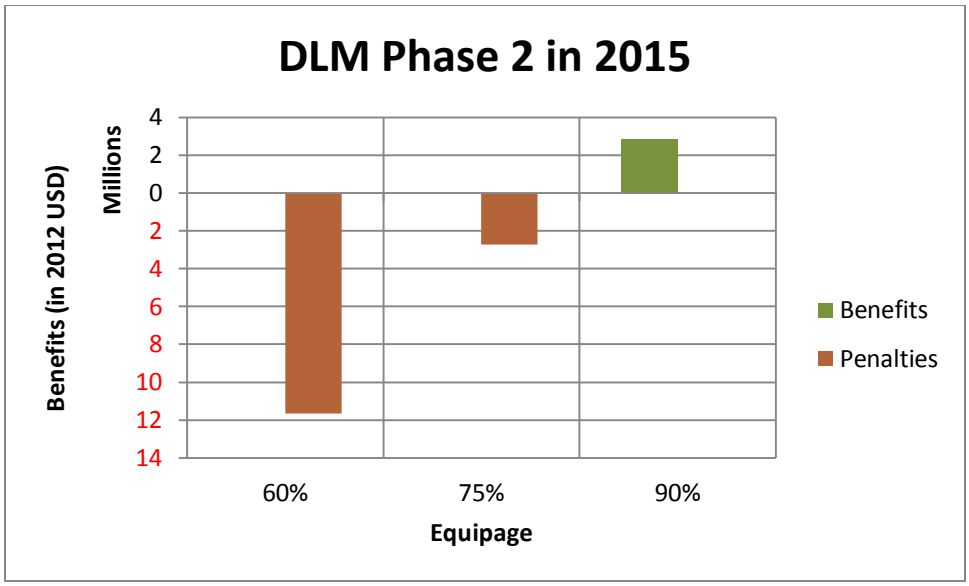


Figure 35: Benefits in DLM Phase 2 in year 2015

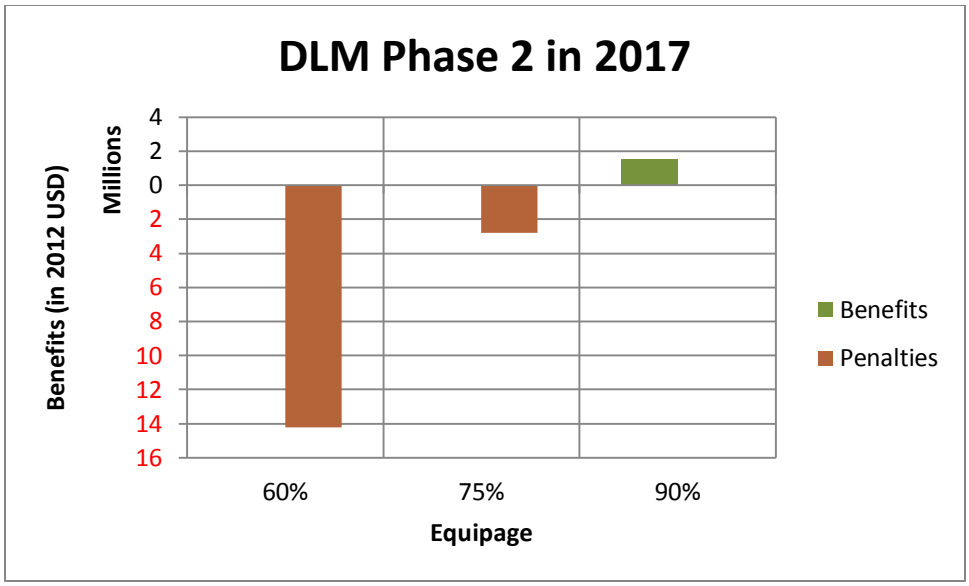


Figure 36: Benefits in DLM Phase 2 in year 2017

The penalties experienced in DLM phase 2 in years 2015 and 2017 are much higher than the penalties in DLM Phase 1. In DLM Phase 2 the mandate is extended to all tracks in the OTS where as in DLM Phase 1 the mandate is confined to the two core tracks only. All unequipped flights desiring to fly above FL360 have to fly below or opt for a random track and Figure 37 shows the changes in FL assignment for all unequipped flights in baseline and 60E cases. In 60E case, since 40% of the flights are unequipped and affected by the mandate, 154 flights are forced to a flights levels below FL 360 thus inflicting huge penalties on them. Compared to baseline case, the demand for FL 350 to FL 310 increased in 60E case, especially for FL 350 and FL 340 it almost doubled. As a result of this high demand, even a few equipped flights whose desired altitudes are FL 350 and FL 340 are forced to lower altitudes. As the equipage level increases, the percent of flights affected by DLM decreases and thus penalties also decrease. In highly equipped 90E case, a very few OTS flights are penalized but most of the random flights experience benefits because of the ability to execute step climbs.

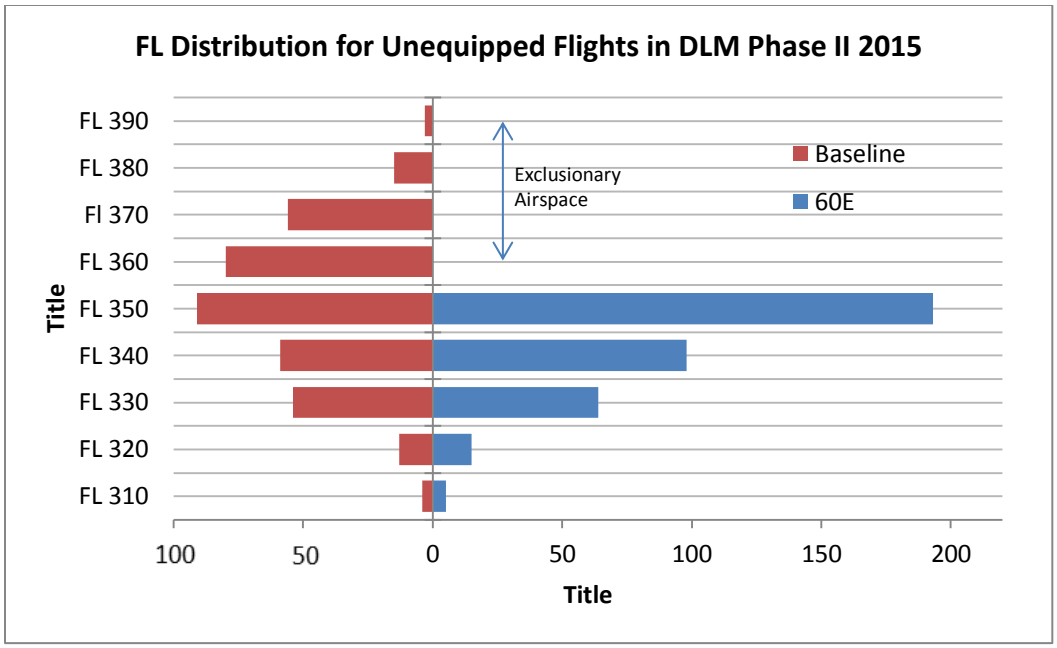


Figure 37: FL Distribution for unequipped flights in DLM Phase II 2015

5.4 RLatSM Phase 1

The track structure in RLatSM phase 1 is same as DLM phase 1 but now the two core tracks above FL 360 are placed at ½ ° spacing. Datalink mandate applies to these core tracks above FL 360. Three cases 2b, 3b and 4b with equipage levels 60%, 75% and 90% respectively with 2015 traffic levels are studied in this scenario.

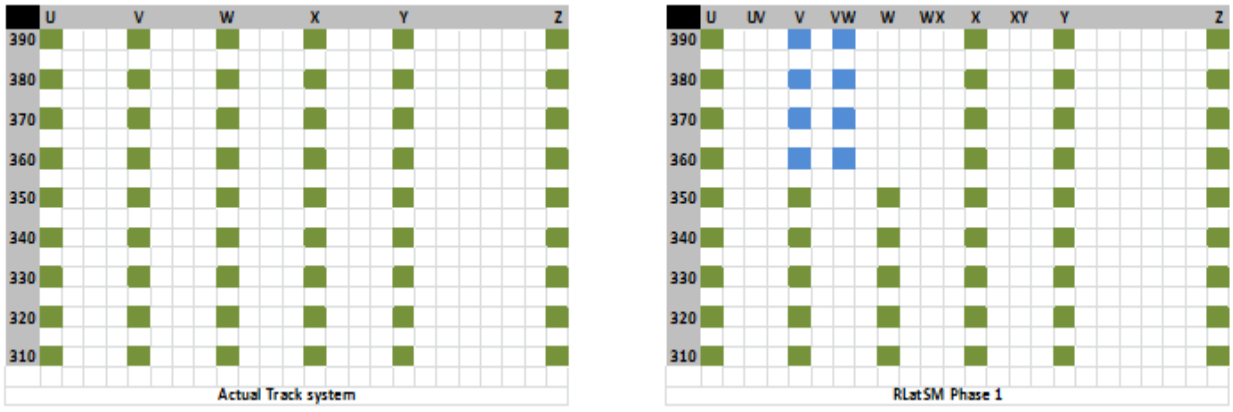


Figure 38: RLatSM Phase 1 track system

The benefits experienced by operators during DLM phase 1 is shown in Figure 39.

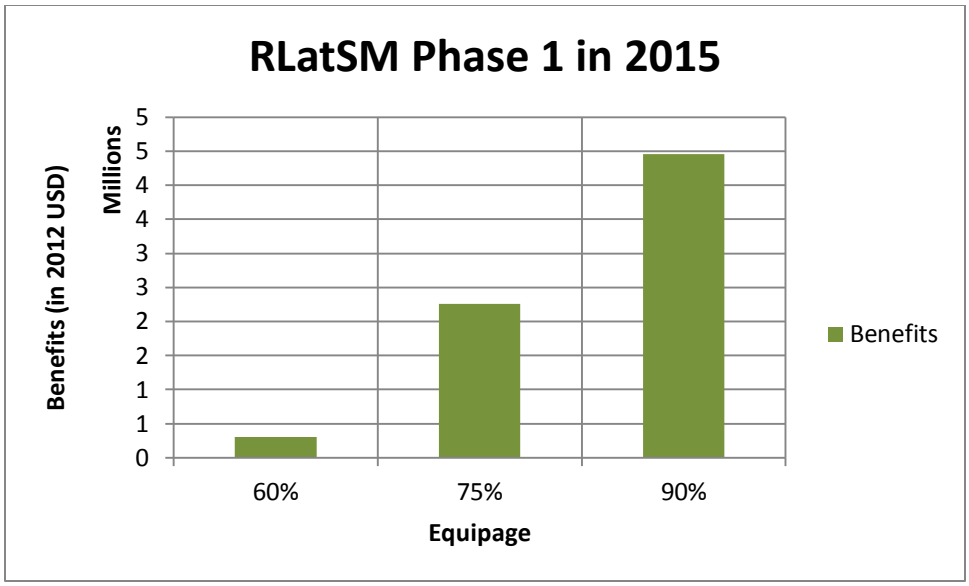


Figure 39: Benefits in RLatSM Phase 1

In RLatSM phase 1 the benefits are modest but positive for all the equipage cases indicating that the reduction in separation allows flights to fly more wind optimum tracks. The penalties experienced due to data link mandate were lessened by the benefits resulting from reduced separation. As the equipage level increases more number of flights are able to fly optimum routes (closely spaced tracks in this case) resulting in higher benefits.

5.5 RLatSM Phase 2

In RLatSM phase 2 the ½° spacing and exclusionary airspace extends to all OTS tracks from FL 360 to FL 390. Three cases 5b, 6b and 7b with equipage levels 60%, 75% and 90% respectively with 2017 traffic levels are studied in this scenario. The benefits for this scenario are shown in Figure 41.

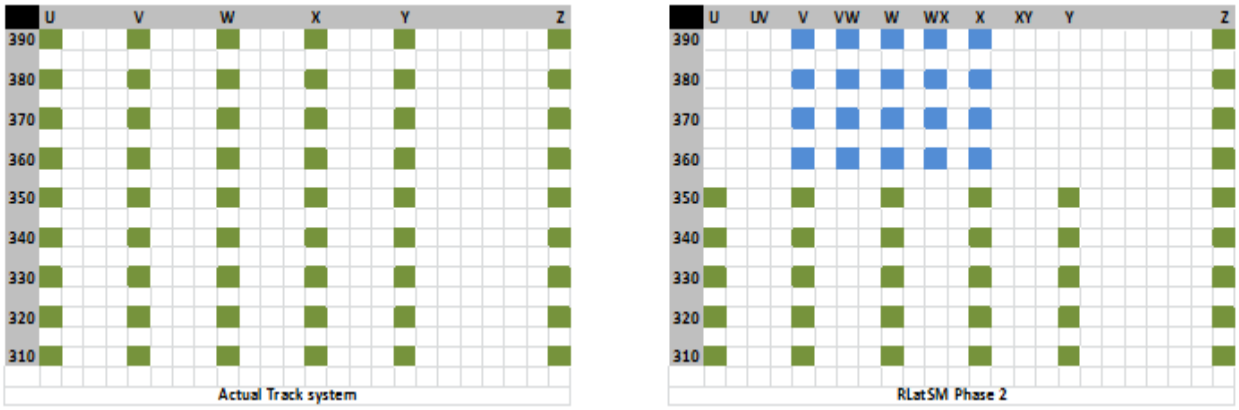


Figure 40: RLatSM Phase 2 track system

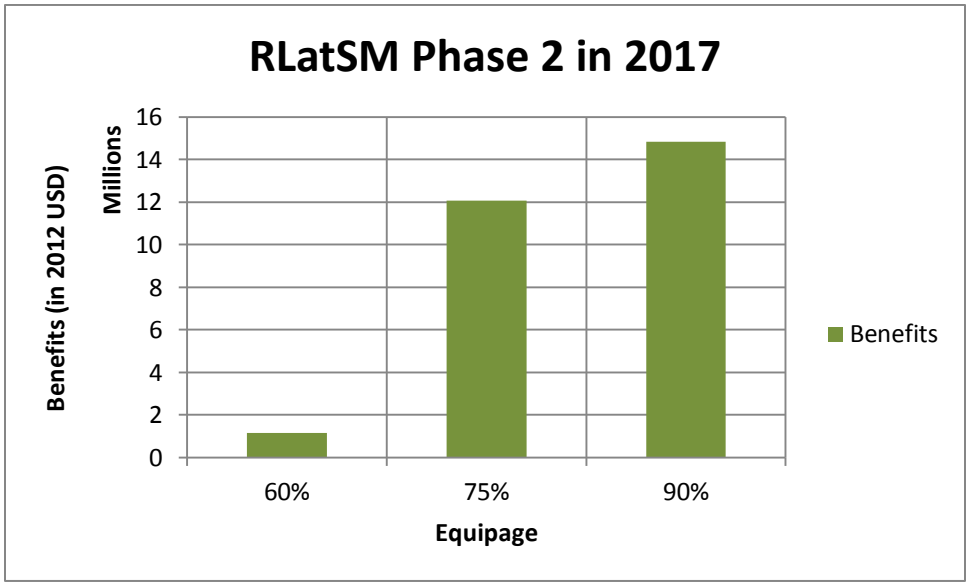


Figure 41: Benefits in RLatSM Phase 2

The benefits RLatSM phase 2 are modest at $1 million in 60E but are significant in 75E and 90E at $10.3 million and $12.6 million respectively. Since all the tracks in this scenario are ½° spaced more number of tracks are designed to be wind optimal, in other words closer to Jet Stream in case of easterly tracks and away from Jet Stream in case of westerly tracks. Also since the track system in RLatSM is compact and occupies less geographic area, this frees up some of the airspace to random flights. This larger random airspace allows more flights to fly direct routes if not great circle paths between their origin and destination. The benefits realized in RLatSM have three components which are:

1. Flights being able to fly more wind optimal tracks in the OTS.

2. Random flights having access to larger airspace and being able to fly more direct routes.
3. Random flights being able to fly optimum flight profiles by performing frequent step climbs.

5.6 DLM Phase 2 & RLatSM Phase 1

DLM and RLatSM are implemented in phases with different time lines, DLM phase 1 starting from February 2013 and RLatSM phase 1 starting from 2015. In 2015 RLatSM phase 1 will converge with DLM phase 2 and in this scenario the combined effect of both CONOPS is studied. Three cases 2ab, 3ab and 4ab with equipage levels 60%, 75% and 90% respectively with 2015 traffic levels are studied in this scenario.

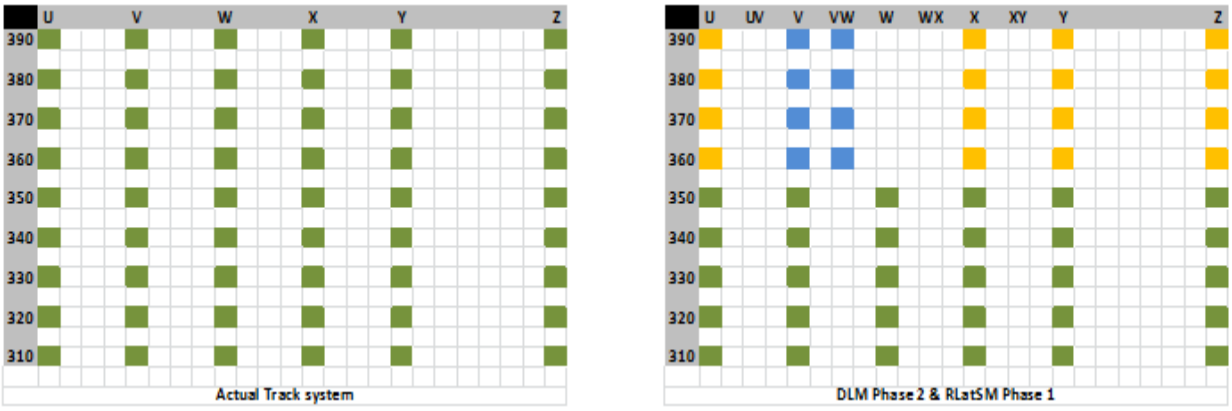


Figure 42: DLM & RLatSM track system

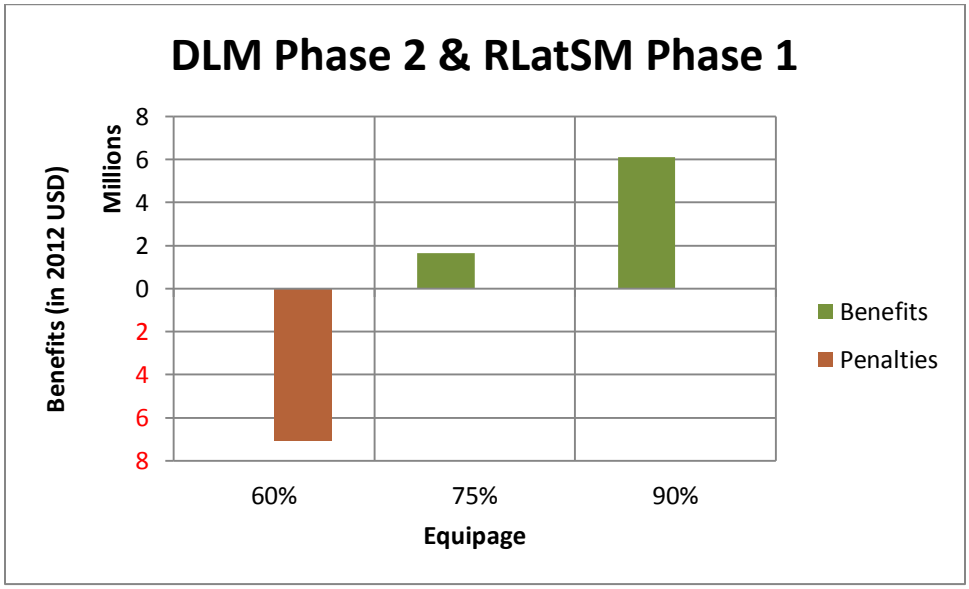


Figure 43: Benefits in DLM Phase 2 & RLatSM Phase 1

In the previous scenarios the CONOPS were studied in isolation but in 2015 both CONOPS DLM and RLatSM are expected to be implemented simultaneously but at different phases. DLM would mature from phase 1 to phase 2 by 2015 whereas RLatSM would be introduced in phase 1. Therefore this scenarios looks at the impact on benefits if these two CONOPS are converged. In 60E and 75E cases negative benefits due to data link mandate are offset by the positive benefits due to RLatSM, however in 90E case benefits in both scenarios add up and provide significant benefits of $6 million.

5.7 DLM Phase 2, RLatSM Phase 1 and RLongSM

Trail runs for RLongSM have already started by NAVCANADA and NATS, so RLongSM is expected to be introduced across all OTS tracks by 2015 along with RLatSM phase 1 and DLM phase 2. This scenarios aims at estimating benefits in converging these three CONOPS in 2015. Three cases 2c, 3c and 4c with equipage levels 60%, 75% and 90% respectively with 2015 traffic levels are studied in this scenario. Reduced separations are applied across all OTS tracks from FL 360 to FL 390.

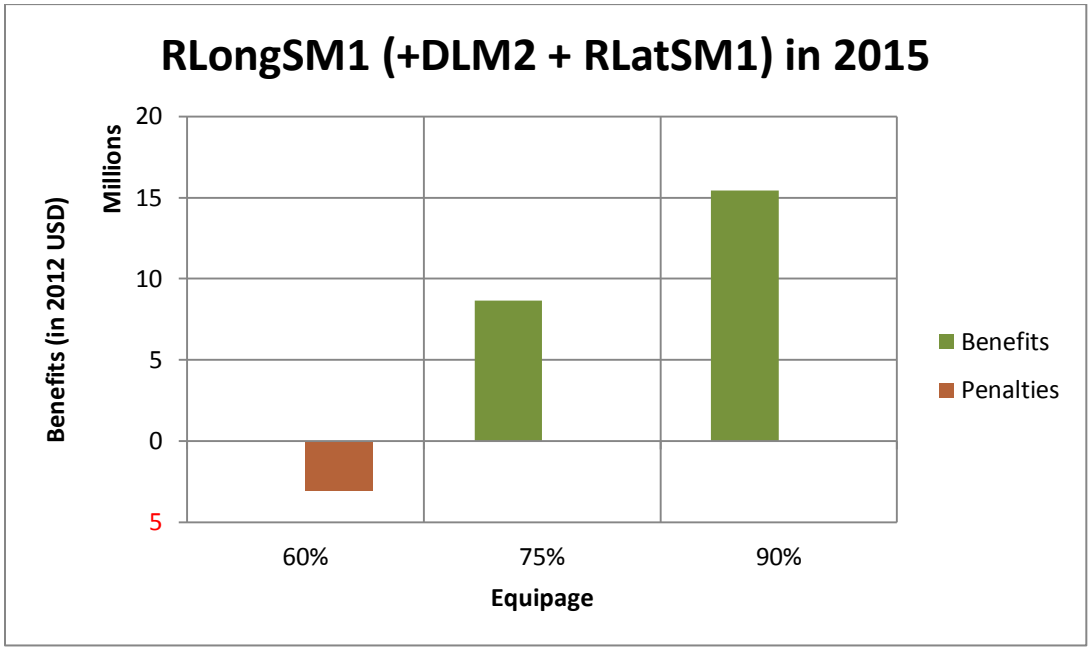


Figure 44: Combined benefits in DLM phase 2, RLatSM phase1 and RLongSM

Even though there are penalties imposed by data link mandate, RLatSM and RLongSM provide enough benefits to nullify the penalties. In isolated DLM phase 2 scenario the penalties for 60E case are $11.6 million whereas they are reduced to $3.1 million in this combined scenario. In 70E case penalties were experienced in DLM phase 2 where as in this combined scenario benefits of $8.6 million were

experienced. Similarly 90E case experiences $15 million benefits while the benefits were only $2.8 million in DLM phase 2.

5.8 Level of Service (LOS)

The level of service is another key performance indicator of the system. In DLM phase 1 the OTS system shows little sensitivity to the mandate because only the two core tracks above FL 360 were made exclusionary. The % of flights assigned to their optimum track & FL remains almost the same in baseline, 60E, 70E and 90E cases. However the % of flights assigned to desired FL drops from 86% in baseline to 83.1% in 60E and gradually recovers to 84.9% in 75E and to 85.5% in 90E. On the other hand the % of flights assigned to desired track increases from 70% in baseline to 72% in 60E but retreats to 71% in 75E and to 70% in 90E. Usually for an OTS flight a vertical deviation of 1000 ft from the optimum altitude results in more penalty than a lateral deviation of 1° from its optimum track.

In baseline case of DLM phase 2-2015 the % of flights assigned to optimum track & FL is 57.5, 3.1% less than the baseline case of DLM phase 1. This is because the mandate is extended to all tracks in 2015 and more all unequipped flights flying at or above FL 360 are forced to move downwards. Moreover there is a slight increase in the traffic levels from 2013 to 2015.

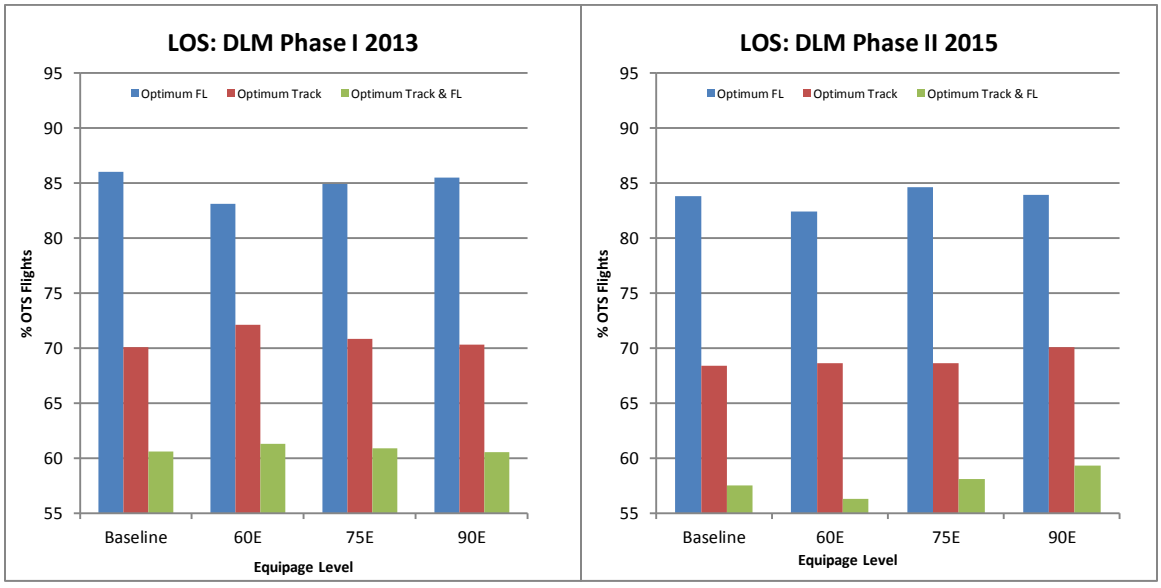


Figure 45: Level of service for DLM I & DLM II

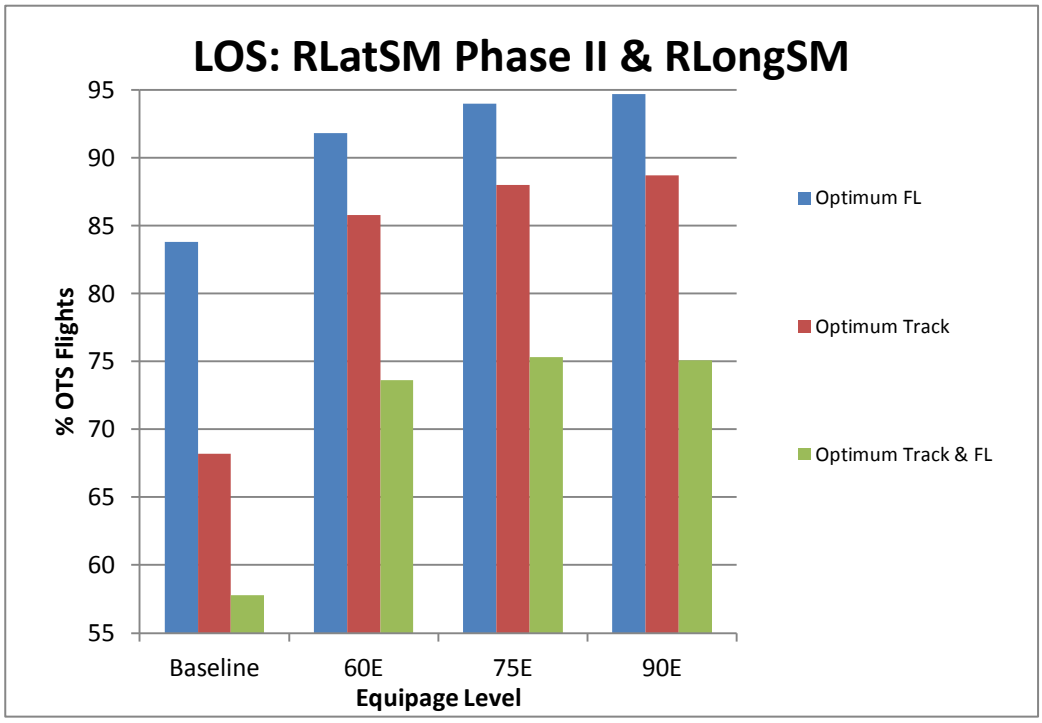


Figure 46: Level of service for RLatSM + RLongSM

In the RLatSM + RLongSM scenario with 2017 traffic, the % of flights assigned to optimum track & FL is 58% in the baseline case. This increasd significantly in the future cases 60E, 75E and 90E to 73.6%, 75.3% and 75.1% respectively. This improvement is because of the increase in the capacity of the system. Reducing the longitudinal separation from 10 minutes to 5 minutes almost doubles the capacity of the OTS and therefore large percent of flights get assigned to their desired track and desired cruise altitudes.

5.9 Summary of Results

In year 2013 only the Data link mandate phase 1 would be introduced in the NAT. This phase has a little negative impact on the system at 60% equipage but has very little or almost zero effect at higher equipage levels.

Substantial changes are anticipated in the NAT airspace in the year 2015 in the form of DLM phase 2, RLatSM phase 1 and RLongSM. The benefits realized by the NAT users in each of the isolated and combined CONOPS scenarios are shown in Figure 47.

CONOPS	Case	Fuel Savings (kg)	Travel Time Savings (Hrs)	Fuel Savings (USD)	Travel Time Savings (USD)	Total Savings
Data Link Phase 1 2013	2a	-2,369,901	-38,584	-2,313,480	-1,678,421	**-3,991,901**
	3a	-1,290,876	-48,289	-1,260,144	-2,100,582	**-3,360,726**
	4a	1,684,948	-10,711	1,644,833	-465,949	**1,178,884**
Data Link Phase 2 2015	5a	-13,267,334	29,411	-12,951,470	1,279,377	**-11,672,093**
	6a	-3,630,529	18,490	-3,544,095	804,328	**-2,739,767**
	7a	1,366,880	34,053	1,334,338	1,481,314	**2,815,651**
Data Link Phase 3 2017	8a	-14,827,737	5,817	-14,474,724	253,035	**-14,221,688**
	9a	-4,242,967	30,865	-4,141,952	1,342,616	**-2,799,336**
	10a	1,801,886	-4,188	1,758,987	-182,187	**1,576,800**

Table 5: Benefits in Data Link scenario

CONOPS	Case	Fuel Savings (kg)	Travel Time Savings (Hrs)	Fuel Savings (USD)	Travel Time Savings (USD)	Total Savings
RLatSM Phase 1 2015	2b	2,371,175	-46,335	2,314,723	-2,015,590	**299,133**
	3b	3,846,989	-34,326	3,755,401	-1,493,192	**2,262,210**
	4b	6,069,567	-33,676	5,925,065	-1,464,925	**4,460,140**
RLatSM Phase 2 2017	5b	-1,018,957	49,664	-994,698	2,160,400	**1,165,702**
	6b	8,679,239	82,458	8,472,607	3,586,942	**12,059,548**
	7b	14,167,705	22,888	13,830,406	995,639	**14,826,045**

Table 6: Benefits in Reduced Lateral scenario

CONOPS	Case	Fuel Savings (kg)	Travel Time Savings (Hrs)	Fuel Savings (USD)	Travel Time Savings (USD)	Total Savings
DLM Phase 2 + RLatSM Phase 1 2015	2ab	-8,930,059	37,715	-8,717,455	1,640,590	**-7,076,866**
	3ab	296,197	30,900	289,145	1,344,164	**1,633,310**
	4ab	4,741,735	34,296	4,628,845	1,491,857	**6,120,702**
RLongSM + DLM Phase1 + RLatSM Phase1 2015	2c	-6,559,291	75,657	-6,403,130	3,291,084	**-3,112,045**
	3c	5,289,962	80,543	5,164,020	3,503,601	**8,667,621**
	4c	11,579,200	94,809	11,303,527	4,124,170	**15,427,697**
RLongSM + DLM Phase 2 + RLatSM Phase 2 2017	5c	5,248,580	95,026	5,123,624	4,133,647	**9,257,270**
	6c	14,315,849	128,562	13,975,023	5,592,454	**19,567,477**
	7c	19,273,743	58,368	18,814,881	2,539,009	**21,353,889**

Table 7: Benefits in converged scenarios

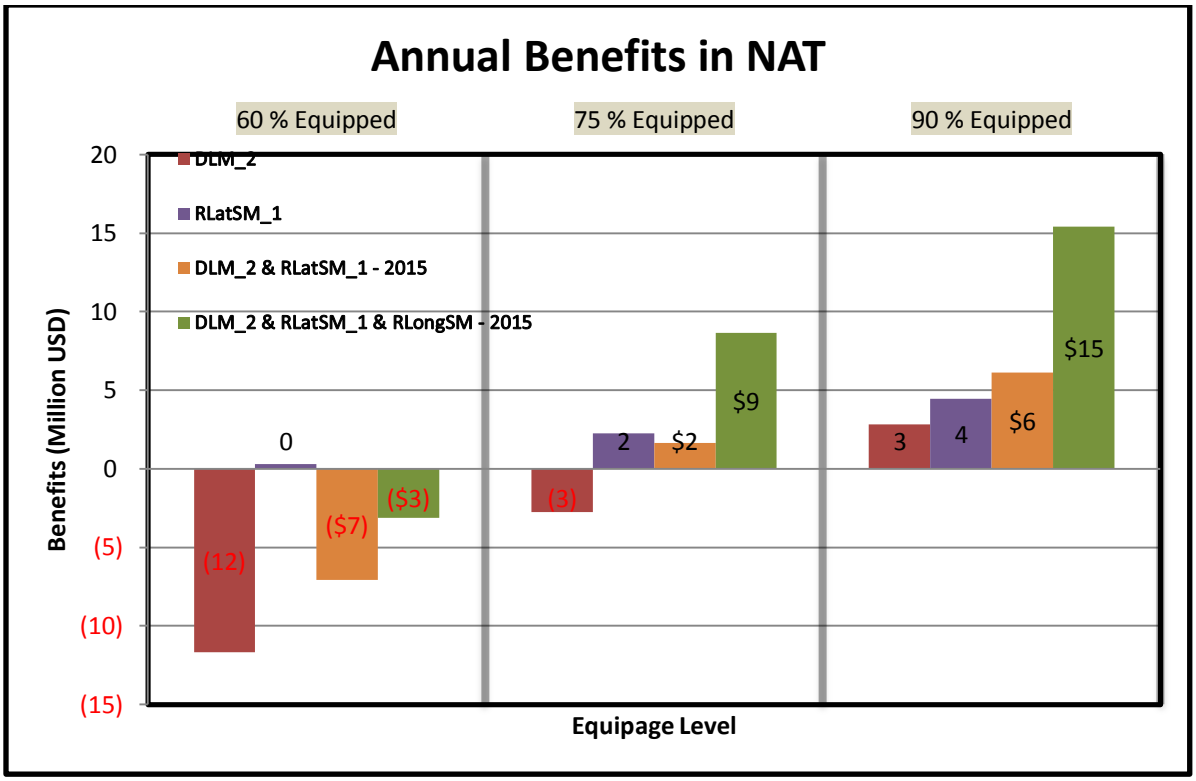


Figure 47: Summary of benefits in year 2015

The year 2015 is cruicial for assessing the impact of Data link mandate and reduced separations because it is the year when all the three CONOPS are expected to converge. In all the scenarios the benefits increase (and penalties decrease) with increasing equipage levels which is as expected. If DLM alone is introduced in 2015 penalties are experienced in 60E and 75E cases while modest benefits are experienced in 90E case. However if RLatSM is also introduced in convergence with DLM then relatively lesser penalties are experienced in 60E case, benefits are experienced in 75E and 90E cases. If RLongSM is also introduced simultaneously in convergence with DLM and RLatSM then significant benefits are experienced in higher equipage cases, while very little penalties are experienced in lower equipage case. While these CONOPS provide incentives to the equipped operators, they also penalize the unequipped operators.

5.10 Conclusions

This study presents a holistic approach to investigate the various impacts of introducing new concepts of operations in the NAT airspace. The results of this research effort explain the nature and magnitude of benefits resulting from the new CONOPS and how they vary with overall equipage levels. The results indicate that unequipped NAT users would experience modest penalties starting from 2013 in DLM phase 1 and the penalties could be more severe in DLM phase 2 starting from 2015 and these penalties become even higher with increasing traffic. On the other hand equipped users are likely to enjoy slight benefits in DLM phase 1 starting from 2013 and modest benefits starting from 2015 due to DLM. Because of Data link mandate, unequipped aircraft are restricted from the exclusionary airspace hence reducing congestion in that airspace. This is similar to creating a HOV lane for equipped aircraft enabling them to fly more desired routes than before. Implementing RLatSM results in modest benefits in its first phase in 2015 and these benefits become significant if implemented in its phase 2 in 2017.RLatSM increases the track's wind optimality and also frees up airspace to random flights. If RLatSM is implemented along with DLM then the overall penalties due to DLM could be offset to some extent with the benefits due to RLatSM. If RLongSM is also implemented simultaneously in convergence with DLM and RLatSM then these overall penalties could be further offset with the benefits from RLongSM. The penalties projected in the lower equipages cases might drive the operators to equip and take part in the technology transition. This complete transition is the key to realize the intended goals, increased safety and increased efficiency.

5.11 Recommendations

Currently the flights modeled in the OTS tracks adhere to a single altitude throughout the NAT track segment which is true in the current NAT OTS. But in the future the new CONOPS like reduced longitudinal separations result in creating a lot of targets of opportunity while on the other hand increased equipage means increased awareness for the pilots. These two factors combinedly enable the flights to perform step climbs even in the OTS tracks and fly more optimized profiles thus saving fuel. Reduced longitudinal separations also produce benefits due to increase in capacity of the OTS and currently only these benefits are captured in the model. This ability to perform step climbs is a major benefit component in introducing reduced longitudinal separation and if the model is capable of modeling this concept then full benefits of RLongSM can be estimated.

One more avenue to improve the model is by adding the capability to calculate the reductions in emissions. Most of the reduction in emissions would be in the cruise phase of the flight and at this altitude cruise NO_x and CO_2 are a major concern.

REFERENCES

1. NAVCANADA. *NAVCANADA.* [Online] [Cited: November 11, 2012.] http://www.navcanada.ca/.

2. **Federal Aviation Administration.** *FAA Aerospace Forecast.* 2012.

3. **Almira R. Williams.** *BENEFITS ASSESSMENT OF REDUCED SEPARATIONS IN NORTH ATLANTIC ORGANIZED TRACK SYSTEM.* CSSI Inc. 2005.

4. **Viviana, Campos Norma.** Encouraging Technology Transition through Value Creation, Capture and Delivery Strategies: The Case of Data Link in the North Atlantic Airspace. 2009.

5. *Economic Impact Analysis of Data Link in the North Atlantic Region.* **Gunnam, Aswin K, Campos, Norma V and Trani, Antonio.** 2012.

6. NCEP/NCAR Reanalysis. *NOAA ESRL Physical Sciences Division.* [Online] http://www.esrl.noaa.gov/psd/.

7. *Characterizing North Atlantic weather patterns for climate-optimal aircraft.* **al., E. A. Irvine et.** s.l. : Wiley Online Library, 2012.

8. **ICAO NAT SPG 48.** *Draft Implementation Plan for the Trial Application of RLatSM in the NAT Region.* 2012.

9. **ICAO NAT SPG 48**. *Implementation plan for the trail application of RLongSM in NAT region.* 2012.

10. **Eurocontrol Experimental Center.** *USER MANUAL FOR THE BASE OF AIRCRAFT DATA (BADA).* 2012. 12/04/10-45.

11. *Advanced aircraft performance modeling for ATM: enhancements to the BADA model.* **Nuic, Angela, Chantal Poinsot, Mihai-George Iagaru, Eduardo Gallo, Francisco A. Navarro, and Carlos Querejeta.** Washington : 24th Digital Avioncs System Conference, 2005.

12. *Modeling of terminal-area airplane fuel consumption.* **Senzig, David A., Gregg G. Fleming, and Ralph J. Iovinelli.** 4, s.l. : Journal of Aircraft, 2009, Vol. 46.

13. https://pilotweb.nas.faa.gov/common/nat.html. [Online]

14. *The NCEP/NCAR 40-year reanalysis project.* **Kalnay, Eugenia, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, Mo Iredell et al.** 3, s.l. : Bulletin of the American meteorological Society, 1996, Vol. 77. 437-471.

15. *Operating in the North Atlantic MNPS Airspace.* **Whelan, Conor.** 01, s.l. : Journal of Navigation, 2000, Vol. 52.

16. **ICAO NAT SPG 48.** *Updated RLatSM Concept of Operations.* 2012.

17. **Stephanie Chung, Joseph Post.** *NATSIM: A system dynamics model of North Atlantic air traffic.* 2008.

18. **Christine M. Gerhardt-Falk, E. A. Elsayed,Dale Livingston,Brian Colamosca.** *Simulation of the North Atlantic Air Traffic and Separation Scenarios.* 2000.

19. https://pilotweb.nas.faa.gov/common/nat.html. [Online]

20. International Civil Aviation Organisation. [Online] http://www.paris.icao.int/.

21. *A Neural Network Model to Estimate Aircraft Fuel Consumption.* **Trani, Antonio, et al., et al.** 2004.

22. **Federal Aviation Administration.** *FAA Aerospace Forecast.* s.l. : FAA, 2012.

23. *Flight Aware.* [Online] http://flightaware.com/.

APPENDIX A: MATLAB CODES FOR NATSAM

NAT OTS Cost Generator

```
% Script to estimate the cost of flying each flight across the NAT for various Fls and track
combinations
% Uses a fixed track structure defined for one day of operations
clear all
clc
InputDir = 'D:\NATSAM_Results\Final_Runs\NATSAM_V1\Switching ExAirspace\';
casename = {'b1-2013'};
%casename = {'5a'};
disp('Check if the track system defined in this script is for correct day')
% pause
%%
for z = 1
%%% Random seed %%%
rng(996)        %%%
%%%%%%%%%%%%%%%%%%%

global Wind badaForOceanic  airportCoordinates airportNames NAT_Tracks flightsNATS_withCost
atmosphere airports_FA_Cell
global BADA_Aircraft_Data BADA_Aircraft_List ETMS_BADA_List Dist_FL_Relation FL_Empirical_CDF
BADA_Aircraft_Coef
global Origin_Airport_Altitude Destination_Airport_Altitude Assn_Cruise_FL
global FeetInNauticalMiles Track_vs_Cruise Course Waypoints
global Wind_Boolean Wind Wind_Parameters Month
global time_increment Aircraft_Index TakeOff_Mass First_Iteration InterpWind

InterpWind = 0;
load([InputDir, char(casename(z)),'\flight.mat'])
load badaForOceanic
load airports_FA_Cell
load ('D:\NAT Track Files\Nat_Tracks_20080726')
load ('D:\Wind Data\NATSIM Wind Data\July_26\OTSWind.mat')
load ('D:\Wind Data\NATSIM Wind Data\July_26\Wind.mat')
load atmosphere
load Dist_FL_Relation
load BADA_Aircraft_Data.mat
load BADA_Aircraft_List.mat
load BADA_Aircraft_Coef.mat
% load OTS_Polygon

FeetInNauticalMiles = 6076.11549;
Detour = 1.05
Wind_Boolean = false;
ExcAirSpace = 0;
noFlights = length(flight);
counter = 0;


%% % % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% specific track structure for July 26
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    trackArrayEast = {'NATT'; 'NATU'; 'NATV'; 'NATW'; 'NATX'; 'NATY'; 'NATZ'; 'NATUV'; 'NATVW';
'NATWX'; 'NATXY'};
    trackArrayWest = {'NATA'; 'NATB'; 'NATC'; 'NATD'; 'NATE'; 'NATF'; 'NATG'; 'NATH';
'NATJ';'NATAB'; 'NATCD'; 'NATDE'; 'NATEF'; 'NATFG'; 'NATGH'};
% % % %% %%%%%%%%%%%%%%%%%%%%%%%%%%Specific Reduced Lateral track system for March 15
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % %
%     trackArrayEast = {'NATU'; 'NATV'; 'NATW'; 'NATX'; 'NATY'; 'NATZ'; 'NATUV'; 'NATVW';
'NATWX'; 'NATXY'};
%     trackArrayWest = {'NATA'; 'NATB'; 'NATC'; 'NATD'; 'NATE'; 'NATF';;'NATAB'; 'NATCD';
'NATDE'; 'NATEF'};
% % %
% % %
%%
NoTracksEast = numel(trackArrayEast);
NoTracksWest = numel(trackArrayWest);
flights_Invalid = [];
```

```
comments = {};comment = 1;
%% Enter computational loop
tic;
for i=1:noFlights
    if mod(i,10) == 0
        save([InputDir, char(casename(z)),'\flights_NATcost.mat'],'flight')
    end
    aircraftType = flight(i).originalAircraft;
    disp(['Calculating aircraft ', num2str(i) ])
    disp(aircraftType)
    Direction = flight(i).direction;
    % _________Make origin & destination airport elevations equal to zero ____________
    Origin_Airport_Altitude = 0; Destination_Airport_Altitude = 0;

    % Find the index of the aircraft in the BADA Table

    [aircraftPosition] = findAircraft(aircraftType);
    Act_BADA_Code = [aircraftType,'__'];
    Aircraft_Index = strcmp(Act_BADA_Code,BADA_Aircraft_List);
    if ~sum(Aircraft_Index)
        comments(comment).info = char(['aircraft ',aircraftType, ' not found', ' for flight ',
num2str(i)]);   comment=comment+1;
        continue
    end
    flight(i).aircraftPosition = aircraftPosition;

    %% Calculate TO mass for each flight
    TakeOff_Mass = estimateTO_mass(flight(i).origin, flight(i).destination,aircraftType);
    if isnan(TakeOff_Mass)
        flights_Invalid = [flights_Invalid i];
        comments(comment).info = char(['flight',num2str(i), ' TO_mass is NaN']);
comment=comment+1;
        continue
    end
    flight(i).TOmass = TakeOff_Mass;

    if strcmp(Direction,'Easterly') == 1
            trackArray = trackArrayEast;
            noTracks = length(trackArray);
            Heading = 'EAST';
    elseif strcmp(Direction,'Westerly') == 1
            trackArray = trackArrayWest;
            noTracks = length(trackArray);
            Heading = 'WEST';
    end

    flightLevelsNAT = 31000:1000:39000;
    startFLIndex = 1;
    noFlightLevels = 9;
%%        find the GCD distance between Origin and Destination.%%%%%%%%%%%%%%%%%%%%%%
        Route_Dist = Detour*flight(i).GC_Dist;
        %Fuelused to climb to typical altitude
        typ_alt = badaForOceanic(aircraftPosition).typicalAltitude/3.28;
        typ_fuelClimb =
interp1(badaForOceanic(aircraftPosition).hiMassClimb(:,3),badaForOceanic(aircraftPosition).hiMass
Climb(:,4),typ_alt);
        typ_mass = TakeOff_Mass - typ_fuelClimb/2;
        PreNATalt = floor(maximumOperationalAltitude(aircraftType,typ_mass)/1000)*1000;
        clear typ_alt typ_fuelClimb typ_mass
        % ---------------------------Assigned a FL based on hemispherical rules. ----------------
-
        Max_Cruise_FL = badaForOceanic(aircraftPosition).maxAltitude;
        Assn_Cruise_FL = Calculate_FL(PreNATalt, Heading, Max_Cruise_FL);
        flight(i).PreNATalt = Assn_Cruise_FL;
        %
_________________________________________________________________________________________
%         [Intersects Fuel] =
Does_GCPath_Intersects_ExclAirspace(Aircraft_Index,Direction,flight(i).origin,flight(i).destinati
on,flight(i).PreNATalt,OTS_Polygon)
```

```
%% Loop aroound  Tracks
        for j=1:noTracks

 %% Loop around FLs

           for k=startFLIndex:noFlightLevels
               FlightLevel = flightLevelsNAT(k);
               if j == 1 && k == 1
                   First_Iteration = 1; else First_Iteration = 0;
               end
               InterpWind = 1;

[optAltitudeAtNAT,FuelClimb1,FuelCruise1,FuelClimb2,timeClimb1,TimeCruise1,TimeClimb2,DistClimb1,
DistCruise1,DistClimb2,MassNAT,WindToNat] = ...
                estimate_Journey_To_entryPoint(flight(i).origin,trackArray{j},aircraftType,...

TakeOff_Mass,Assn_Cruise_FL,FlightLevel,badaForOceanic(aircraftPosition).typicalMach,Direction);
                if strcmp(Direction,'Easterly')
                    track/flight level combination_wind = NATWind_East;
                else
                    track/flight level combination_wind = NATWind_West;
                end

                % find cost inside NAT
                InterpWind = 0;
               [optAltitudeEndNAT,routeDistanceInNAT,travelTimeNAT, fuelUsedNAT, finalMassNAT,
WindInNat] = ...
                estimate_Journey_Inside_NAT(trackArray{j},aircraftType,...

MassNAT,FlightLevel,badaForOceanic(aircraftPosition).typicalMach,Direction,track/flight level
combination_wind(j,k));
            % check if the aircraft is able to climb
            Dist_covered = DistClimb1 + DistCruise1 + DistClimb2 +
routeDistanceInNAT*1.07;%DistCruise1 is factor (*1.07) but routeDistanceInNAT is not. So
factoring it here.
            Dist_Left = Route_Dist - Dist_covered;
            %%

            [FlightLevel,DistClimb3,TimeClimb3, FuelClimb3, finalMassNAT] = ...
                estimate_stepClimb_after_NAT(Dist_Left,aircraftType,...
                finalMassNAT,FlightLevel,badaForOceanic(aircraftPosition).typicalMach,WindInNat);
% passed 'WindInNat' to be used in climb

            % Find cost from NAT to destination

            %%
                InterpWind = 1;

[DistCruise3,TravelTimeFromNAT,FuelUsedFromNAT,massTOD,fuelUsedDescent,DistDescend,timeToDescend,
WindFromNat] = ...

estimate_Journey_From_exitPoint(flight(i).destination,trackArray{j},aircraftType,...

finalMassNAT,FlightLevel,badaForOceanic(aircraftPosition).typicalMach,Direction,DistClimb3);

            % Save the costs needed for future computations

            totalFuel = FuelClimb1 + FuelCruise1 + FuelClimb2 + fuelUsedNAT + FuelClimb3 +
FuelUsedFromNAT + fuelUsedDescent;
            totalTime = TimeCruise1 + timeClimb1/60 + TimeClimb2/60 + travelTimeNAT +
TimeClimb3/60 + TravelTimeFromNAT + timeToDescend/60;
            totalDistance = DistClimb1 + DistCruise1 + DistClimb2 + routeDistanceInNAT +
DistClimb3 + DistCruise3 + DistDescend;
            NatEntryTime = flight(i).DepTime + timeClimb1/60 + TimeCruise1;% + TimeClimb2/60;
%% All Entry times
            flight(i).EntryTime(10-k,j) = NatEntryTime;
%%

%%%%%%%%%%%%%%%%%%%%%%%% NOTE  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  Indexing of some of the cost matrices is done as (10-k,j) which
%%%  means columns represent tracks and rows represent FL (FL 390 is first
```

```
%%%  row and FL 310 is last row)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % accumulate wind data for each segment
            flight(i).Wind(10-k,j) = WindInNat; % a vector containing wind in three segments of
flight
            flight(i).Wind2Nat(10-k,j) = WindToNat;
            flight(i).WindFromNat(10-k,j) = WindFromNat;

            %% Check if the altitude is feasible and if altitude is unfeasible - write 999999

            if optAltitudeAtNAT < flightLevelsNAT(k)
                flight(i).costFuel(j,k) = 999999;
                flight(i).costTime(j,k) = 999999;
                flight(i).costDistance(j,k) = 999999;
                flight(i).FuelClimb1(10-k,j) = 999999;  flight(i).FuelCruise1(10-k,j) = 999999;
flight(i).FuelClimb2(10-k,j) = 999999;   flight(i).fuelUsedNAT(10-k,j) = 999999;
                flight(i).FuelClimb3(10-k,j) = 999999; flight(i).FuelUsedFromNAT(10-k,j) =
999999;   flight(i).fuelUsedDescent(10-k,j) = 999999;

                flight(i).TimeCruise1(10-k,j) = 999999;  flight(i).timeClimb1(10-k,j) = 999999;
flight(i).TimeClimb2(10-k,j) = 999999;flight(i).travelTimeNAT(10-k,j) = 999999;
                flight(i).TimeClimb3(10-k,j) = 999999; flight(i).TravelTimeFromNAT(10-k,j) =
999999;   flight(i).timeToDescend(10-k,j) = 999999;

                flight(i).DistClimb1(10-k,j) = 999999;  flight(i).DistCruise1(10-k,j) = 999999;
flight(i).DistClimb2(10-k,j) = 999999;  flight(i).routeDistanceInNAT(10-k,j) = 999999;
                flight(i).DistClimb3(10-k,j) = 999999; flight(i).DistCruise3(10-k,j) = 999999;
flight(i).DistDescend(10-k,j) = 999999;
                %flight(i).NatEntryTime = 999999; %Since this is initialized at begining instead
of 999999 zero is populated in the cells.
                % A value of 999999 makes the average of all NAT entry times equal to 999999.
            else
                flight(i).costFuel(j,k) = totalFuel;    flight(i).costTime(j,k) = totalTime;
flight(i).costDistance(j,k) = totalDistance;

                flight(i).FuelClimb1(10-k,j) = FuelClimb1;  flight(i).FuelCruise1(10-k,j) =
FuelCruise1; flight(i).FuelClimb2(10-k,j) = FuelClimb2;flight(i).fuelUsedNAT(10-k,j) =
fuelUsedNAT;
                flight(i).FuelClimb3(10-k,j) = FuelClimb3; flight(i).FuelUsedFromNAT(10-k,j) =
FuelUsedFromNAT;   flight(i).fuelUsedDescent(10-k,j) = fuelUsedDescent;

                flight(i).TimeCruise1(10-k,j) = TimeCruise1;  flight(i).timeClimb1(10-k,j) =
timeClimb1; flight(i).TimeClimb2(10-k,j) = TimeClimb2;flight(i).travelTimeNAT(10-k,j) =
travelTimeNAT;
                flight(i).TimeClimb3(10-k,j) = TimeClimb3; flight(i).TravelTimeFromNAT(10-k,j) =
TravelTimeFromNAT;   flight(i).timeToDescend(10-k,j) = timeToDescend;

                flight(i).DistClimb1(10-k,j) = DistClimb1;  flight(i).DistCruise1(10-k,j) =
DistCruise1;  flight(i).DistClimb2(10-k,j) = DistClimb2;  flight(i).routeDistanceInNAT(10-k,j) =
routeDistanceInNAT;
                flight(i).DistClimb3(10-k,j) = DistClimb3; flight(i).DistCruise3(10-k,j) =
DistCruise3;   flight(i).DistDescend(10-k,j) = DistDescend;

            end


           end


        end
        %%%Remove all zeros from the flight(i).EntryTime and calculate mean.
        flight(i).EntryTime(flight(i).EntryTime==0) = [];
        flight(i).NatEntryTime = mean(mean(flight(i).EntryTime));

        disp('Completed all flight levels')
        disp(' ')


end
```

```
toc
flight(flights_Invalid) = [];
save([InputDir, char(casename(z)),'\flights_NATcost.mat'],'flight')
end

%% display the comments observed during the cost calculations
for comment = 1:numel(comments)
    disp(comments(comment).info)
end
```

Random Cost Generator

```
%%% calculate random flight fuel cost for all flights.
clear
clc
InputDir = 'D:\NATSAM_Results\Final_Runs\NATSAM_V1\Switching ExAirspace\';
casename = {'bl-2017'; '5b'; '6b'; '7b'; 'bl-2050'};

global badaForOceanic  Dist_FL_Relation  airportNames airportCoordinates Wind
atmosphere Dist_FL_Relation airports_FA_Cell
global Detour Prob_4thClimb Prob_3rdClimb Prob_2ndClimb Max_Cruise_FL
load badaForOceanic
load atmosphere
load ('D:\Wind Data\NATSIM Wind Data\July_26\Wind.mat')
load Dist_FL_Relation
load airports_FA_Cell
Prob_dist_2ndClimb_forCases = [1 1 1 1 1];Prob_dist_3rdClimb_forCases = [0.12
0.12 0.20 0.30 0.12]; Prob_dist_4thClimb_forCases = [0.12 0.12 0.20 0.30
0.12];
Detour = 1.04;

for z = 1:4
    rng(997)
load([InputDir, char(casename(z)),'\flightsNATS_Final.mat'])
 Prob_2ndClimb = Prob_dist_2ndClimb_forCases(z);
 Prob_3rdClimb = Prob_dist_3rdClimb_forCases(z);
 Prob_4thClimb = Prob_dist_4thClimb_forCases(z);
for i = 1:numel(flight)
    disp(['Calculating flight ',num2str(i), '   *** ', char(casename(z))])
    if mod(i,50) == 0
        save([InputDir,
char(casename(z)),'\flightsNATS_RNDcost.mat'],'flight')
    end

    %%
            aircraftType = flight(i).originalAircraft;
        [aircraftPosition] = findAircraft(aircraftType);

        typ_alt = badaForOceanic(aircraftPosition).typicalAltitude/3.28;
        typ_fuelClimb =
interp1(badaForOceanic(aircraftPosition).hiMassClimb(:,3),badaForOceanic(airc
raftPosition).hiMassClimb(:,4),typ_alt);
        typ_mass = flight(i).TOmass - typ_fuelClimb/2;
        PreNATalt =
floor(maximumOperationalAltitude(aircraftType,typ_mass)/1000)*1000;
        clear typ_alt typ_fuelClimb typ_mass
        Max_Cruise_FL = badaForOceanic(aircraftPosition).maxAltitude;
        if strcmp(flight(i).direction,'Easterly')
```

```
            Heading = 'EAST';
        else Heading = 'WEST'; end
        Assn_Cruise_FL = Calculate_FL(PreNATalt, Heading, Max_Cruise_FL);
%%
[totalFuel totalTravelTime routeDistance AvgCruiseFLs allFLs avgWind] =
CalculateRandomFlight(flight(i).origin,flight(i).destination,flight(i).origin
alAircraft,...

flight(i).TOmass,Assn_Cruise_FL,badaForOceanic(aircraftPosition).typicalMach,
flight(i).direction);
%                       flight(i).RNDTrack = 'OTS2RND';
                    flight(i).RNDFL_avg = AvgCruiseFLs;
                    flight(i).RNDFuelBurned = totalFuel;
                    flight(i).RNDDistTravelled = routeDistance;
                    flight(i).RNDTimeTravelled = totalTravelTime;
                    flight(i).RNDavgWind = avgWind;
                    flight(i).allFLs = allFLs;

end
save([InputDir, char(casename(z)),'\flightsNATS_RNDcost.mat'],'flight')

%%%%%%%%%%Assigning RND/OTS flags to all flights based on their NAT Entry
Time %%%%%%%%%%%%%%%
%%% OTS validity times %%%%%%
Eastbound_Start = [0 1350];  Eastbound_End = [420 1770];
Westbound_Start = 570; Westbound_End = 1050;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
count = 0;countRnd = 0;
for i = 1:numel(flight)
        SimulationTime_UTC = flight(i).NatEntryTime;
         Direction = flight(i).direction;
    if strcmp(Direction,'Easterly') == 1 %&& strcmp(flight(i).RteType,'RND')
== 0
%         ExcAirSpace = ExcAirSpace_Actual;
        if SimulationTime_UTC <= Eastbound_Start(1)
            OTS=0;countRnd = countRnd+1;
            flight(i).Time_type = 'RND';
        elseif SimulationTime_UTC <= Eastbound_End(1)
            OTS=1;count = count+1;
            flight(i).Time_type = 'OTS';
        elseif SimulationTime_UTC <= Eastbound_Start(2)
                OTS=0;countRnd = countRnd+1;
                flight(i).Time_type = 'RND';
        elseif SimulationTime_UTC <= Eastbound_End(2)
                OTS=1;count = count+1;
                flight(i).Time_type = 'OTS';
        else
         OTS=0;countRnd = countRnd+1;
         flight(i).Time_type = 'RND';
        end
    else
        if SimulationTime_UTC <= Westbound_Start || SimulationTime_UTC >=
Westbound_End
            OTS=0;countRnd = countRnd+1;
            flight(i).Time_type = 'RND';
        else
```

```
            OTS=1;count = count+1;
            flight(i).Time_type = 'OTS';
        end
    end
end
save([InputDir,char(casename(z)),'\flightsNATS_RNDcost'],'flight')
end
disp(['Total OTS flights ',num2str(count)])
disp(['Total RND flights ',num2str(countRnd)])
```

Wind Module

```
function [wind_vect] = getwindvect(altitude,lat,lon)
global Wind

%% calculate the correct wind vectors
 altlen = length(altitude);
    [Alt_mb] = changeAltTOmillibars(altitude);
    for k = 1:length(Alt_mb)
         [val AltInd(k)] = min(abs(Wind.Level - Alt_mb(k)));
    end
    tempind = find(lon<0);
    lon(tempind) = lon(tempind) +360;
    tempind = find(lon>357.5);
    lon(tempind) = 0;

    for i =1:numel(lat)
              Uwind(i) =
      interp3(Wind.Lat,Wind.Lon,Wind.Level,Wind.Uwnd,lat(i),lon(i),Alt_mb);
              Vwind(i) =
      interp3(Wind.Lat,Wind.Lon,Wind.Level,Wind.Vwnd,lat(i),lon(i),Alt_mb);
    end
        Uwind = Uwind';
        Vwind = Vwind';

     if length(lat) < 2
         wind_vect = 0;
         return
     end
     [Course dist] = legs(lat,lon);
     dist = dist';

     Direction_Vector = [cosd(Course), sind(Course)];
Wind_Vector = [Vwind(1:end-1), Uwind(1:end-1)];

% PROJECTION ONTO Direction_vector
for w = 1:length(Course)
      wind_proj(w) = dot(Direction_Vector(w,:), Wind_Vector(w,:)) /
      (norm(Direction_Vector(w,:)));
end
```

```
wind_vect = 1.9438*sum(wind_proj.*dist)/sum(dist);      % 1.9438 is the
conversion factor for M/s to knots.

end
```

Initial Climb And Cruise From TOC To NAT Entry

```
% Function to estimate the takeoff mass of the aircraft
% for three types of aircraft

% Inputs:
%
% 1) origin airport (4 letter)
% 2) NAT Track
% 3) aircraft type (3 types only)
% 4) Takeoff mass (kg)
% 5) Initial cruise_alt to reach NAT (estimated) in feet
% 6) Mach number
% 7) set of winds speeds (knots)
% 8) vector of cruise_alts for wind (feet)

% Outputs

% 1) optimal cruise_alt to enter NAT
% 2) route Distance to NAT
% 3) travel time to NAT (minutes) - accounts for climb time
% 4) initial entry mass at NAT (kg)
% routeDistanceToNAT,TravelTimetoNAT,FuelUsedtoNAT,
initialMassNAT,FuelClimb1,timeToClimb,DistClimb1
function
[optimalcruise_alt,FuelClimb1,FuelCruise1,FuelClimb2,timeClimb1,TimeCruise1,TimeClimb2,DistClimb1
,DistCruise1,DistClimb2,initialMassNAT,Wind] = ...
            estimate_Journey_To_entryPoint(origin,NAT_Track,aircraftType,...
            TOmass,cruise_alt,FinalAlt,machNumber,Direction)

global badaForOceanic  airportCoordinates airportNames NAT_Tracks  InterpWind
global Aircraft_Index Max_Cruise_FL Heading TakeOff_Mass Waypoints
global First_Iteration Common_Climb_Profile
%check if the atlitudes are achievable
%  load badaForOceanic
%  load airports_FA_Cell
% load NAT_Tracks_Capacity

badaData = badaForOceanic;

detourFactor= 1.05;                         % adds 0% to the GCD Distance to plan for
contingencies
buffetBoundarycruise_alt = 3500;        % feet from maximum cruise_alt

 % Find airport information

flagForOriginAirport = strcmp(airportNames,origin);                          % finds position of
origin airport
indexOrigin = find(flagForOriginAirport);
latlonOrigin = airportCoordinates(indexOrigin,:);                              % coordinates of
origin airport


% Match the track used
if strcmp(Direction,'Easterly') == 1

        if strcmp(NAT_Track,'NATT')==1
            latlonDestination = [NAT_Tracks.NATT.lat(1) NAT_Tracks.NATT.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATU')==1
            latlonDestination = [NAT_Tracks.NATU.lat(1) NAT_Tracks.NATU.lon(1)];       %
coordinates of entry point for track
```

```
        elseif strcmp(NAT_Track,'NATV')==1
            latlonDestination = [NAT_Tracks.NATV.lat(1) NAT_Tracks.NATV.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATW')==1
            latlonDestination = [NAT_Tracks.NATW.lat(1) NAT_Tracks.NATW.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATX')==1
            latlonDestination = [NAT_Tracks.NATX.lat(1) NAT_Tracks.NATX.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATY')==1
            latlonDestination = [NAT_Tracks.NATY.lat(1) NAT_Tracks.NATY.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATZ')==1
            latlonDestination = [NAT_Tracks.NATZ.lat(1) NAT_Tracks.NATZ.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATUV')==1
            latlonDestination = [NAT_Tracks.NATUV.lat(1) NAT_Tracks.NATUV.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATTU')==1
            latlonDestination = [NAT_Tracks.NATTU.lat(1) NAT_Tracks.NATTU.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATVW')==1
            latlonDestination = [NAT_Tracks.NATVW.lat(1) NAT_Tracks.NATVW.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATWX')==1
            latlonDestination = [NAT_Tracks.NATWX.lat(1) NAT_Tracks.NATWX.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATXY')==1
            latlonDestination = [NAT_Tracks.NATXY.lat(1) NAT_Tracks.NATXY.lon(1)];       %
coordinates of entry point for track
        end
elseif strcmp(Direction,'Westerly') == 1

        if strcmp(NAT_Track,'NATA')==1
            latlonDestination = [NAT_Tracks.NATA.lat(1) NAT_Tracks.NATA.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATB')==1
            latlonDestination = [NAT_Tracks.NATB.lat(1) NAT_Tracks.NATB.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATC')==1
            latlonDestination = [NAT_Tracks.NATC.lat(1) NAT_Tracks.NATC.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATD')==1
            latlonDestination = [NAT_Tracks.NATD.lat(1) NAT_Tracks.NATD.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATE')==1
            latlonDestination = [NAT_Tracks.NATE.lat(1) NAT_Tracks.NATE.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATF')==1
            latlonDestination = [NAT_Tracks.NATF.lat(1) NAT_Tracks.NATF.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATG')==1
            latlonDestination = [NAT_Tracks.NATG.lat(1) NAT_Tracks.NATG.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATH')==1
            latlonDestination = [NAT_Tracks.NATH.lat(1) NAT_Tracks.NATH.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATJ')==1
            latlonDestination = [NAT_Tracks.NATJ.lat(1) NAT_Tracks.NATJ.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATDE')==1
            latlonDestination = [NAT_Tracks.NATDE.lat(1) NAT_Tracks.NATDE.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATEF')==1
            latlonDestination = [NAT_Tracks.NATEF.lat(1) NAT_Tracks.NATEF.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATFG')==1
            latlonDestination = [NAT_Tracks.NATFG.lat(1) NAT_Tracks.NATFG.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATGH')==1
```

```
            latlonDestination = [NAT_Tracks.NATGH.lat(1) NAT_Tracks.NATGH.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATAB')==1
            latlonDestination = [NAT_Tracks.NATAB.lat(1) NAT_Tracks.NATAB.lon(1)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATCD')==1
            latlonDestination = [NAT_Tracks.NATCD.lat(1) NAT_Tracks.NATCD.lon(1)];       %
coordinates of entry point for track
        end
end

    if isempty(NAT_Track) == 1
        ME = MException('TrackNotAssigned','Track is not assigned: Check NAT_Tracks.mat weather
the tracks exists or not')
        throw(ME)
    end

% Generate a GCD traversal from Origin to Destination
noWaypoints = 30;
[DistanceGCD, latGCDRoute, lonGCDRoute] =
generateGCD(latlonOrigin(1),latlonOrigin(2),latlonDestination(1), latlonDestination(2),
noWaypoints);
[course_temp GCDDist] = legs(latGCDRoute,lonGCDRoute);
cumGCDDist = cumsum(GCDDist);
Waypoints(:,1) = latGCDRoute; Waypoints(:,2) = lonGCDRoute;

% Select aircraft index from BADA data set

    [aircraftPosition] = findAircraft(aircraftType);
% These are climb 2 calculations and are done in advance
FinalAlt_m = FinalAlt/3.28;   cruise_alt_m = cruise_alt/3.28;
if FinalAlt_m-cruise_alt_m > 0
    DistFinal =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,2)
,FinalAlt_m);
    DistInitial =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,2)
,cruise_alt_m);
    fuelFinal =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,4)
,FinalAlt_m);
    fuelInitial =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,4)
,cruise_alt_m);
    TimeFinal =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,1)
,FinalAlt_m);
    TimeInitial =
interp1(badaData(aircraftPosition).nominalClimb(:,3),badaData(aircraftPosition).nominalClimb(:,1)
,cruise_alt_m);
else
    DistFinal =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,2),FinalAlt_m);
    DistInitial =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,2),cruise_alt_m);
    fuelFinal =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,4),FinalAlt_m);
    fuelInitial =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,4),cruise_alt_m);
    TimeFinal =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,1),FinalAlt_m);
    TimeInitial =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,1),cruise_alt_m);
end
```

```
DistClimb2 = abs(DistFinal- DistInitial); FuelClimb2 = abs(fuelFinal-fuelInitial); TimeClimb2 =
abs(TimeFinal-TimeInitial);


routeDistanceToNAT = DistanceGCD * detourFactor - DistClimb2;              % in nm



 % Do the calculations to reach the TOC point (climb
          % calculations)
    if First_Iteration
        Climb_Profile = Generate_Climb_Profile(Aircraft_Index,TakeOff_Mass);
        FuelClimb1 = Climb_Profile.Total_Fuel_kg;
        timeClimb1 = Climb_Profile.Time_For_Climb_hrs*3600; %timeClimb1 is in secs
        DistClimb1 = Climb_Profile.Distance_For_Climb_nm;
        Common_Climb_Profile = Climb_Profile;
    else
        FuelClimb1 = Common_Climb_Profile.Total_Fuel_kg;
        timeClimb1 = Common_Climb_Profile.Time_For_Climb_hrs*3600;
        DistClimb1 = Common_Climb_Profile.Distance_For_Climb_nm;
    end
    currentmass = TOmass-FuelClimb1;

     [tem, tempIndex] = min(abs(cumGCDDist - DistClimb1));
     TOC_Ind = tempIndex+1;
     latGCDRoute_cruise = latGCDRoute(TOC_Ind:end);
     lonGCDRoute_cruise = lonGCDRoute(TOC_Ind:end);
    maxAltAtcurrentMass = maximumOperationalAltitude(aircraftType,currentmass);
    maxAltAtcurrentMass = floor(maxAltAtcurrentMass/1000) * 1000;
    distanceToCover = routeDistanceToNAT-DistClimb1-DistClimb2;
    if distanceToCover < 0
        distanceToCover = 0
    end
         [FuelCruise1,TimeCruise1,Wind,DistCruise1] =
fuel_TT_calculator_function_cruise2(currentmass,...
                  distanceToCover
,aircraftType,cruise_alt,machNumber,latGCDRoute_cruise,lonGCDRoute_cruise,InterpWind);


      routeDistanceToNAT = DistClimb1 + DistCruise1 + DistClimb2;

      initialMassNAT = TOmass - FuelClimb1 - FuelCruise1 - FuelClimb2;
      optimalcruise_alt = maximumOperationalAltitude(aircraftType,initialMassNAT);            %
new operational cruise_alt

      optimalcruise_alt = floor(optimalcruise_alt/1000) * 1000;         % rounds to nearest 1000
feet
```

Cruise In NAT OTS

```
% Function to estimate the takeoff mass of the aircraft
% for three types of aircraft

% Inputs:
%
% 1) origin airport (4 letter)
% 2) NAT Track
% 3) aircraft type (3 types only)
% 4) Takeoff mass (kg)
% 5) Initial altitude to reach NAT (estimated) in feet
% 6) Mach number
% 7) set of winds speeds (knots)
% 8) vector of altitudes for wind (feet)

% Outputs

% 1) optimal altitude to enter NAT
% 2) route distance to NAT
% 3) travel time to NAT (minutes) - accounts for climb time
% 4) initial entry mass at NAT (kg)
```

```
function [optimalAltitude,routeDistanceInNAT,travelTime, fuelUsed, finalMassNAT, Wind] = ...
            estimate_Journey_Inside_NAT(NAT_Track,aircraftType,...
            initMassNAT,altitude,machNumber,Direction,track/flight level combination_wind)

global badaForOceanic  NAT_Tracks InterpWind


badaData = badaForOceanic;

%% Match the track used

if strcmp(Direction,'Easterly') == 1
        if strcmp(NAT_Track,'NATT')==1
            routeDistanceInNAT = max(NAT_Tracks.NATT.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATT.lat; Lon = NAT_Tracks.NATT.lon;
        elseif strcmp(NAT_Track,'NATU')==1
            routeDistanceInNAT = max(NAT_Tracks.NATU.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATU.lat; Lon = NAT_Tracks.NATU.lon;
        elseif strcmp(NAT_Track,'NATV')==1
            routeDistanceInNAT = max(NAT_Tracks.NATV.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATV.lat; Lon = NAT_Tracks.NATV.lon;
        elseif strcmp(NAT_Track,'NATW')==1
            routeDistanceInNAT = max(NAT_Tracks.NATW.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATW.lat; Lon = NAT_Tracks.NATW.lon;
        elseif strcmp(NAT_Track,'NATX')==1
            routeDistanceInNAT = max(NAT_Tracks.NATX.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATX.lat; Lon = NAT_Tracks.NATX.lon;
        elseif strcmp(NAT_Track,'NATY')==1
            routeDistanceInNAT = max(NAT_Tracks.NATY.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATY.lat; Lon = NAT_Tracks.NATY.lon;
        elseif strcmp(NAT_Track,'NATZ')==1
            routeDistanceInNAT = max(NAT_Tracks.NATZ.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATZ.lat; Lon = NAT_Tracks.NATZ.lon;
        elseif strcmp(NAT_Track,'NATUV')==1
            routeDistanceInNAT = max(NAT_Tracks.NATUV.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATUV.lat; Lon = NAT_Tracks.NATUV.lon;
        elseif strcmp(NAT_Track,'NATTU')==1
            routeDistanceInNAT = max(NAT_Tracks.NATTU.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATTU.lat; Lon = NAT_Tracks.NATTU.lon;
        elseif strcmp(NAT_Track,'NATVW')==1
            routeDistanceInNAT = max(NAT_Tracks.NATVW.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATVW.lat; Lon = NAT_Tracks.NATVW.lon;
        elseif strcmp(NAT_Track,'NATWX')==1
            routeDistanceInNAT = max(NAT_Tracks.NATWX.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATWX.lat; Lon = NAT_Tracks.NATWX.lon;
        elseif strcmp(NAT_Track,'NATXY')==1
            routeDistanceInNAT = max(NAT_Tracks.NATXY.cDistance);                        %
distance traveled in track
            Lat = NAT_Tracks.NATXY.lat; Lon = NAT_Tracks.NATXY.lon;
        end



elseif strcmp(Direction,'Westerly') == 1

        if strcmp(NAT_Track,'NATA')==1
            routeDistanceInNAT = max(NAT_Tracks.NATA.cDistance);                        %
distance traveled in track
```

```
            Lat = NAT_Tracks.NATA.lat; Lon = NAT_Tracks.NATA.lon;
        elseif strcmp(NAT_Track,'NATB')==1
            routeDistanceInNAT = max(NAT_Tracks.NATB.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATB.lat; Lon = NAT_Tracks.NATB.lon;
        elseif strcmp(NAT_Track,'NATC')==1
            routeDistanceInNAT = max(NAT_Tracks.NATC.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATC.lat; Lon = NAT_Tracks.NATC.lon;
        elseif strcmp(NAT_Track,'NATD')==1
            routeDistanceInNAT = max(NAT_Tracks.NATD.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATD.lat; Lon = NAT_Tracks.NATD.lon;
        elseif strcmp(NAT_Track,'NATE')==1
            routeDistanceInNAT = max(NAT_Tracks.NATE.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATE.lat; Lon = NAT_Tracks.NATE.lon;
        elseif strcmp(NAT_Track,'NATF')==1
            routeDistanceInNAT = max(NAT_Tracks.NATF.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATF.lat; Lon = NAT_Tracks.NATF.lon;
        elseif strcmp(NAT_Track,'NATG')==1
            routeDistanceInNAT = max(NAT_Tracks.NATG.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATG.lat; Lon = NAT_Tracks.NATG.lon;
        elseif strcmp(NAT_Track,'NATH')==1
            routeDistanceInNAT = max(NAT_Tracks.NATH.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATH.lat; Lon = NAT_Tracks.NATH.lon;
        elseif strcmp(NAT_Track,'NATJ')==1
            routeDistanceInNAT = max(NAT_Tracks.NATJ.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATJ.lat; Lon = NAT_Tracks.NATJ.lon;
        elseif strcmp(NAT_Track,'NATDE')==1
            routeDistanceInNAT = max(NAT_Tracks.NATDE.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATDE.lat; Lon = NAT_Tracks.NATDE.lon;
        elseif strcmp(NAT_Track,'NATEF')==1
            routeDistanceInNAT = max(NAT_Tracks.NATEF.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATEF.lat; Lon = NAT_Tracks.NATEF.lon;
        elseif strcmp(NAT_Track,'NATFG')==1
            routeDistanceInNAT = max(NAT_Tracks.NATFG.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATFG.lat; Lon = NAT_Tracks.NATFG.lon;
        elseif strcmp(NAT_Track,'NATGH')==1
            routeDistanceInNAT = max(NAT_Tracks.NATGH.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATGH.lat; Lon = NAT_Tracks.NATGH.lon;
        elseif strcmp(NAT_Track,'NATAB')==1
            routeDistanceInNAT = max(NAT_Tracks.NATAB.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATAB.lat; Lon = NAT_Tracks.NATAB.lon;
        elseif strcmp(NAT_Track,'NATCD')==1
            routeDistanceInNAT = max(NAT_Tracks.NATCD.cDistance);                    %
distance traveled in track
            Lat = NAT_Tracks.NATCD.lat; Lon = NAT_Tracks.NATCD.lon;
        end
end

    if isempty(NAT_Track) == 1
        ME = MException('TrackNotAssigned','Track is not assigned: Check NAT_Tracks.mat weather
the tracks exists or not')
        throw(ME)
    end

    %% Select aircraft index from BADA data set

    [aircraftPosition] = findAircraft(aircraftType);

% Calculate fuel used in the track traversal
```

```
     [FuelUsedInNAT,TravelTimeInNAT,Wind,DistInNat] =
fuel_TT_calculator_function_cruise2(initMassNAT,...

routeDistanceInNAT,aircraftType,altitude,machNumber,Lat,Lon,InterpWind,track/flight level
combination_wind);

% Calculate mass at exit point in NAT

      finalMassNAT = initMassNAT - FuelUsedInNAT ;          % estimate initialmass at NAT

% calculate the optimal altitude at exit NAT point based on calculations
% done above

      optimalAltitude = maximumOperationalAltitude(aircraftType,finalMassNAT);          % new
operational altitude
      % optimalAltitude = maximumAltitude(aircraftType,finalMassNAT,0) - buffetBoundaryAltitude;
% maximum cruise altitude based on TO mass

      optimalAltitude = floor(optimalAltitude/1000) * 1000;         % rounds to nearest 1000 feet

      travelTime = TravelTimeInNAT;
      fuelUsed = FuelUsedInNAT;
```

Step Climb After NAT

```
function [Climb3Alt,routeDistanceInClimb3,travelTimeClimb3, fuelUsedClimb3, finalMassClimb3] =
estimate_stepClimb_after_NAT(Dist_Left,...
    aircraftType,MassAfterNat,flightLevel,typicalMach,WindInNat)
MaxoperationalAlt = floor(maximumOperationalAltitude(aircraftType,MassAfterNat)/1000)*1000;
diff =  MaxoperationalAlt - flightLevel;
if Dist_Left >= 600 && diff >= 3000
    Climb3Alt = flightLevel + 3000;

    [fuelUsedClimb3,travelTimeClimb3,averageFuelBurn_kg_min,routeDistanceInClimb3] =
fuel_TT_calculator_function_climb(MassAfterNat,...
              aircraftType,flightLevel,Climb3Alt,typicalMach,WindInNat);
          finalMassClimb3 = MassAfterNat - fuelUsedClimb3;
elseif Dist_Left >= 350 && diff >= 2000
    Climb3Alt = flightLevel + 2000;
    [fuelUsedClimb3,travelTimeClimb3,averageFuelBurn_kg_min,routeDistanceInClimb3] =
fuel_TT_calculator_function_climb(MassAfterNat,...
              aircraftType,flightLevel,Climb3Alt,typicalMach,WindInNat);
          finalMassClimb3 = MassAfterNat - fuelUsedClimb3;
else

            routeDistanceInClimb3 = 0;
            travelTimeClimb3 = 0;
            fuelUsedClimb3 = 0;
            finalMassClimb3 = MassAfterNat;
            Climb3Alt = flightLevel;

end
```

Cruise From NAT Exit To TOD And Descent

```
% Function to estimate the Journey from NAT to the destination airport
% for three types of aircraft

% Inputs:
%
% 1) origin airport (4 letter)
% 2) NAT Track
% 3) aircraft type (3 types only)
% 4) Takeoff mass (kg)
% 5) Initial altitude to reach NAT (estimated) in feet
% 6) Mach number
% 7) set of winds speeds (knots)
```

```
% 8) vector of altitudes for wind (feet)

% Outputs

% 1) optimal altitude to enter NAT
% 2) route distance to NAT
% 3) travel time to NAT (minutes) - accounts for climb time
% 4) initial entry mass at NAT (kg)

function
[DistCruise3,TravelTimeFromNAT,FuelUsedFromNAT,massTOD,fuelUsedDescent,distanceToDescend,timeToDe
scend,Wind] = ...
            estimate_Journey_From_exitPoint(destination,NAT_Track,aircraftType,...
            exitMass,altitude,machNumber,Direction,DistUsedInClimb3)

%  load badaForOceanic
%  load airports_FA_Cell
%  load NAT_Tracks_Capacity

global badaForOceanic  airportCoordinates airportNames NAT_Tracks InterpWind

badaData = badaForOceanic;

detourFactor= 1.05;                         % adds 0% to the GCD distance to plan for
contingencies

 % Find airport information

flagForDestinationAirport = strcmp(airportNames,destination);                          % finds
position of origin airport
indexDestination = find(flagForDestinationAirport);
latlonDestination = airportCoordinates(indexDestination,:);                              %
coordinates of origin airport

%% Match the track used
if strcmp(Direction,'Easterly') == 1

        if strcmp(NAT_Track,'NATT') == 1
            latlonOrigin = [NAT_Tracks.NATT.lat(end) NAT_Tracks.NATT.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATU') == 1
            latlonOrigin = [NAT_Tracks.NATU.lat(end) NAT_Tracks.NATU.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATV') == 1
            latlonOrigin = [NAT_Tracks.NATV.lat(end) NAT_Tracks.NATV.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATW') == 1
            latlonOrigin = [NAT_Tracks.NATW.lat(end) NAT_Tracks.NATW.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATX') == 1
            latlonOrigin = [NAT_Tracks.NATX.lat(end) NAT_Tracks.NATX.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATY') == 1
            latlonOrigin = [NAT_Tracks.NATY.lat(end) NAT_Tracks.NATY.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATZ') == 1
            latlonOrigin = [NAT_Tracks.NATZ.lat(end) NAT_Tracks.NATZ.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATUV') == 1
            latlonOrigin = [NAT_Tracks.NATUV.lat(end) NAT_Tracks.NATUV.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATTU') == 1
            latlonOrigin = [NAT_Tracks.NATTU.lat(end) NAT_Tracks.NATTU.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATVW') == 1
            latlonOrigin = [NAT_Tracks.NATVW.lat(end) NAT_Tracks.NATVW.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATWX') == 1
            latlonOrigin = [NAT_Tracks.NATWX.lat(end) NAT_Tracks.NATWX.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATXY') == 1
```

```
            latlonOrigin = [NAT_Tracks.NATXY.lat(end) NAT_Tracks.NATXY.lon(end)];       %
coordinates of entry point for track
        end

elseif strcmp(Direction,'Westerly') == 1

        if strcmp(NAT_Track,'NATA') == 1
            latlonOrigin = [NAT_Tracks.NATA.lat(end) NAT_Tracks.NATA.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATB') == 1
            latlonOrigin = [NAT_Tracks.NATB.lat(end) NAT_Tracks.NATB.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATC') == 1
            latlonOrigin = [NAT_Tracks.NATC.lat(end) NAT_Tracks.NATC.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATD') == 1
            latlonOrigin = [NAT_Tracks.NATD.lat(end) NAT_Tracks.NATD.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATE') == 1
            latlonOrigin = [NAT_Tracks.NATE.lat(end) NAT_Tracks.NATE.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATF') == 1
            latlonOrigin = [NAT_Tracks.NATF.lat(end) NAT_Tracks.NATF.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATG') == 1
            latlonOrigin = [NAT_Tracks.NATG.lat(end) NAT_Tracks.NATG.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATH') == 1
            latlonOrigin = [NAT_Tracks.NATH.lat(end) NAT_Tracks.NATH.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATJ') == 1
            latlonOrigin = [NAT_Tracks.NATJ.lat(end) NAT_Tracks.NATJ.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATDE') == 1
            latlonOrigin = [NAT_Tracks.NATDE.lat(end) NAT_Tracks.NATDE.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATAB') == 1
            latlonOrigin = [NAT_Tracks.NATAB.lat(end) NAT_Tracks.NATAB.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATEF') == 1
            latlonOrigin = [NAT_Tracks.NATEF.lat(end) NAT_Tracks.NATEF.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATFG') == 1
            latlonOrigin = [NAT_Tracks.NATFG.lat(end) NAT_Tracks.NATFG.lon(end)];       %
coordinates of entry point for track
        elseif strcmp(NAT_Track,'NATGH') == 1
            latlonOrigin = [NAT_Tracks.NATGH.lat(end) NAT_Tracks.NATGH.lon(end)];       %
coordinates of entry point for track
         elseif strcmp(NAT_Track,'NATCD') == 1
            latlonOrigin = [NAT_Tracks.NATCD.lat(end) NAT_Tracks.NATCD.lon(end)];       %
coordinates of entry point for track

        end

end

    if isempty(NAT_Track) == 1
        ME = MException('TrackNotAssigned','Track is not assigned: Check NAT_Tracks.mat weather
the tracks exists or not')
        throw(ME)
    end


% Generate a GCD traversal from Origin to Destination
noWaypoints = 30;
[distanceGCD, latGCDRoute, lonGCDRoute] =
generateGCD(latlonOrigin(1),latlonOrigin(2),latlonDestination(1), latlonDestination(2),
noWaypoints);
[course_temp GCDdist] = legs(latGCDRoute,lonGCDRoute);
cumGCDdist_Inv = sum(GCDdist) - cumsum(GCDdist);
```

```
routeDistanceFromNAT = distanceGCD * detourFactor;              % in nm

    % Select aircraft index from BADA data set

    [aircraftPosition] = findAircraft(aircraftType);

 % Do the calculations from NAT to TOD point (climb
            % calculations)

   maxTimeDescent = max(badaData(aircraftPosition).nominalDescent(:,1));    % finds final time in
descent profile (seconds)
   maxFuelDescent = max(badaData(aircraftPosition).nominalDescent(:,4));    % finds final fuel in
descent profile (kg)

    altitude_m = altitude/3.28;             % altitude in meters
    if altitude_m > badaData(aircraftPosition).nominalDescent(1,3)
        altitude_m = badaData(aircraftPosition).nominalDescent(1,3);
    end

    fuelUsedDescent = maxFuelDescent - interp1(badaData(aircraftPosition).nominalDescent(:,3),...
                badaData(aircraftPosition).nominalDescent(:,4),altitude_m);
    timeToDescend = maxTimeDescent - interp1(badaData(aircraftPosition).nominalDescent(:,3),...
                badaData(aircraftPosition).nominalDescent(:,1),altitude_m);
    distanceToDescend =
interp1(badaData(aircraftPosition).nominalDescent(:,3),badaData(aircraftPosition).nominalDescent(
:,2),altitude_m);

    %calculate the point at which cruise(otherthan NAT or from NAT exit to TOD)
    %starts by comparing the distanceToDescend with cumulative GCD distance
    [tempVal tempIndex] = min(abs(cumGCDdist_Inv - distanceToDescend));
    TOD_Ind = tempIndex+1;
     latGCDRoute_cruise = latGCDRoute(1:TOD_Ind);
     lonGCDRoute_cruise = lonGCDRoute(1:TOD_Ind);

% Calculate fuel used from NAT exit point to the TOD point

     DistCruise3 = routeDistanceFromNAT - DistUsedInClimb3 -  distanceToDescend;    % distance in
nm


     [FuelUsedFromNAT,TravelTimeFromNAT, Wind] = fuel_TT_calculator_function_cruise2(exitMass,...

DistCruise3,aircraftType,altitude,machNumber,latGCDRoute_cruise,lonGCDRoute_cruise,InterpWind);



% Calculate mass at TOD point

      massTOD = exitMass - FuelUsedFromNAT;          % estimate final mass at TOD point
```

Generate Climb Profile

```
% This function generates the climb profile of the aircraft/airport
% combination being called
%
% Climb_Profile = Generate_Climb_Profile(Aircraft_Index)
%
% INPUT:
%   Aircraft_Index = index of AC/AP combo from T100 tracks structure
%
% OUTPUT:
%   Climb_Profile  =  motion profile of aircraft's climb

function Climb_Profile =
Generate_Climb_Profile(Aircraft_Index,Initial_Weight_kg)
```

```
% --------------
% DEFINE GLOBAL VARIABLES
% --------------
global BADA_Aircraft_Data rocTable CLIMB_TAS Altitude_Table CLIMB_FUEL_NOM
global FeetInNauticalMiles Time Assn_Cruise_FL Distance
global Current_Lat Current_Lon i Origin_Airport_Altitude Wind_Proj
global Waypoints Initial_Lat Initial_Lon BADA_Aircraft_Coef


global j true_air_speed rate_of_climb fuel_flow

ftpsecTOknots = 0.592483801;
% ------------------------------------------------------------------------

%load BADA_Aircraft_Data

% --------------
% GET AIRCRAFT DATA
% --------------
CLIMB_TAS      = BADA_Aircraft_Data(Aircraft_Index).CLIMB_TAS;
CLIMB_FUEL_NOM = BADA_Aircraft_Data(Aircraft_Index).CLIMB_FUEL_NOM;
Altitude_Table = BADA_Aircraft_Data(Aircraft_Index).FL * 100; % IN FEET
rocTable       = BADA_Aircraft_Data(Aircraft_Index).CLIMB_ROC_NOM; % RATE OF
CLIMB
% ------------------------------------------------------------------------

% --------------
% CORRECT AIRCRAFT DATA FOR NON-ZERO AIRPORT ALTITUDE
% --------------
if Origin_Airport_Altitude > 0
    % USE HYDROSTATICS TO GET DENSITY RATIO
    h = Origin_Airport_Altitude*0.3048; % IN METERS ABOVE SEALEVEL
    L = -.0065; % LAPSE RATE IN TROPOSHERE< IN (KELVIN/METER)
    T_SL = 288.15; % SEA LEVEL TEMPERATURE IN KELVIN, ACCORDING TO BADA
    g0 = 9.81; % BASE GRAVITATIONAL CONSTANT, IN m/s^2
    R = 287.0368; % GAS CONSTANT, IN N.m/kg.K

    T_h = T_SL + L*h; % TEMPERATURE AT AIRPORT ALTITUDE, IN KELVIN
    sigma = (T_h/T_SL)^(-(g0/(L*R)+1)); % DENSITY RATIO, rho_h/rho_SL

    % CORRECT ALTITUDE TABLE TO START AT ABOUT AIRPORT ALTITUDE
    [~,Data_Fix_Index,~] =
find(Altitude_Table>floor(Origin_Airport_Altitude),1);
    temp_TAS = CLIMB_TAS(Data_Fix_Index:end);
    temp_FUEL = CLIMB_FUEL_NOM(Data_Fix_Index:end);
    temp_ROC = rocTable(Data_Fix_Index:end);
    temp_ALT = Altitude_Table(Data_Fix_Index:end);
    CLIMB_TAS_new = temp_TAS;
    CLIMB_FUEL_NOM_new = temp_FUEL;
    rocTable_new = temp_ROC;
    Altitude_Table_new = temp_ALT;

    % REPLACE DATA WITH DENSITY RATIO-CORRECTED VALUES
    Length = length(CLIMB_TAS_new);
```

```
    CLIMB_TAS_new = CLIMB_TAS(1:Length)/sigma;
    CLIMB_FUEL_NOM_new = CLIMB_FUEL_NOM(1:Length)*sigma;
    rocTable_new = rocTable(1:Length)*sigma;

    % FOR ALTITUDES ABOVE 20000 FT, USE ORIGINAL BADA DATA
    [~,Index_Old_Data,~] = find(Altitude_Table == 20000);
    [~,Index_New_Data,~] = find(Altitude_Table_new == 20000);

    CLIMB_TAS_new(Index_New_Data:end) = CLIMB_TAS(Index_Old_Data:end);
    CLIMB_FUEL_NOM_new(Index_New_Data:end) =
CLIMB_FUEL_NOM(Index_Old_Data:end);
    rocTable_new(Index_New_Data:end) = rocTable(Index_Old_Data:end);

    Altitude_Table = Altitude_Table_new;
    CLIMB_TAS = CLIMB_TAS_new;
    CLIMB_FUEL_NOM = CLIMB_FUEL_NOM_new;
    rocTable = rocTable_new;

end
% -------------------------------------------------------------------------

% --------------
% DEFINE BOUNDARY AND INITIAL CONDITIONS
% --------------
% INITIAL LATITUDE AND LONGITUDE
Initial_Lat = Waypoints(1,1);
Initial_Lon = Waypoints(1,2);

% DEFINE TIME SPAN (ONE HOUR IN LENGTH)
Time_Initial = 0.0;                         % INITIAL TIME (SECONDS)
Time_Final   = 3600.0;                      % FINAL TIME (SECONDS)
Time_Span    = [Time_Initial Time_Final];   % SPAN TIME

% DEFINE INITIAL MASS
Aircraft_Mass_Initial   = Initial_Weight_kg; % KG

% DEFINE INITIAL STATE VARIABLES
yClimb_Initial  = [Origin_Airport_Altitude Aircraft_Mass_Initial 0];
% y(1) - ALTITUDE (FEET)
% y(2) - WEIGHT (KG)
% y(3) - DISTANCE TRAVELLED (FEET)

% DEFINE INITIAL F_Climb VARIABLES
i = 0;
j = 0;
Current_Lat = 0;
Current_Lon = 0;
Distance = 0;
Time = 0;
Wind_Proj = 0;
% -------------------------------------------------------------------------

% --------------
% GENERATE CLIMB PROFILE
% --------------
```

```
[Climb_Time,Climb_Data] = rk4sys('F_Climb_Coef', Time_Span, yClimb_Initial,
30, []);

% FIND INDEX FOR REACHING CRUISE FLIGHT LEVEL
Altitude_ft = Climb_Data(:,1);
Index_To_Reach_MaxFL = find(Altitude_ft >= Assn_Cruise_FL, 1, 'first');
if isempty(Index_To_Reach_MaxFL) == 1
    Index_To_Reach_MaxFL = length(Altitude_ft)-1; % GET INDEX OF LAST ITEM IF
NOTHING TO TRUNCATE
end

true_air_speed = true_air_speed(1:Index_To_Reach_MaxFL);
rate_of_climb = rate_of_climb(1:Index_To_Reach_MaxFL);
fuel_flow = fuel_flow(1:Index_To_Reach_MaxFL);

% TRUNCATE CLIMB PROFILE TO STOP AT CRUISE FLIGHT LEVEL; CONVERT DATA
Altitude_ft             = Altitude_ft(1:Index_To_Reach_MaxFL);
Distance_nm             =
Climb_Data(1:Index_To_Reach_MaxFL,3)/FeetInNauticalMiles;
Time_hrs                = Climb_Time(1:Index_To_Reach_MaxFL) / 3600;
Weight_kg               = Climb_Data(1:Index_To_Reach_MaxFL,2);

Rate_of_Climb_ftmin     = rate_of_climb';
Speed_knots             = true_air_speed';%(Distance_nm(2:end) -
Distance_nm(1:end-1)) ./ (Time_hrs(2:end) - Time_hrs(1:end-1));
FuelFlow_kgmin          = fuel_flow';%-(Weight_kg(2:end) - Weight_kg(1:end-
1)) ./ (Time_hrs(2:end) - Time_hrs(1:end-1)) / 60;
Distance_nm_For_Speed   = (Distance_nm(2:end) + Distance_nm(1:end-1)) / 2;
Total_Fuel_kg           = Weight_kg(1) - Weight_kg(end);

Latitude_Pts            = Current_Lat(1:Index_To_Reach_MaxFL);
Longitude_Pts           = Current_Lon(1:Index_To_Reach_MaxFL);
Wind_Vectors            = Wind_Proj(1:Index_To_Reach_MaxFL);
% -------------------------------------------------------------------------

% --------------
% SAVE CLIMB PROFILE IN STRUCTURE
% --------------
Climb_Profile.Distance_For_Climb_nm = max(Distance_nm);
Climb_Profile.Time_For_Climb_hrs    = max(Time_hrs);
Climb_Profile.Total_Fuel_kg         = Total_Fuel_kg;

Climb_Profile.Altitude_ft           = Altitude_ft;
Climb_Profile.Distance_nm           = Distance_nm;
Climb_Profile.Time_hrs              = Time_hrs;
Climb_Profile.Weight_kg             = Weight_kg;

Climb_Profile.Rate_of_Climb_ftmin   = Rate_of_Climb_ftmin;
Climb_Profile.Speed_knots           = Speed_knots;
Climb_Profile.FuelFlow_kgmin        = FuelFlow_kgmin;
Climb_Profile.Distance_nm_For_Speed = Distance_nm_For_Speed;

Climb_Profile.Latitude_Pts          = Latitude_Pts;
Climb_Profile.Longitude_Pts         = Longitude_Pts;
Climb_Profile.Wind_Vectors_knots    = Wind_Vectors * ftpsecTOknots;
```

```
% ------------------------------------------------------------------------

return
```

Cruise Fuel Calculator

```
% Program to estimate fuel burn and travel time for an aircraft
% flying a long haul route over the Atlantic or North Pacific

% Date: November 3, 2011

function
[totalFuelUsed,totalTravelTime,averageWindSpeed,distanceTraveledCruise] =
fuel_TT_calculator_function_cruise2(initialMass,...

distanceToCover,aircraftType,cruiseAltitude,machNumber,Lat,Lon,InterpWind,tra
ck/flight level combination_wind)

% Inputs:
% badaForOceanic = BADA struct file with aerodynamic coefficients for each
aircraft
% cruiseAltitude = desired cruise altitude (feet)
% initialMass = initial aircraft mass (kg)
% routeFlown = route to be studied (from FAA routes for now)

% Outputs

% fuelUsed = total fuel used in trip (kg)
% travelTime = total travel time in trip (minutes)
if distanceToCover < 0
    totalFuelUsed = 0;
    totalTravelTime = 0;
    averageWindSpeed = 0;
else

% _________ Load the files with aerodynamic data _____________
global badaForOceanic

%load badaForOceanic

% Constants

k_ft_to_m = 1/3.28;
mps_to_knots = 3600/1852;   % meters per second to knots
deltaTime = 10;                     % initial cruise delta time (minutes)

% Detect routes between the OD pairs

routeDistance = distanceToCover;    % extracts the information for a given
route

% Select aircraft index from BADA data set
```

```
    [aircraftPosition] = findAircraft(aircraftType);

% define badaData

badaData = badaForOceanic;          % change in variable name
%clear badaForOceanic                    % clear the old variable

% Do the climb calculation for travel time and fuel used

% Nominal climb profile contains 4 columns:
% 1 = time (seconds)
% 2 = distance traveled (nm)
% 3 = altitude (m)
% 4 = fuel used (kg)

cruiseDistance = distanceToCover;        % distance in cruise in nm
% averageWindSpeed = interp1(windAltitudeVector,averageWind,cruiseAltitude);


if InterpWind == 0
 averageWindSpeed = track/flight level combination_wind;
elseif InterpWind == 1
    averageWindSpeed = getwindvect_1(cruiseAltitude,Lat,Lon);
elseif InterpWind ==2
    averageWindSpeed = getwindvect(cruiseAltitude,Lat,Lon);
else
        ME = MException('TrackNotAssigned','Track is not assigned: Check
NAT_Tracks.mat weather the tracks exists or not')
        throw(ME)
end
% Calculate the true airspeed given mach number

[speedOfSound,density,temperature] = isan(cruiseAltitude * k_ft_to_m);
% altitude in meters
cruiseSpeed = machNumber * speedOfSound * mps_to_knots;         % in knots

% Start Cruise integration algorithm calling the BADA function to estimate
% fuel burn

actualFuel(1) = 0;                                      % initial value of
Fuel state in cruise mode (kg)
mass(1) = initialMass;                                  % starting mass for
cruise mode (kg)
actualDistance(1) = 0;                                 % initial value of
distance traveled in cruise mode (m)
cruiseTimeCount(1) =0;
i = 1;                                                          % index used in
while loop for numerical integration

% Solve ODEs here using simple Euler first-order method (crude but
thrustworthy)

FlightCounter = 0;
```

```
while actualDistance(i) <= cruiseDistance                  % iterate until the
aircraft reaches final cruise condition

    % calculate fuel and aerodynamic values

    [drag,fuelFlow(i),cl,cd] =
cruiseCalculations_BADA39(cruiseSpeed,cruiseAltitude,mass(i),aircraftType,bad
aData);

    % update accumulators

    actualFuel(i+1) = actualFuel(i) + fuelFlow(i) * deltaTime;
% fuel state (kg)
    mass(i+1) = mass(i) - fuelFlow(i) * deltaTime ;
% Distance traveled along the flight path (m)
    cruiseTimeCount(i+1) = cruiseTimeCount(i) + deltaTime;
% counts travel time (minutes)

    actualDistance(i+1) = actualDistance(i) + (cruiseSpeed/60 + ...
        averageWindSpeed/60) * deltaTime ;      % distance traveled  (nm)

% speed is in nm/min

    cruiseTimeCount(i+1) = cruiseTimeCount(i) + deltaTime;        % update
counter for time (to measure when simulation stops)
    i=i+1;                                                 % update index to
keep vector for state variables

            % FLight Counter
%         if mod(FlightCounter, 50) == 0
%             disp(['Iterations: ', num2str(FlightCounter)]);
%         end
        FlightCounter = FlightCounter + 1; % Counts the number of instances
in loop

end

% detect final state

fState = i-1;             % index of final state

% write temporary states

finalTimeTemp = cruiseTimeCount(fState);                          % time when
simulation stops
fuelUsedInCruiseTemp = actualFuel(fState);
distanceTraveledCruiseTemp = actualDistance(fState);
massEndOfCruiseTemp = mass(fState);
lastFuelFlow = fuelFlow(fState);                                          %
final fuel flow detected (kg/min)

% Correct for the distance overflown in the last segment in the cruise
```

```
deltaDistance = distanceTraveledCruiseTemp - cruiseDistance;        % delta
distance overflown (nm)
groundSpeed = ((cruiseSpeed + averageWindSpeed)/60);                %
nm/minute
timeOverflown = deltaDistance / groundSpeed;                                %
in minutes
deltaFuel = lastFuelFlow * timeOverflown;

totalTravelTime  = finalTimeTemp - timeOverflown;                   % minutes
totalFuelUsed = fuelUsedInCruiseTemp - lastFuelFlow * timeOverflown;
finalMass = massEndOfCruiseTemp - lastFuelFlow * timeOverflown;
% corrected mass
distanceTraveledCruise = distanceTraveledCruiseTemp - deltaDistance;
end
```

Assignment Module

```
% Script to estimate the cost of flying each flight across teh NAT for
% various altitudes and track combinations

% Uses a fixed track structure defined for one day of operations

clear all
clc
chngTOrnd = 0;
global Ex_0_East Ex_0_West RLat_1_East RLat_1_West RLat_2_East RLat_2_West
RLat_1_EastCore RLat_1_WestCore East_del_trk West_del_trk Ex_1_coreEast
Ex_1_coreWest
global badaForOceanic  Dist_FL_Relation  airportNames airportCoordinates Wind
atmosphere airports_FA_Cell
global Detour East_Rnd_Trks West_Rnd_Trks default_Intrail dyamic_headway
load badaForOceanic
load atmosphere
% load ('D:\Wind Data\NATSIM Wind Data\March15\Wind.mat')
load Dist_FL_Relation
load airports_FA_Cell

InputDir =
'D:\NATSAM_Results\Final_Runs\NATSAM_V1\July26\DLM_RLatSM_RLongSM_2015\';


%global Wind badaForOceanic  airportCoordinates airportNames NAT_Tracks
casename = {'2a'; '3a'; '4a'; '5a'; '6a'; '7a'; '8a'; '9a'; '10a'; '2b';
'3b'; '4b'; '5b'; '6b'; '7b'; 'bl-2013'; 'bl-2015'; 'bl-2017'; 'bl-2020';
'5ab'; '6ab'; '7ab'; '5c'; '6c'; '7c'};
rlatsm = [0 0 0 0 0 0 0 0 0 1 1 1 2 2 2 0 0 0 0 1 1 1 1 1 1];
exclairspace = [1 1 1 2 2 2 2 2 2 1 1 1 2 2 2 0 0 0 0 2 2 2 2 2 2];
rlongsm = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1];
EntryTimes = [];
default_Intrail = 10;
dyamic_headway = 1;

Detour = 1.04;
```

```
%      trackArrayEast = {'NATT'; 'NATU'; 'NATV'; 'NATW'; 'NATX'; 'NATY';
'NATZ'; 'NATUV'; 'NATVW'; 'NATWX'; 'NATXY'};
%                            1       2       3       4       5       6       7
8       9        10        11%
%     trackArrayWest = {'NATA'; 'NATB'; 'NATC'; 'NATD'; 'NATE'; 'NATF';
'NATG'; 'NATH'; 'NATJ';'NATAB'; 'NATCD'; 'NATDE'; 'NATEF'; 'NATFG'; 'NATGH'};
%                           1       2       3       4       5       6       7
8       9      10       11       13      14        15        16      %

%%% Provide track details
%%%%%%%%%%%%%%%%%%%%%%%%  Track system for RLatSM Phase 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     Ex_0_East = [2 3 4 5 6 7];   RLat_1_East = [2 3 9 10 11 7];
RLat_2_East = [2 8 3 9 4 7]; RLat_1_EastCore = [3 9]; East_del_trk = [10 5 11
6];
     Ex_0_West = [1 2 3 4 5 6 7 8 9];   RLat_1_West = [1 2 3 4 5 13 14 15 9];
RLat_2_West = [1 10 2 4 12 5 13 6 9];  RLat_1_WestCore = [5 13]; West_del_trk
= [3 7 8 14 15];
     Ex_1_coreEast = [3 4]; Ex_1_coreWest = [5 6]; East_Rnd_Trks = [5 6];
West_Rnd_Trks = [3 7 8];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for z = 23:25
    casename(z);
%     Iter = 0;
% if Iter == 0
    load([InputDir,char(casename(z)),'\flightsNATS_RNDcost.mat'])
    load('D:\NAT Track Files\OTS_Polygon_20080726.mat')
% elseif Iter == 1
%     load([InputDir,char(casename(z)),'\flightsNATS_Final1.mat'])
% elseif Iter == 2
%     load([InputDir,char(casename(z)),'\flightsNATS_Final2.mat'])
% end
%load badaForOceanic
%load airports_FA_Cell
%load NAT_Tracks_Capacity
%load Track_structure_Capacity_Analysis
%load Wind.mat
% % % Define Case variables and switches to be able to switch between
% % % different scenarios
% % % First switch is RLatSM (Reduced lateral separation minima)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  RLatSM = 0 means no additional tracks
%%%
%%%  RLatSM = 1 means Phase 1 ==> one additional core track between the two
OTC tracks              %%%
%%%  RLatSM = 2 means Phase 2 ==> Additional core tracks @ 1\2 degree spacing
across the entire OTS%%%
%%%  All RLatSM (additional core tracks) exist only at FL 360 to FL 390
inclusive                   %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% % % Second switch is Exclusive airspace which switches from "No mandate
scenario" to "Mandate in effect scenario" % % %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ExcAirSpace = 0 means all aircraft can fly this airspace
%%%
%%% ExcAirSpace = 1 means only Datalink equipped aircraft can fly 3 central
core tracks-FL360 to FL390  %%%
%%% ExcAirSpace = 2 means only Datalink equipped aircraft can fly all tracks-
FL360 to FL390             %%%
%%% ExcAirSpace extends from FL 360 to FL 390 inclusive
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Define flight levels based on the switches%%%

    casename(z);
    RLatSM_Actual = rlatsm(z);
    ExcAirSpace_Actual = exclairspace(z);
    RLongSM = rlongsm(z);
    disp(['***  Case ',char(casename(z)), '  **** RLatSM : ',
num2str(RLatSM_Actual), '  **** ExAirSpace : ', num2str(ExcAirSpace_Actual),'
***'])
    disp(['***  The Default in-trail separation is
',num2str(default_Intrail), ' mins  and value for dynamic head way is
',num2str(dyamic_headway),'  *******************'])


rndcount = 0;

disp('Did you specify the rules in Validate_Track_FL script???')
disp('If yes, simply proceed by pressing any key. If No, then stop simulation
and specify them.')
% %pause

noFlights = length(flight);                                   % find number
of flights
counter = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Uncomment/comment depending on how many
tracks you have %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     trackArrayEast = {'NATU'; 'NATV'; 'NATW'; 'NATX'; 'NATY'; 'NATZ';
'NATUV'; 'NATVW'; 'NATWX'; 'NATXY'};                     %
%     trackArrayWest = {'NATA'; 'NATB'; 'NATC'; 'NATD'; 'NATE'; 'NATF';
'NATG'; 'NATH'; 'NATJ'; 'NATAB'; 'NATDE'; 'NATEF'; 'NATFG'; 'NATGH'};     %
%     DatumLat_East = [1 2 3 4 5 6 7 1.5 2.5 3.5 4.5 5.5];
%
%     DatumLat_West = [1 2 3 4 5 6 7 8 9 1.5 4.5 5.5 6.5 7.5];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %&&&&&&&&&&&&&&&&&&&&&&  Uncomment/comment depending on how many tracks you
have  &&&&&&&&&&&&&&&&&&&&&&&&&&&%%
    trackArrayEast = {'NATT'; 'NATU'; 'NATV'; 'NATW'; 'NATX'; 'NATY'; 'NATZ';
'NATUV'; 'NATVW'; 'NATWX'; 'NATXY'};
```

```
    trackArrayWest = {'NATA'; 'NATB'; 'NATC'; 'NATD'; 'NATE'; 'NATF'; 'NATG';
'NATH'; 'NATJ';'NATAB'; 'NATCD'; 'NATDE'; 'NATEF'; 'NATFG'; 'NATGH'};
    DatumLat_East = [1 2 3 4 5 6 9 2.5 3.5 4.5 5.5];
%
    DatumLat_West = [1 2 3 4 5 6 7 8 9 1.5 3.5 4.5 5.5 6.5 7.5];
%

%
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&%

NAT_State_East = zeros(numel(trackArrayEast),9,1);
NAT_State_West = zeros(numel(trackArrayWest),9,1);
NAT_Speed_East = zeros(numel(trackArrayEast),9,1);
NAT_Speed_West = zeros(numel(trackArrayWest),9,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% OTS validity times %%%%%%
Eastbound_Start = [0 1350];  Eastbound_End = [420 1770];
Westbound_Start = 570; Westbound_End = 1050;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Enter computational loop
tic;
for i=1:noFlights
% disp(num2str(i))
    SimulationTime_UTC = flight(i).NatEntryTime;
    First_RndFlightCalc = 1;
    ChangedToRandom = 0;
    flight(i).ChangeToRND = 'No';
    Equipage = flight(i).Equipage;
    actInd = findAircraft(flight(i).originalAircraft);
    flight(i).BadaAct = badaForOceanic(actInd).aircraftName;
    Speed_Mach = badaForOceanic(actInd).typicalMach;
    Direction = flight(i).direction;

    TrackAssigned = 0;
    NoObjection = 0;
    Rnd_better = 0;
%     if Iter == 0
        EntryTime = flight(i).NatEntryTime;
        RND_Fuel = flight(i).RNDFuelBurned;
        Min_OTS_fuel = flight(i).fuelCostVector(1,1);

        if strcmp(flight(i).Time_type,'OTS')
                ExcAirSpace = ExcAirSpace_Actual;RLatSM = RLatSM_Actual;
                In_OTS_Period = 1;
        else
                ExcAirSpace = 0;RLatSM = 0;
                In_OTS_Period = 0;
        end

    if strcmp(Direction,'Easterly') == 1 %&& strcmp(flight(i).RteType,'RND')
== 0

            counter = 1;
```

```
            FirstNoObjection = 0;

        if ~In_OTS_Period
            if (RND_Fuel +150- Min_OTS_fuel) < 0
                fileRND = 1;TakeOTS = 0;
                    flight(i).AssignedTrack = 'RND1';
                    flight(i).AssignedFL = flight(i).RNDFL_avg;
                    flight(i).FuelBurned = flight(i).RNDFuelBurned;
                    flight(i).DistTravelled = flight(i).RNDDistTravelled;
                    flight(i).TimeTravelled = flight(i).RNDTimeTravelled;
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
                    Rnd_better = Rnd_better + 1;
            else
                fileRND=0;TakeOTS = 1;
                    flight(i).AssignedTrack = 'RND1';
                    flight(i).AssignedFL =
flight(i).fuelCostVector(counter,2);
                    flight(i).FuelBurned =
flight(i).fuelCostVector(counter,1);
                        flInd = flight(i).fuelCostVector(counter,2)/1000;
flInd = flInd-30;
                        TrkInd = flight(i).fuelCostVector(counter,3);
                    flight(i).DistTravelled =
flight(i).costDistance(TrkInd,flInd);
                    flight(i).TimeTravelled =
flight(i).costTime(TrkInd,flInd);
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
                    Rnd_better = Rnd_better + 1;

            end
        else
            Inside =
sum(inpolygon(flight(i).GC_Lat,flight(i).GC_Lon,OTS_Polygon_20080726.EastLowe
r.lat,OTS_Polygon_20080726.EastLower.lon));
            if (RND_Fuel +150- Min_OTS_fuel) < 0 && ~Inside
                fileRND = 1;TakeOTS = 0;
                    flight(i).AssignedTrack = 'RND2';
                    flight(i).AssignedFL = flight(i).RNDFL_avg;
                    flight(i).FuelBurned = flight(i).RNDFuelBurned;
                    flight(i).DistTravelled = flight(i).RNDDistTravelled;
                    flight(i).TimeTravelled = flight(i).RNDTimeTravelled;
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
            else
                fileRND=0;TakeOTS = 1;



                while TrackAssigned == 0
                        TrackIndex = flight(i).fuelCostVector(counter,3);
                        FL = flight(i).fuelCostVector(counter,2);
                        FL_Index = (FL-30000)/1000 ;
```

```
                        [NoObjection CheckRandomCost Switch_To_RndTrk,RLong]
=
Validate_Track_FL(Direction,FL,TrackIndex,Equipage,ExcAirSpace,RLatSM,RLongSM
);
                        if NoObjection && ~FirstNoObjection     %First valid
track
                            FirstNoObjection = 1; bestTrkInd = TrackIndex;
                           flight(i).BestTrack =
char(trackArrayEast(TrackIndex));
                           flight(i).BestFL = FL;
                        end

                        Inj_No = NAT_State_East(TrackIndex,FL_Index,1);
                        Speed_Diff = Speed_Mach -
NAT_Speed_East(TrackIndex,FL_Index,1);
                        [In_Trail] =
findInTrail(Speed_Diff,FL,ExcAirSpace,RLatSM,RLong);

                    if (EntryTime -
NAT_State_East(TrackIndex,FL_Index,Inj_No+1)) >= In_Trail && NoObjection
                        TrackAssigned = 1;
                        NAT_Speed_East(TrackIndex,FL_Index,1) = Speed_Mach;
                        flight(i).AssignedTrack =
char(trackArrayEast(TrackIndex));
                        flight(i).AssignedFL = FL;
                        flight(i).FuelBurned =
flight(i).fuelCostVector(counter,1);
                        flInd = flight(i).fuelCostVector(counter,2)/1000;
flInd = flInd-30;
                        TrkInd = flight(i).fuelCostVector(counter,3);
                        flight(i).DistTravelled =
flight(i).costDistance(TrkInd,flInd) + flight(i).fuelCostVector(counter,4);
                        flight(i).TimeTravelled =
flight(i).costTime(TrkInd,flInd);
                        flight(i).LatDeviation =
abs(DatumLat_East(bestTrkInd) - DatumLat_East(TrackIndex));
                        flight(i).Vert_dev = abs(flight(i).BestFL - FL);
                        NAT_State_East(TrackIndex,FL_Index,1) = Inj_No+1;
                        NAT_State_East(TrackIndex,FL_Index,Inj_No+2) =
EntryTime;
                        if Switch_To_RndTrk
                                    chngTOrnd = chngTOrnd+1;
                                    flight(i).ChangeToRND = 'Yes';
                                    flight(i).LatDeviation = 'O2R';
                                    flight(i).Vert_dev = 'O2R';
                        end

                    else
                        counter = counter+1;
                    end
                end
            end
        end
```

```
    elseif strcmp(Direction,'Westerly') == 1 %&&
strcmp(flight(i).RteType,'RND') == 0

            counter = 1;
            FirstNoObjection = 0;

        if ~In_OTS_Period
            if (RND_Fuel +150- Min_OTS_fuel) < 0
                fileRND = 1;TakeOTS = 0;
                    flight(i).AssignedTrack = 'RND1';
                    flight(i).AssignedFL = flight(i).RNDFL_avg;
                    flight(i).FuelBurned = flight(i).RNDFuelBurned;
                    flight(i).DistTravelled = flight(i).RNDDistTravelled;
                    flight(i).TimeTravelled = flight(i).RNDTimeTravelled;
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
                    Rnd_better = Rnd_better + 1;
            else
                fileRND=0;TakeOTS = 1;
                    flight(i).AssignedTrack = 'RND1';
                    flight(i).AssignedFL =
flight(i).fuelCostVector(counter,2);
                    flight(i).FuelBurned =
flight(i).fuelCostVector(counter,1);
                        flInd = flight(i).fuelCostVector(counter,2)/1000;
flInd = flInd-30;
                        TrkInd = flight(i).fuelCostVector(counter,3);
                    flight(i).DistTravelled =
flight(i).costDistance(TrkInd,flInd);
                    flight(i).TimeTravelled =
flight(i).costTime(TrkInd,flInd);
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
                    Rnd_better = Rnd_better + 1;

            end
        else
            Inside =
sum(inpolygon(flight(i).GC_Lat,flight(i).GC_Lon,OTS_Polygon_20080726.WestLowe
r.lat,OTS_Polygon_20080726.WestLower.lon));
            if (RND_Fuel +150- Min_OTS_fuel) < 0 && ~Inside
                fileRND = 1;TakeOTS = 0;
                    flight(i).AssignedTrack = 'RND2';
                    flight(i).AssignedFL = flight(i).RNDFL_avg;
                    flight(i).FuelBurned = flight(i).RNDFuelBurned;
                    flight(i).DistTravelled = flight(i).RNDDistTravelled;
                    flight(i).TimeTravelled = flight(i).RNDTimeTravelled;
                    flight(i).LatDeviation = 'NA';
                    flight(i).Vert_dev = 'NA';
            else
                fileRND=0;TakeOTS = 1;



                while TrackAssigned == 0
                        TrackIndex = flight(i).fuelCostVector(counter,3);
```

```
                        FL = flight(i).fuelCostVector(counter,2);
                        FL_Index = (FL-30000)/1000 ;

                        [NoObjection CheckRandomCost Switch_To_RndTrk,RLong]
=
Validate_Track_FL(Direction,FL,TrackIndex,Equipage,ExcAirSpace,RLatSM,RLongSM
);
                        if NoObjection && ~FirstNoObjection     %First valid
track
                            FirstNoObjection = 1; bestTrkInd = TrackIndex;
                           flight(i).BestTrack =
char(trackArrayWest(TrackIndex));
                           flight(i).BestFL = FL;
                        end

                        Inj_No = NAT_State_West(TrackIndex,FL_Index,1);
                        Speed_Diff = Speed_Mach -
NAT_Speed_West(TrackIndex,FL_Index,1);
                        [In_Trail] =
findInTrail(Speed_Diff,FL,ExcAirSpace,RLatSM,RLong);

                    if (EntryTime -
NAT_State_West(TrackIndex,FL_Index,Inj_No+1)) >= In_Trail && NoObjection
                        TrackAssigned = 1;
                        NAT_Speed_West(TrackIndex,FL_Index,1) = Speed_Mach;
                        flight(i).AssignedTrack =
char(trackArrayWest(TrackIndex));
                        flight(i).AssignedFL = FL;
                        flight(i).FuelBurned =
flight(i).fuelCostVector(counter,1);
                        flInd = flight(i).fuelCostVector(counter,2)/1000;
flInd = flInd-30;
                        TrkInd = flight(i).fuelCostVector(counter,3);
                        flight(i).DistTravelled =
flight(i).costDistance(TrkInd,flInd)+ flight(i).fuelCostVector(counter,4);
                        flight(i).TimeTravelled =
flight(i).costTime(TrkInd,flInd);
                        flight(i).LatDeviation =
abs(DatumLat_West(bestTrkInd) - DatumLat_West(TrackIndex));
                        flight(i).Vert_dev = abs(flight(i).BestFL - FL);
                        NAT_State_West(TrackIndex,FL_Index,1) = Inj_No+1;
                        NAT_State_West(TrackIndex,FL_Index,Inj_No+2) =
EntryTime;
                        if Switch_To_RndTrk
                                    chngTOrnd = chngTOrnd+1;
                                    flight(i).ChangeToRND = 'Yes';
                                    flight(i).LatDeviation = 'O2R';
                                    flight(i).Vert_dev = 'O2R';
                        end

                    else
                        counter = counter+1;
                    end
                end
            end
        end
```

```
    end
end

% pause
toc
% disp(['Total number of flights outside OTS time: ', num2str(rndcount)])

% if Iter == 0
    save([InputDir,char(casename(z)),'\flightsNATS_Output.mat'], 'flight')
% else
%
save([InputDir,char(casename(z)),'\flightsNATS_Final',num2str(Iter),'.mat'],
'flight')
% end
clear BestFL BestTrack CoreTracksEast CoreTracksWest Direction ExcAirSpace FL
FL_Index Inj_No
clear NoObjection RLatSM TrackAssigned TrackIndex counter flight i noFlights
noTracksEast noTracksWest

East_Summary=NAT_State_East(:,:,1);
West_Summary=NAT_State_West(:,:,1);
NAT_State_East(:,:,1) = [];     NAT_State_West(:,:,1) = [];
East_Summary = flipud(East_Summary');
West_Summary = flipud(West_Summary');
[r c] = size(East_Summary);
for i = 1:r
 East_Summary(i,c+1) = sum(East_Summary(i,:));
 end
for i = 1:c
 East_Summary(r+1,i) = sum(East_Summary(:,i));
 end

[r c] = size(West_Summary);
for i = 1:r
West_Summary(i,c+1) = sum(West_Summary(i,:));
end
for i = 1:c
West_Summary(r+1,i) = sum(West_Summary(:,i));
end

East_Summary(end,end) =  sum(East_Summary(1:end-1,end));
West_Summary(end,end) =  sum(West_Summary(1:end-1,end));

clear i r c ans

% if Iter == 0
    save ([InputDir,char(casename(z)),'\NATEntryStats.mat'])
    sheet = char(casename(z));
% else
%     save
([InputDir,char(casename(z)),'\NATEntryStats',num2str(Iter),'.mat'])
%     sheet = ['Case ',char(casename(z)),num2str(Iter)];
% end
```

```
alpha = {'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R'
'S' 'T'};
start_row = 3; gap_rows = 4;
    [NOrowE NOcolE] = size(East_Summary); east_loc =
[char(alpha(1)),num2str(start_row),':',char(alpha(NOcolE)),num2str(start_row+
NOrowE-1)];
    [NOrowW NOcolW] = size(West_Summary); west_loc =
[char(alpha(1)),num2str(start_row+NOrowE-
1+gap_rows),':',char(alpha(NOcolW)),num2str(start_row+NOrowE-
1+gap_rows+NOrowW-1)];

xlswrite([InputDir,'Simulation_Results.xlsm'],East_Summary,sheet,east_loc);

xlswrite([InputDir,'Simulation_Results.xlsm'],West_Summary,sheet,west_loc);
    East_heading = [char(alpha(1)),num2str(start_row-
1),':',char(alpha(NOcolE-1)),num2str(start_row-1)];
    West_heading = [char(alpha(1)),num2str(start_row+NOrowE+gap_rows-
2),':',char(alpha(NOcolW-1)),num2str(start_row+NOrowE+gap_rows-2)];

xlswrite([InputDir,'Simulation_Results.xlsm'],trackArrayEast',sheet,East_head
ing);

xlswrite([InputDir,'Simulation_Results.xlsm'],trackArrayWest',sheet,West_head
ing);
    FL = 390:-10:310;FL = FL';FL_E_head =
['A',num2str(start_row),':','A',num2str(start_row+NOrowE-2)];
    xlswrite([InputDir,'Simulation_Results.xlsm'],FL,sheet,FL_E_head);
    FL_W_head = ['A', num2str(start_row+NOrowE+gap_rows-
1),':','A',num2str(start_row+NOrowE+gap_rows+NOrowW-3)];
    xlswrite([InputDir,'Simulation_Results.xlsm'],FL,sheet,FL_W_head);


end
```

Track FL Request Validation Module

```
function [NoObjection CheckRandomCost Switch_To_RndTrk,RLong] =
Validate_Track_FL(Direction,FL,TrackIndex,Equipage,ExcAirSpace,RLatSM,RLongSM
)
      if strcmp(Equipage,'E')
          Equipped = 1;
      else Equipped = 0; end
     CheckRandomCost = 0;
     if strcmp(Direction,'Easterly')
         East = 1; West = 0;
     else East = 0; West = 1; end
     global Ex_0_East Ex_0_West RLat_1_East RLat_1_West RLat_2_East
RLat_2_West RLat_1_EastCore RLat_1_WestCore East_del_trk West_del_trk
     global Ex_1_coreEast Ex_1_coreWest East_Rnd_Trks West_Rnd_Trks

%      Ex_0_East = [1 2 3 4 5 6];   RLat_1_East = [7 8 9 4 5 6];
RLat_2_East = [3 9 4 10 5 6]; RLat_1_EastCore = [9 4]; East_del_trk = [1 2 7
8];
%      Ex_0_West = [1 2 3 4 5 6];   RLat_1_West = [1 2 3 4 9 10];
RLat_2_West = [7 2 8 4 9 5];  RLat_1_WestCore = [4 9]; West_del_trk = [1 3 10
6];
```

```
%      Ex_1_coreEast = [3 4]; Ex_1_coreWest = [4 5];
     Switch_To_RndTrk = 0;
    %%% Flight admission rules for Data link mandate
    %%% RLong = 0 by default for all flights and = 1 for only cleared flights
%%
    RLong = 0;
     if ExcAirSpace == 0 && RLatSM == 0
         if East
               if  ismember(TrackIndex,Ex_0_East)
                    NoObjection = 1;
               else NoObjection = 0;
               end
         else
               if  ismember(TrackIndex,Ex_0_West)
                    NoObjection = 1;
               else NoObjection = 0;
               end
         end

     elseif ExcAirSpace == 1 && RLatSM == 0
                          if Equipped
                              if East
                                        if  ismember(TrackIndex,Ex_0_East)
                                             NoObjection = 1;
                                        else NoObjection = 0; end
                                   else
                                        if ismember(TrackIndex,Ex_0_West)
                                             NoObjection = 1;
                                        else NoObjection = 0; end
                                   end

                          elseif FL >= 36000
                                  if East
                                        if ismember(TrackIndex,Ex_0_East) &&
~ismember(TrackIndex,Ex_1_coreEast)
                                             NoObjection = 1;
                                        else NoObjection = 0; end
                                   else
                                        if ismember(TrackIndex,Ex_0_West) &&
~ismember(TrackIndex,Ex_1_coreWest)
                                             NoObjection = 1;
                                        else NoObjection = 0; end
                                   end
                             else
                                  if East
                                        if  ismember(TrackIndex,Ex_0_East)
                                             NoObjection = 1;
                                        else NoObjection = 0;
                                        end
                                   else
                                        if  ismember(TrackIndex,Ex_0_West)
                                             NoObjection = 1;
                                        else NoObjection = 0;
                                        end

                                   end
```

```
                    end
elseif ExcAirSpace == 2 && RLatSM == 0
        if Equipped
            if East
                  if  ismember(TrackIndex,Ex_0_East)
                       NoObjection = 1;
                  else NoObjection = 0;
                  end
            else
                  if  ismember(TrackIndex,Ex_0_West)
                       NoObjection = 1;
                  else NoObjection = 0;
                  end

            end
        elseif FL < 36000
            if East
                  if  ismember(TrackIndex,Ex_0_East)
                       NoObjection = 1;
                  else NoObjection = 0;
                  end
            else
                  if  ismember(TrackIndex,Ex_0_West)
                       NoObjection = 1;
                  else NoObjection = 0;
                  end

            end
        else
            NoObjection = 0;
        end



 %%% Flight admission rules for RLatSM

elseif ExcAirSpace == 1 && RLatSM == 1
        if FL < 36000
                if East
                       if ismember(TrackIndex,Ex_0_East)
                                NoObjection = 1;
                       else NoObjection = 0; end

                else
                       if ismember(TrackIndex,Ex_0_West)
                                NoObjection = 1;
                       else NoObjection = 0; end

                end
        else
               if East
                   if ismember(TrackIndex,RLat_1_East)
                       if Equipped
                            NoObjection = 1;
                       else
```

```
                        if  ismember(TrackIndex,RLat_1_EastCore)
                            NoObjection = 0;
                        else NoObjection = 1;
                        end
                    end
                else NoObjection = 0;
                end
            else
                if ismember(TrackIndex,RLat_1_West)
                    if Equipped
                         NoObjection = 1;
                    else
                        if  ismember(TrackIndex,RLat_1_WestCore)
                            NoObjection = 0;
                        else NoObjection = 1;
                        end
                    end
                else NoObjection = 0;
                end
            end
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif ExcAirSpace == 2 && RLatSM == 1
    if FL >= 36000
            if ~Equipped
               NoObjection = 0;
            elseif East
                    if ismember(TrackIndex,RLat_1_East)
                        NoObjection = 1; CheckRandomCost = 0;
                        if RLongSM
                            RLong = 1;
                        end
                    else
                       NoObjection = 0; CheckRandomCost = 1;
                    end
            elseif West
                if ismember(TrackIndex,RLat_1_West)
                    NoObjection = 1; CheckRandomCost = 0;
                        if RLongSM
                            RLong = 1;
                        end
                else
                        NoObjection = 0; CheckRandomCost = 1;
                end
            else pause
                disp('Unable to resolve')
            end
    elseif East
                    if ismember(TrackIndex,Ex_0_East)
                        NoObjection = 1; CheckRandomCost = 0;
                    else
                       NoObjection = 0; CheckRandomCost = 1;
                    end
    elseif West
                    if ismember(TrackIndex,Ex_0_West)
                        NoObjection = 1; CheckRandomCost = 0;
                    else
```

```
                        NoObjection = 0; CheckRandomCost = 1;
                     end
        else
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    elseif ExcAirSpace == 2 && RLatSM == 2

           if East
               if ~Equipped
                   if FL >= 36000
                       if ismember(TrackIndex,East_Rnd_Trks)
                           NoObjection = 1; Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                       else
                           NoObjection = 0; CheckRandomCost = 1;
                       end
                   else
                       if ismember(TrackIndex,Ex_0_East)
                           NoObjection = 1;
                       elseif ismember(TrackIndex,East_Rnd_Trks)
                           NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                       else NoObjection = 0; end
                   end
               else
                   if FL >= 36000
                       if ismember(TrackIndex,RLat_2_East)
                            NoObjection = 1;CheckRandomCost = 0;
                            if RLongSM
                                RLong = 1;
                            end

                       elseif ismember(TrackIndex,East_Rnd_Trks)
                            NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                       else NoObjection = 0;CheckRandomCost = 1;
                       end
                   else
                           if ismember(TrackIndex,Ex_0_East)
                           NoObjection = 1;
                       elseif ismember(TrackIndex,East_Rnd_Trks)
                           NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                       else NoObjection = 0; end
                   end
               end
           elseif West
               if ~Equipped
                   if FL >= 36000
                       if ismember(TrackIndex,West_Rnd_Trks)
                           NoObjection = 1; Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                       else
                           NoObjection = 0; CheckRandomCost = 1;
                       end
                   else
```

```
                        if ismember(TrackIndex,Ex_0_West)
                            NoObjection = 1;
                        elseif ismember(TrackIndex,West_Rnd_Trks)
                            NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                        else NoObjection = 0;     end
                    end
                else
                    if FL >= 36000
                        if ismember(TrackIndex,RLat_2_West)
                             NoObjection = 1;CheckRandomCost = 0;
                             if RLongSM
                                 RLong = 1;
                             end

                        elseif ismember(TrackIndex,West_Rnd_Trks)
                             NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                        else NoObjection = 0;CheckRandomCost = 1;
                        end
                    else
                            if ismember(TrackIndex,Ex_0_West)
                            NoObjection = 1;
                        elseif ismember(TrackIndex,West_Rnd_Trks)
                            NoObjection = 1;Switch_To_RndTrk = 1;
CheckRandomCost = 0;
                        else NoObjection = 0;     end
                    end
                end
            end
    end
```