Aviation Global Demand Forecast Model Development: Air Transportation Demand Distribution and Aircraft Fleet Evolution

Edwin Ruben Freire Burgos

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, Chair

Linbing Wang

Montasir M. Abbas

September 8, 2017

Blacksburg, Virginia

Keywords: Air Travel Demand, Fratar Model, Trip Distribution, Worldwide Aircraft Fleet, Aircraft Fleet Evolution, New Generation Aircraft, NASA's Advanced Technology Aircraft

Aviation Global Demand Forecast Model Development: Air Transportation Demand Distribution and Aircraft Fleet Evolution

Edwin Ruben Freire Burgos

ABSTRACT

The Portfolio Analysis Management Office (PAMO) for the Aeronautics Research Mission Directorate (ARMD) at NASA Headquarters tasked the Systems Analysis and Concepts Directorate at NASA Langley to combine efforts with Virginia Tech to develop a global demand model with the capability to predict future demand in the air transportation field. A previous study (Alsalous, 2015) started the development of the Global Demand Mode (GDM) to predict air travel demand based on Gross Domestic Product (GDP) and population trends for 3,974 airports worldwide. The study was done from year 2016 to year 2040.

This research project intends to enhance the GDM capabilities. A Fratar model is implemented for the distribution of the forecast demand during each year. The Fratar model uses a 3,974 by 3,974 origin-destination matrix to distribute the demand among 55,612 unique routes in the network. Moreover, the GDM is capable to estimate the aircraft fleet mix per route and the number of flights per aircraft that are needed to satisfy the forecast demand. The model adopts the aircraft fleet mix from the Official Airline Guide data for the year 2015. Once the aircraft types are distributed and flights are assigned, the GDM runs an aircraft retirement and replacement analysis to remove older generation aircraft from the network and replace them with existing or newer aircraft. The GDM continues to evolve worldwide aircraft fleet by introducing 14 new

generation aircraft from Airbus, Boeing, Bombardier, and Embraer and 5 Advanced Technology Aircraft from NASA.

Aviation Global Demand Forecast Model Development: Air Transportation Demand Distribution and Aircraft Fleet Evolution

Edwin Ruben Freire Burgos

GENERAL AUDIENCE ABSTRACT

The Portfolio Analysis Management Office (PAMO) for the Aeronautics Research Mission Directorate (ARMD) at NASA Headquarters tasked the Systems Analysis and Concepts Directorate at NASA Langley to combine efforts with Virginia Tech to develop a global demand model with the capability to predict future demand in the air transportation field. A previous study (Alsalous, 2015) started the development of the Global Demand Mode (GDM) to predict air travel demand based on Gross Domestic Product (GDP) and population trends for 3,974 airports worldwide. The study was done from year 2016 to year 2040.

The previous study done by Alsaous, predicts how many seats will be departing out of the 3,974 airports worldwide. This project intends to use the outputs of the GDM and distribute the seats predicted among the airports. The objective is to predict how many seats will be offered that will be departing from airport “A” and arriving at airport “B”. For this, a Fratar model was implemented.

The second objective of this project is to estimate what will the aircraft fleet be in the future and how many flights will be needed to satisfy the predicted air travel demand. If the number of seats going from airport A to airport B is known, then, by analyzing real data it can be estimated what type of aircraft will be flying from airport “A” to airport “B”

and how many flights each aircraft will have to perform in order to satisfy the forecasted demand.

Besides of estimating the type of aircraft that will be used in the future, the modeled created is capable of introducing new aircraft that are not part of the network yet. Fourteen new generation aircraft from Airbus, Boeing, Bombardier, and Embraer and 5 Advanced Technology Aircraft from NASA.

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to Dr. Trani for his advice and mentorship during the past two years. He has been an excellent professor and mentor. I thank you for all the help, advice, and knowledge you shared with me. Your expertise in the air transportation field has been a great motivation for me.

I would like to thank Nicolas Hinze for teaching me how to improve the source code of the model and make it more efficient. Also, I would like to express my gratitude to my committee members Dr. Abbas and Dr. Wang for their guidance.

A special gratitude is given to Ty Vincent Marien and Sam Dollyhigh who are part of the team at NASA Langley Research Center for their guidance and input throughout the project. I am grateful for working as part of this great team.

Last but not least, I would like to express my deepest thanks to my family, to my parents and sister who believed in me way before all these was possible. I am who I am thanks to them.

Contents

List of Figures

List of Tables

1 Introduction

The Aeronautics Research Mission Directorate (ARMD) at NASA Headquarters is responsible for establishing a strategic systems analysis capability focused on understanding the system-level impacts of NASA programs, the potential for integrated solutions, and the development of high-level options for new investment and partnership. To this end, ARMD's Portfolio Analysis Management Office (PAMO) has tasked the Systems Analysis and Concepts Directorate at NASA Langley to formalize, develop, and utilize, a framework that efficiently employs a variable fidelity capability to aid in such assessments.

The Global Demand Model (GDM hereon) is a global aviation demand model that forecasts the annual commercial air traffic operations for 3,974 airports. The model uses Gross Domestic Product (GDP) and population trends to predict air transportation demand. The first objective of this research project is to improve the GDM capabilities by distributing the air transportation demand between all the airports in the network. A Fratar Model has been implemented to create such network and distribute the predicted demand by the GDM among all the airports.

The second objective of this research project is to develop an airline fleet assignment module to predict changes to the airline fleet in the future. Part of this objective is to understand how the worldwide aircraft fleet has been changing over time and how it could continue to evolve in the upcoming years. The analysis takes into consideration operational aircraft fleet trends between the years 2000 and 2015.

2 Global Demand Model

The Global Demand Model employs an econometric model to predict airport demand for 3,974 airports worldwide (Alsalous, 2015). This research project analyzes trip distribution methods that could be implemented in air transportation and aircraft fleet evolution over time. A literature review was prepared and is addressed in the following section.

2.1 Literature Review

With the purpose of accomplished a better understanding of transportation planning methods and worldwide aircraft fleet mix, a number of journal articles and other references were reviewed. There are several alternatives for trip distribution analysis. The literature review was focused on two of these techniques; the Gravity Model and the Fratar Model.

Comparison of Neural Networks and Gravity Models in Trip Distribution (Tillema, van Zuilekom et al. 2006) examined the performance of neural networks in trip distribution modeling and compares the results with commonly used doubly constrained gravity models. The research work concluded that neural networks provided better results compared to gravity model outputs when data are scarce. The authors also concluded that even when data is not scarce the assumption that the gravity model could outperform neural networks seems less certain. A final conclusion indicates that the neural networks can improve trip distribution analysis; however, at a higher level of difficulty, it would be more complex than the gravity model.

Excellent research work that implement the Gravity Model for trip distribution analysis are: Exact methods for gravity trip-distribution models (Holmberg and Jörnsten 1989), Gravity models for airline passenger volume estimation (Grosche, Rothlauf et al. 2007), and Utilizing Traveler Demand Modeling to Predict Future Commercial Flight Schedule in the NAS (Viken, Dollyhigh et al. 2006).

The Gravity Model would have a high level of difficulty if implemented for this research project. This method, following Newton's gravitational law, is based on the assumption that all trips starting from a given zone are attracted by the various traffic generators. This attraction is in direct proportion to the size of the generator and in inverse proportion to the spatial separation between the areas (Heanue and Pyers 1966). Due to the high level of complexity for the problem established on this project which tries to analyze a single worldwide 3,974 by 3,974 origin-destination matrix; among other parameters; the use of the gravity model was considered but not selected.

Prediction of Future Origin-Destination Matrix of Air Passenger by Fratar and Gravity Model (Ceha and Ohta 1997) combined two problems of the transportation forecasting analysis in the air transportation field. At the time of this study, the authors indicated the need to predict what would be the future air transportation demand regarding passengers and the schedule of commercial airlines. For the first part of the analysis which was the generation of air travel demand the authors implemented an ordinary least square model using historical data. For the second part of the analysis, the authors incorporated two methods to distribute the demand. First, the Gravity model was used for measuring and distributing the trip generation as being directly related to the number of passengers between airports. Secondly, the Fratar model was

implemented to estimate the future origin-destination matrix between locations. It seems that the use of the Fratar model to distribute the demand was showing satisfactory and reasonable results. On this research, it was concluded that even when no definitive conclusions were drawn about the procedure applied, the origin-destination matrix was most useful for planning tool in various air transportation studies. It is interesting how after many years of research and the development of new technology prediction of future demand can be uncertain.

2.2 Methodology and Assumptions

The Global Demand Model employs an econometric model to predict airport demand (seats) between 2016 and 2040 for 3,974 airports worldwide (Alsalous, 2015). Seats are used as a surrogate for passenger demand in this model since there are no global databases containing the number of passengers between each origin and destination airport for 3,974 airports. On this research project, the capabilities of the GDM are improved by incorporating a trip distribution analysis and creating an aircraft fleet evolution.

2.2.1 Fratar Model

The GDM creates a global airport network and employs the Fratar method to distribute future airline seats. There are several growth-factor-based trip distribution models such as Uniform Growth Factor, Single-Constrained Growth Factor, Average Growth Factor, Detroit Growth Factor, Fratar Method, among others. The Fratar method has been proven to be computationally the most efficient of the growth factor alternatives (Heanue and Pyers 1966). The principle of the Fratar method is that using an existing data set as a baseline the distribution of trips from a zone is proportional to the current trips

departing the zone modified by a growth factor from the zone to which the trips are being attracted.

Data from the Official Airline Guide (OAG) for the year 2015 is used as the baseline year. The OAG data is extracted, organized, and analyzed to generate the global demand distribution that happened during the year of 2015. This procedure creates a 3,974 by 3,974 origin-destination matrix to be used as baseline data or “current/present” trips. Growth factors are calculated for all the airports in the network for each individual year from 2016 to 2040.

The method utilizes a symmetric origin-destination matrix through the entire analysis. This forces the mathematical procedure to establish two main assumptions. First, the number of seats offered by departing flight at an origin airport must be equal to the number of seats offered by arriving flight at the same airport. Second, the number of seats from an origin airport (i) to a destination airport (j) which is represented by (T_{i-j}), must be equal to the number of seats offered from a destination airport (j) to origin airport (i). These assumptions create a conservation of flows between origin and destination airports throughout the entire network.

The assumptions create an ideal condition which is rare to happen in reality due to numerous factors. For example, in the first assumption, OAG 2015 data indicates that the number of seats offered from John F. Kennedy International Airport (JFK) to Adolfo Suárez Madrid–Barajas Airport (MAD) was 425,925 seats. For the opposite route, the number of seats offered from MAD airport to JFK airport was 426,859 seats which indicate a difference of 0.22%. For the second assumption, the total number of seats offered by departing flight and arriving flights at JFK airport were 35,051,804 seats and

35,048,091 seats respectively. In the case of MAD airport, the total number of seats offered by departing flight and arriving flights were 29,139,962 seats and 29,102,626 seats respectively. These values indicate a difference of 0.01% for JFK airport and 0.13% for MAD airport.

In order to comply with the assumption of the Fratar method, the values for each origin-destination pair in the matrix are averaged. In the previous example, once the assumptions are implemented the number of seats from JFK airport to MAD airport and from MAD airport to JFK airport are both equal, 426,392 seats. The implementation of the first assumption forces the second assumption to happen. Therefore, the total number of seats offered by departing flight and arriving flights become equals. For JFK airport, the value obtained is 35,049,948 seats and for MAD airport the value obtained is 29,121,294 seats. Even when real data provided by OAG is used, due to the initial assumption of the Fratar method the values becomes slightly different from the original data. The model assumes that the number of airports is constant over the forecast period. For that reason, current routes are kept and new routes are not created.

The Fratar model is defined in Equation (1).

$$T_{ij,y} = t_{ij} * G_{i,y} * G_{j,y} * K_{i,y} \quad (1)$$

where,

$T_{ij,y}$ = Future seats distribution between origin airport (i) and destination airport (j)

$t_{ij,y}$ = Current seats distribution between origin airport (i) and destination airport (j)

$G_{i,y}$ = Growth factor at origin airport (i) for year (y)

$G_{j,y}$ = Growth factor at destination airport (j) for year (y)

$K_{i,y}$ = Growth factor between origin airport (i) and each of the destination airports (j)

i = Airport of origin

j = Airport of destination

y = Year being analyzed

The growth factor of each airport is calculated by dividing the forecast demand (GDM phase 1 output) by the current demand at the airport of interest (OAG 2015 data). The fourth factor of the Fratar equation is defined in Equation (2).

$$K_{i,y} = \frac{\sum_{j=1}^{n} t_{ij}}{\sum_{j=1}^{n} G_{j,y} * t_{ij}} \quad (2)$$

where,

$K_{i,y}$ = Growth factor between origin airport (i) and each of the destination airports (j)

i = Airport of origin

j = Airport of destination

n = Total number of destinations for origin airport (i)

$G_{j,y}$ = Growth factor at destination airport (j) for year (y)

The model uses OAG 2015 data as initial demand (t_{i-j}) to predict future demand between airports ($T_{ij,y}$)

2.2.2 Global Aircraft Fleet Analysis

A further step of the GDM, which is not discussed in this document, is to estimate fuel and emission impacts worldwide. The GDM predicts air travel demand (number of seats) for 3,974 airports and distributes that demand over 55,612 unique origin-destination pairs in the network. An aircraft fleet evolution model to allocate aircraft type

and the number of flights for each individual route that can satisfy the GDM demand forecast was developed in order to conduct the analysis for phase 3.

The equations employed in the aircraft flights assignment are presented in Equations (3-5).

$$A\_R_{ij}^{k} = \frac{t_{ij}^{k}}{t_{ij}} \tag{3}$$

$$T_{ij,y}^{k} = T_{ij,y} * A\_R_{ij}^{k} \tag{4}$$

$$F_{ij,y}^{k} = \frac{T_{ij,y}^{k}}{SC_x} \tag{5}$$

where,

$A\_R_{ij}^{k}$ = Aircraft ratio for aircraft type (k) from origin airport (i) to destination airport (j)

t_{ij}^{k} = Seats offered by aircraft type (k) from origin airport (i) to destination airport (j) during baseline year

t_{ij} = Total seats offered from origin airport (i) to destination airport (j) during baseline year

$T_{ij,y}^{k}$ = Seats assigned to aircraft type (k) from origin airport (i) to destination airport (j) in year (y)

$T_{ij,y}$ = Total seats from origin airport (i) to destination airport (j) in year (y)

$F_{ij,y}^{k}$ = Flights assigned to aircraft type (k) from origin airport (i) to destination airport (j) in year (y)

SC_x = Seating capacity of aircraft type (k)

From OAG 2015 the aircraft fleet distribution, aircraft utilization ratio and flight assignment for all the origin-destination pairs was obtained. Table 1 presents an example of aircraft fleet distribution and aircraft utilization ratio for a specific route. The

route used for this example is from London Heathrow Airport (LHR) to Dubai International Airport (DXB). According to OAG 2015 data, the total number of seats offered on this particular route was 1,704,508 seats. The aircraft fleet was Airbus 330-300, Airbus 380-800, Boeing 747-400, Boeing 777-200, and Boeing 787-8. The number of seats that each aircraft type offered can be obtained from the data. The aircraft utilization ratio is calculated as follows: the number of seats offered divided by the total demand for a particular route. These details are presented in Table 1 and indicate that 76% of the total demand was offered by the Airbus 380-800 and only 5% of the total demand offered by the Boeing 787-8. The Airbus 380-800, introduced on 2007 has a higher seating capacity than the Boeing 787-8 which was introduced on 2011. This mathematical procedure is performed for each of the 55,612 origin-destination pairs.

Table 1: Example of Aircraft Fleet Distribution and Aircraft Utilization Ratio (OAG 2015) between London Heathrow Airport (LHR) AND Dubai International Airport (DXB).

Aircraft Type	Seats Offered	Aircraft utilization Ratio
Airbus 330-300	102,270	0.06
Airbus 380-800	1,295,426	0.76
Boeing 747-400	119,316	0.07
Boeing 777-200	102,270	0.06
Boeing 787-8	85,225	0.05

Aircraft from OAG 2015 data were categorized into 39 unique commercial aircraft types. Table 2 presents the list of the 39 unique commercial aircraft along with an average aircraft seating capacity.

Table 2: 2015 Commercial Aircraft List and Seating Capacity.

Aircraft	Average Seating Capacity	Aircraft	Average Seating Capacity
Airbus 310	218	Boeing 767-300	261
Airbus 319	128	Boeing 767-400	296
Airbus 320	150	Boeing 777-200	313
Airbus 321	187	Boeing 777-300	396
Airbus 330-200	234	Boeing 777-200L	313
Airbus 330-300	293	Boeing 777300-W	340
Airbus 340-600	300	Boeing 787-8	242
Airbus 380-800	490	Beechcraft 99	15
Avions de Transport Régional 42-500	47	Cessna 208	14
Avions de Transport Régional 47-500	70	Canadair Challenger 600	14
Boeing 717-200	106	Bombardier Regional Jet CRJ-200	50
Boeing 737-300	128	Bombardier Regional Jet CRJ-900	76
Boeing 737-500	108	Bombardier Havilland Dash8-200	40
Boeing 737-700	128	Bombardier Havilland Dash8-300	78
Boeing 737-800	160	Embraer ERJ-135	37
Boeing 737-900	174	Embraer ERJ-145	50
Boeing 747-400	416	Embraer ERJ-170	70
Boeing 747-800	410	Embraer ERJ-190	94
Boeing 757-200	200	McDonnell Douglas MD-82	155
Boeing 767-200	216		

The number of flights needed (per aircraft) to satisfy the demand is calculated as follows. The number of seats assigned divided by seating capacity. After the process is completed the following data is known: aircraft fleet type used on every route and the number of flights that each aircraft will perform in order to satisfy the forecast demand.

2.2.2.1 Aircraft Retirement and Replacement Analysis

Aircraft fleets are expected to evolve over time. New generation aircraft such as the Airbus 320 neo, Boeing 737-8MAX, and the Bombardier CS 100/300 has already started to replace older generation of commercial aircraft. Likewise, this trend of older generation of commercial aircraft being replaced by new generation aircraft will continue in the next decades as aircraft manufacturers such as Airbus and Boeing have received combined orders for more than 8,500 single-aisle aircraft.

Aircraft utilization trends between years 2000 to 2015 were analyzed (OAG data). Aircraft showing a decreasing utilization trend will be the candidates to be removed from the network in the future. Similarly, those aircraft showing an increasing utilization trend, in some cases, will be the candidates to replace those aircraft that will be removed from the network.

The aircraft utilization is based on the number of routes flew by an aircraft. Those aircraft with decreasing utilization trends (i.e., candidate for aircraft retirement) are presented in Figure 1 and Figure 2. The data shows faster retirement trends for the Boeing 757-200 and Boeing/McDonnell Douglas MD-80s from global fleets. Also, recent trends show that airlines retire aircraft when they reach 27 years. On the contrary, those aircraft with increasing utilization trends (i.e., candidates to replace the retiring aircraft) are presented in Figure 3 and Figure 4. The data shows a fast growth on aircraft such

as Boeing 737-800, Boeing 777-300, and Embraer 170. Table 3 presents the retiring aircraft, the existing candidate aircraft expected to replace them, and the expected final year of the retirement process. Aircraft size and seating capacity were used to match retiring aircraft with the corresponding aircraft of replacement. The retirement and replacement analysis is a process that occurs gradually over time. The year of retirement for an aircraft model presented in Table 3 was estimated by combining aircraft utilization trends and assuming 27 years of average commercial aircraft use. The age of the commercial fleet was estimated using the BuchAir commercial fleet database (BuchAir, 2013)

The equations employed in the aircraft retirement and replacement analysis are presented in Equations (6-7).

$$D_{ij,y}^{k} = D_{ij,y}^{m} * P_{ij,y}^{k} \quad (6)$$

$$D_{ij,y\,(new)}^{m} = D_{ij,y}^{m} - D_{ij,y}^{k} \quad (7)$$

where,

$D_{ij,y}^{k}$ = Demand to be replaced by aircraft type (k) from aircraft type (m) from origin airport (i) to destination airport (j) in year (y)

$D_{ij,y}^{m}$ = Demand assigned to aircraft type (m) from origin airport (i) to destination airport (j) in year (y)

$P_{ij,y}^{k}$ = Percent of replacement of aircraft type (k) from origin airport (i) to destination airport (j) in year (y)

$D_{ij,y\,(new)}^{m}$ = New demand for the retiring aircraft (m)

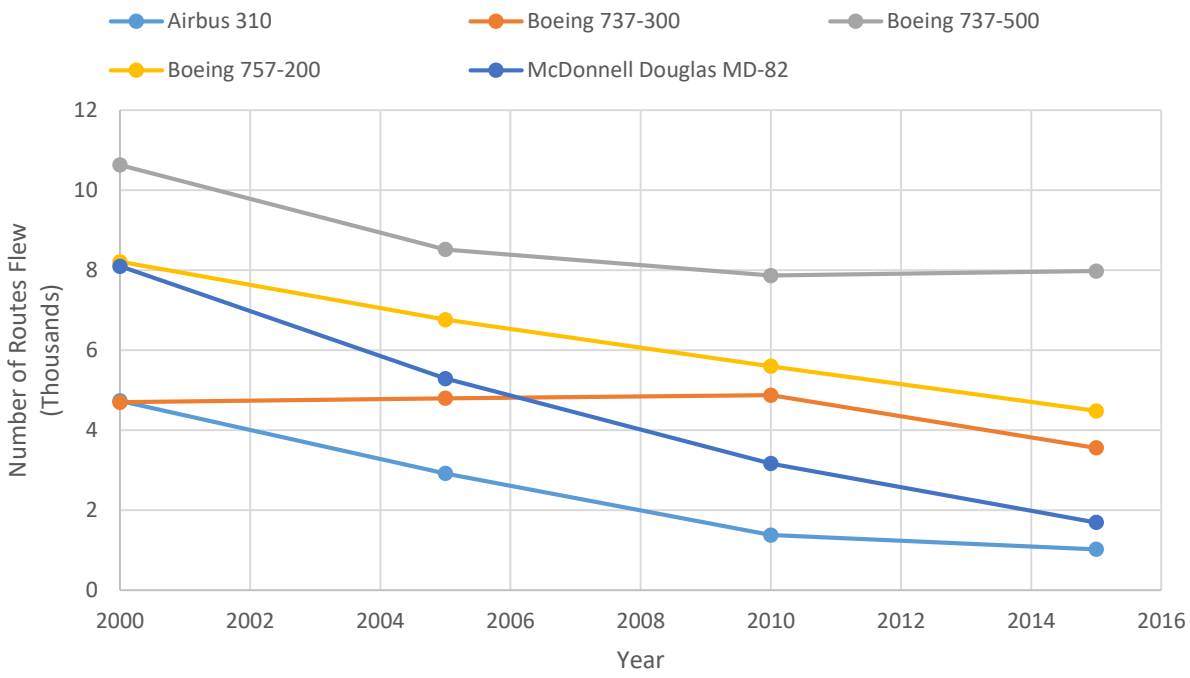


Figure 1: Aircraft Utilization Trends Over Time for Selected Aircraft with Decreasing Utilization Trends. Aircraft Set #1.

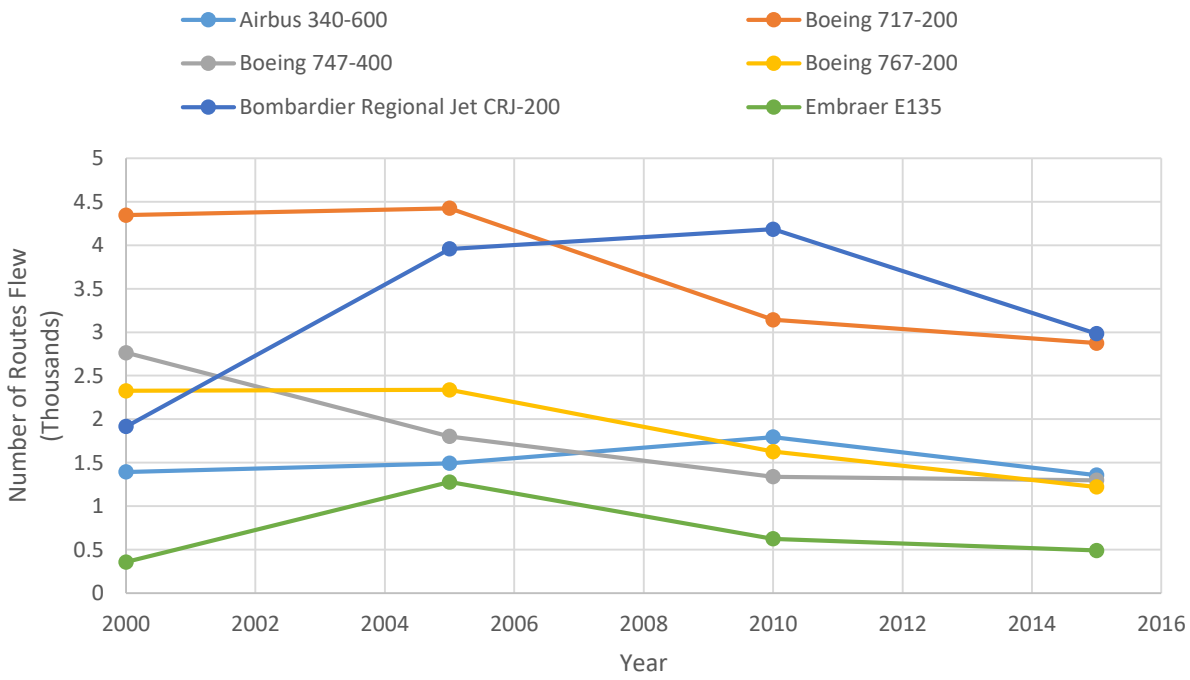


Figure 2: Aircraft Utilization Trends Over Time for Selected Aircraft with Decreasing Utilization Trends. Aircraft Set #2.

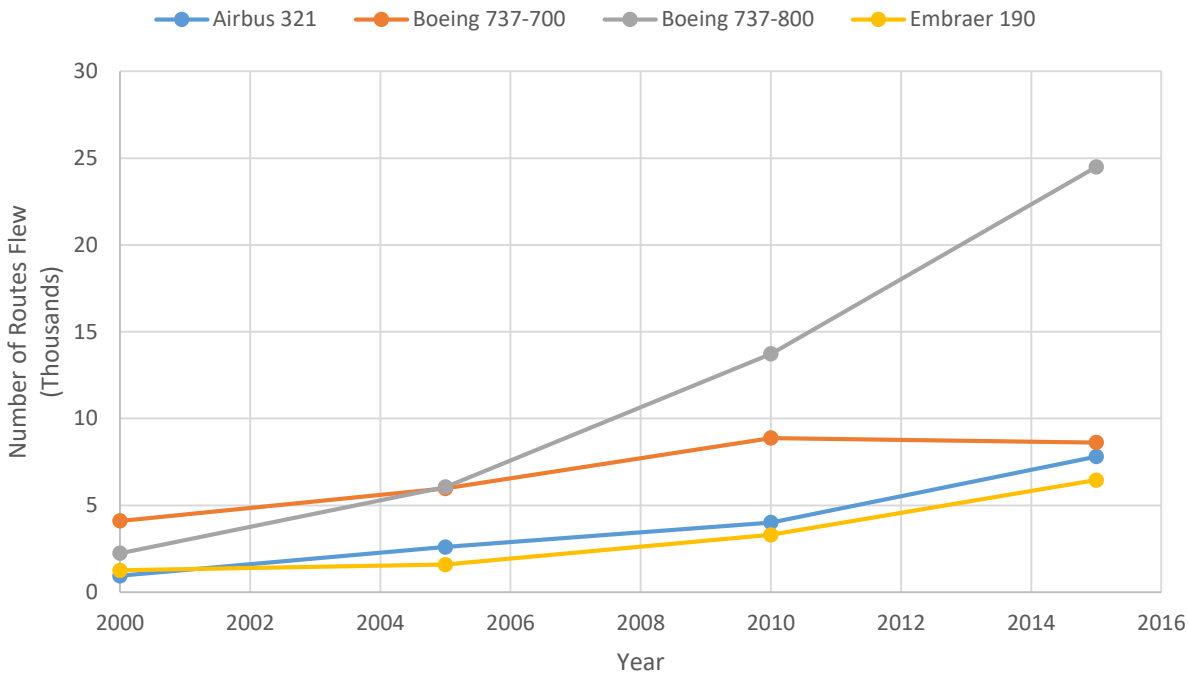


Figure 3: Aircraft Utilization Trends Over Time for Selected Aircraft with Increasing Utilization Trends. Aircraft Set #1.

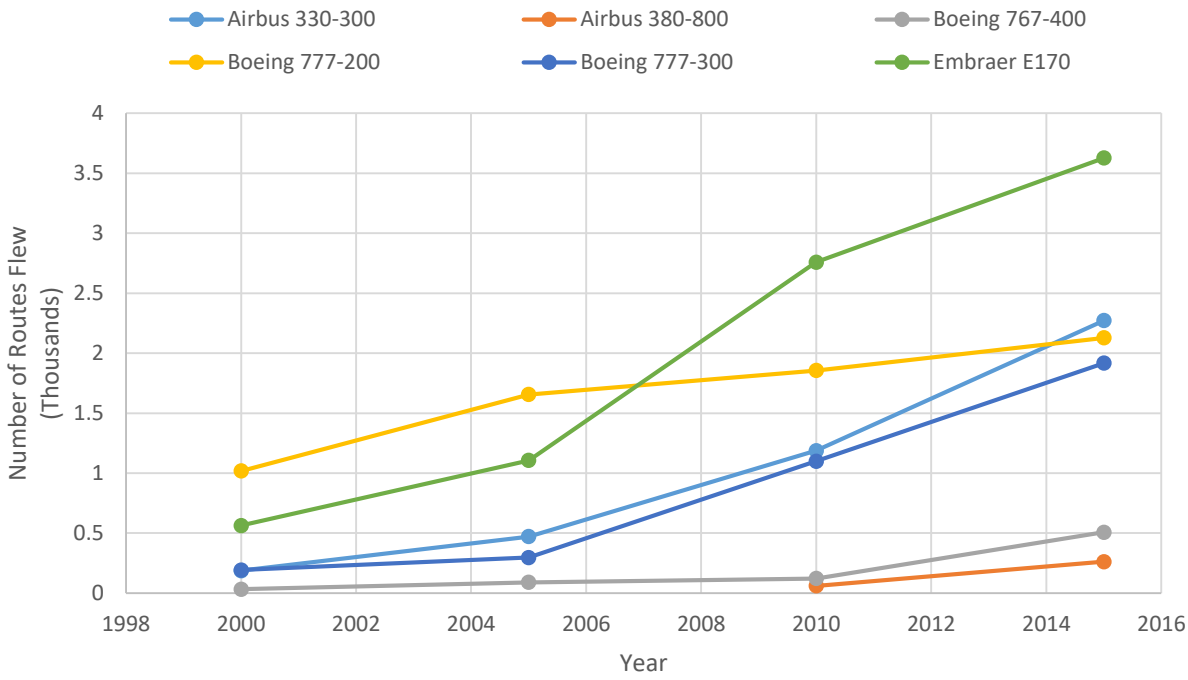


Figure 4: Aircraft Utilization Trends Over Time for Selected Aircraft with Increasing Utilization Trends. Aircraft Set #2.

Table 3: Retiring and Replacement Aircraft Model.

Retiring Aircraft	Replacement Aircraft	Year of Retirement
Airbus 310	Airbus 330-200	2018
Airbus 340-600	Airbus 350 Boeing 777-300	2019
Boeing 717-200	Embraer E190	2028
Boeing 737-300-	Boeing 737-700	2019
Boeing 737-500	Boeing 737-700	2026
Boeing 747-400	Airbus 380-800 Boeing 777-300	2020
Boeing 757-200	Airbus 321 Airbus 330-300	2029
Boeing 767-200	Boeing 787-800	2022
Bombardier Regional Jet CRJ-200	Bombardier Regional Jet CRJ-200 Embraer E170	2020
Embraer E135	Embraer E135 Embraer E170	2020
McDonnell Douglas MD-82	Boeing 737-800	2018

As presented in Table 3, there are three different cases for the aircraft retiring and replacement process. First, an aircraft is replaced by another aircraft. For example, the Airbus 310 is replaced by the Airbus 330-200. Second, an aircraft is replaced by two different aircraft. For example, the Boeing 757-200 is replaced by the Airbus 321 and the Airbus 330-300. The third and final case is where an aircraft is partially replaced. For example, the utilization of the Embraer 135 will decrease but will continue to operate and part of the demand will be re-assigned to the Embraer E170. Table 4 shows the replacement percentages for aircraft with two replacement aircraft. The other retiring aircraft will be 100% replaced by the corresponding replacement aircraft. In the

example of the Embraer E135, this aircraft will continue to operate 40% of its current demand (demand assigned on the aircraft fleet distribution and flights assignment process) and the remaining 60% will be replaced by the Embraer E170. This aircraft retirement and replacement analysis is done for all year, from 2016 to 2040.

Table 4: Aircraft Replacement Percentage for Retiring Aircraft with 2 Aircraft of Replacement.

Retiring Aircraft	Replacement Aircraft	Replacement %
Airbus 340-600	Airbus 350	50%
	Boeing 777-300	50%
Boeing 747-400	Airbus 380-800	20%
	Boeing 777-300	80%
Boeing 757-200	Airbus 321	50%
	Airbus 330-300	50%
Bombardier Regional Jet CRJ-200	Bombardier Regional Jet CRJ-200	50%
	Embraer E170	50%
Embraer E135	Embraer E135	40%
	Embraer E170	60%

2.2.2.2 Introduction of New Generation Aircraft into the Network

Aircraft manufacturers continuously upgrade their existing aircraft products. Newly introduced aircraft are expected to replace older generation aircraft flying today. For example, Airbus certified the Airbus A320 with a new engine option (neo) in 2016. Airbus provides two high by-pass ratio engine options: the GE/Snecma CFM International LEAP-1A and the Pratt and Whitney PW1000G. The new engines provide 12%-15% in fuel savings over older generation aircraft. Similarly, Boeing introduced the Boeing 737-8 MAX in June of 2017 with the new GE/Snecma CFM International LEAP-

1B engine. Airbus has more than 5,054 Airbus A320neo variant orders and Boeing has over 3,500 orders of the 737 MAX family.

The GDM introduces new generation aircraft to replace older generation aircraft. For example, some of the aircraft that did not show a decreasing trend could retire after 2040 but it will start its retiring process earlier. These new generation aircraft are named N+1 aircraft in the analysis. Table 5 presents the list of the new generation aircraft (N+1). The N+1 aircraft are from Airbus, Boeing, Bombardier, and Embraer manufacture. The GDM does not include future new generation aircraft expected from Russia and China such as the Irkut MC-21 and the Comac 919 respectively. However, the GDM does estimate aircraft fleet distribution and flight assignment for these two countries using N+1 aircraft to satisfy the forecast demand.

The new generation aircraft (N+1) has a maximum estimated annual production rate that has been obtained from publicly available aircraft manufacturer data. Table 5 shows the annual production rate for each N+1 aircraft and the year in which the aircraft is expected to be introduced into the network. Besides the N+1 aircraft, NASA has proposed the introduction of 5 new advanced technology aircraft. The five new aircraft are named NASA's N+2 aircraft. Table 5 presents the maximum annual production rate for NASA's N+2 aircraft. The five NASA's N+2 aircraft design introduced by Nickol and Haller (2016) are shown in Figure 5. The GDM assumes that these new aircraft from NASA could be available in the year 2030.

The equations employed in the new generation aircraft flights assignment are presented from Equations (8-10).

$$D_{ij,y}^{k} = D_{ij,y}^{m} * P_{ij,y}^{k} \quad (8)$$

$$D_{ij,y\,(new)}^{m} = D_{ij,y}^{m} - D_{ij,y}^{k} \quad (9)$$

$$H_{y,(new)}^{k} = H_{y}^{k} - (TT_{ij} * D_{ij,y}^{k}) \quad (10)$$

where,

$D_{ij,y}^{k}$ = Demand to be replaced by aircraft type (k) from aircraft type (m) from origin airport (i) to destination airport (j) in year (y)

$D_{ij,y}^{m}$ = Demand assigned to aircraft type (m) from origin airport (i) to destination airport (j) in year (y)

$P_{ij,y}^{k}$ = Percent of replacement of aircraft type (k) from origin airport (i) to destination airport (j) in year (y)

$D_{ij,y\,(new)}^{m}$ = New demand for the retiring aircraft (m)

H_{y}^{k} = Available flight-hours for aircraft type (k) before the assignment of new demand

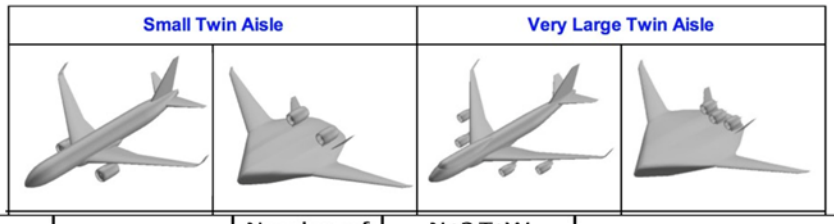
$H_{y,(new)}^{k}$ = Available flight-hours for aircraft type (k) after the assignment of new demand

TT_{ij} = Travel time from origin airport (i) to destination airport (j)

Table 5: New Generation Aircraft, Maximum Annual Production Rate, and Year of Introduction into Service.

Aircraft	Aircraft Name	Annual Production Rate	Introduction Year
New Generation N+1 Aircraft	Airbus 319neo	60	2021
	Airbus 320neo	504	2017
	Airbus 321neo	200	2019
	Airbus 330neo	120	2017

Aircraft	Aircraft Name	Annual Production Rate	Introduction Year
	Airbus 350-1000	48	2019
	Boeing 737-7MAX	100	2018
	Boeing 737-8MAX	200	2018
	Boeing-737-9MAX	100	2018
	Boeing 777-8X	50	2021
	Boeing 777-9X	50	2021
	Bombardier CS100	45	2017
	Bombardier CS300	45	2017
	Embraer E190E2	48	2019
	Embraer E195E2	48	2019
NASA's N+2 Aircraft	T+W 98	60	2030
	T+W 160	240	2030
	T+W 216	108	2030
	T+W 301	96	2030
	T+W 400	36	2030



Vehicle Class	Abbreviation	Number of Passengers	N+2 T+W Nomenclature	Unconventional	Abbreviation
Regional Jet	RJ	98	T+W98	Over-Wing-Nacelle	OWN98
Single Aisle	SA	160	T+W160	Over-Wing-Nacelle	OWN160
Small Twin Aisle	STA	216	T+W216	Hybrid-Wing-Body	HWB216
Large Twin Aisle	LTA	301	T+W301	Hybrid-Wing-Body Mid-Fuselage Nacelle	HWB301 MFN301
Very Large Twin Aisle	VLTA	400	T+W400	Hybrid-Wing-Body	HWB400

Figure 5: NASA's N+2 Aircraft (Nickol and Halley, 2016).

The aircraft retirement and replacement analysis is executed for 55,612 unique routes. Up to this point on the analysis, all the aircraft in the network are aircraft that showed an increase utilization trend. After the aircraft retirement and replacement analysis, the introduction of the new generation N+1 aircraft and NASA's N+2 aircraft is implemented. The GDM relies on four parameters criteria in order to identify the routes in where the N+1 and NASA's N+2 aircraft will be introduced.

First, an aircraft matched is established based on the aircraft type and the average seating capacity of the aircraft. The GDM start introducing the new generation aircraft one by one beginning with the N+1, followed by NASA's N+2 aircraft. It identifying if a similar aircraft to the one being introduced is being used in the route. The aircraft mapping between the new generation aircraft and the so-called similar aircraft (aircraft that are already part of the network) is presented in Table 6 and Table 7. Similar to the retirement and replacement analysis, NASA's N+2 aircraft will be replacing more than one aircraft type. The GDM provides a higher opportunity for demand replacement to NASA's N+2 aircraft by matching the aircraft with more than one option.

Table 6: N+1 Aircraft and Similar Aircraft Based on Average Seating Capacity and Aircraft Type.

N+1 Aircraft	Similar Aircraft	N+1 Aircraft	Similar Aircraft
Airbus 319neo	Airbus 319	Boeing 737-9MAX	Boeing 737-900
Airbus 320neo	Airbus 320	Boeing 777-8X	Boeing 777-300
Airbus 321neo	Airbus 321	Boeing 777-9X	Boeing 777-300
Airbus 330neo	Airbus 330-300	Bombardier CS100	Airbus 319
Airbus 350-1000	Airbus 350	Bombardier CS300	Airbus 320
Boeing 737-7MAX	Boeing 737-700	Embraer E190E2	Embraer E190
Boeing 737-8MAX	Boeing 737-800	Embraer E195E2	Airbus 320

Table 7: NASA's N+2 and Similar Aircraft Based on Average Seating Capacity and Aircraft Type.

NASA's N+2 Aircraft	Similar Aircraft
T+W 98	Bombardier Regional Jet CRJ-900 Embraer E170 Embraer E190 Bombardier CS100 Bombardier CS300 Embraer E190E2 Embraer E195E2
T+W 160	Airbus 319neo Airbus A320neo Boeing 737-7MAX Boeing 737-8MAX
T+W 216	Airbus 321neo Boeing 737-9MAX
T+W 301	Airbus 330neo Airbus 350 Boeing 787-8 Boeing 777-300 Boeing 777-8X
T+W 400	Airbus 380-800 Boeing 747-8 Boeing 777-300 Boeing 777-200LR Boeing 777-300ER Boeing 777-9X Boeing 350-1000

Second, the GDM selects only heavily use routes base on demand. These routes are identified by calculating the growth between 2016 and 2040. The growth is calculated for each route as follows. Total demand (number of seats) in the year 2040 divided by

the total demand in the year 2016. If the growth is equal to 1.5 or higher, then the route becomes a potential candidate to introduce the N+1 aircraft or NASA's N+2 aircraft.

Third, aircraft do not fly 24 hours/day and or 365 days/year. For this reason, the GDM constantly check against an available flight-hours constraint. N+1 aircraft and NASA's N+2 aircraft are subjected to this constraint. Using data from the "MIT Airline Data Project" (http://web.mit.edu/airlinedata/www/default.html) a total of 3,630 hours per year are assigned to each N+1 aircraft and NASA's N+2 aircraft. Using a travel time input table the GDM tries to introduce an aircraft (if criteria are meet). It checks if the aircraft has enough flight-hours available to fly the route. If the aircraft has enough flight-hours available and meet the other criteria, it is introduced into the network. The number of hours required to fly the route is deducted from the aircraft available flight-hours.

Fourth, if all the criteria are meet the GDM will introduce the aircraft into the network and will start assigning demand to the aircraft by constantly checking against the flight-hours constraint. For N+1 aircraft and NASA's N+2 aircraft the GDM will try to replace up to 40% and up to 60%, respectively, of the demand that was assigned to the similar aircraft (see Table 6 or Table 7 as needed). For example, the GDM will check if a particular N+1 aircraft has enough flight-hours available to replace 40% of the demand of its similar aircraft. For this, the GDM simultaneously verifies and combine the number of flights (demand) that the 40% would represent, travel time, and flight-hours available to check if the replacement of 40% of the demand is possible. If not, it will check for 39% of the demand until either reaching a feasible percent for demand replacement or reaching 0% and moving on to the next aircraft type. If 0% of the demand is reached it means that the aircraft has zero flight-hours available. In this case, the GDM will not

allow the assignment of more demand for that particular aircraft. This process is done for 55,612 routes from years 2016 to 2040.

A final step in the GDM is to back engineer part of the procedure to calculate the total number of aircraft in the network. At first, aircraft and flights were assigned to satisfy the forecast demand without being constrained to a flight-hours per aircraft limitation. This constraint is only applied to new generation aircraft (N+1 aircraft and NASA's N+2 aircraft). For this reason, the model relies on calculated travel times between each origin-destination pair to convert from number of flights to number of aircraft

2.3 Scenarios Analyzed

The GDM estimates global air travel demand (seats), demand distribution, aircraft fleet mix, and the number of flights per aircraft between years 2016 and 2040. Four scenarios were created to test the model. Table 8 presents a summary of these four scenarios.

Table 8: Summary of the Four Scenarios Analyzed in the GDM Model.

Parameters	Scenario			
	1	1.5	2	3
Demand distribution by Fratar method	X	X	X	X
Do Nothing – Aircraft fleet is not altered throughout the analysis	X			
Introduction of N+1 aircraft into the network		X	X	X
Introduction of NASA's N+2 aircraft into the network			X	X
Introduce of new generation aircraft to routes with growth > 50%		X	X	
Introduce new generation aircraft in all routes				X
Flight-hour constraint for N+1 and NASA's N+2 aircraft		X	X	X

Parameters	Scenario			
	1	1.5	2	3
Add N+1 and NASA's N+2 aircraft to routes flown with similar aircraft		X	X	X
Baseline production rate of N+1 aircraft		X	X	X
Baseline production rate of NASA's N+2 aircraft			X	
High production rate of NASA's N+2 aircraft				X

2.3.1 Description of Scenario 1

This scenario is the "do nothing" alternative. After the Fratar model distributes the demand (see section 2.2.1), only one further step is considered. The OAG 2015 is analyzed and aircraft fleet distribution of that year is replicated. The aircraft fleet mix and flights assignment is based only on OAG 2015 data without contemplating any changes in the future. Table 2 showed the aircraft used for this scenario. The objective of this baseline scenario is to understand a future where the current aircraft fleet mix continues to operate without the introduction of new generation aircraft.

2.3.2 Description of Scenario 1.5

The scenario 1.5 takes into consideration changes in the global aircraft fleet mix. The Fratar model (see section 2.2.1) is implemented for demand distribution. As in scenario 1, the aircraft fleet distribution and aircraft utilization ratio is adopted as a starting point for the aircraft evolution process. Once the aircraft fleet mix and number of flights are determined, the aircraft retirement and replacement analysis is conducted. The description of this process can be found in section 2.2.2.1.

The GDM will continue to execute the introduction of new generation aircraft into the network, which is described in section 2.2.2.2. Only the new generation N+1 aircraft will

be introduced into the network. The aircraft that the N+1 aircraft will be replacing were presented in Table 6 and the final year of retirement for those aircraft being removed from the network over time are presented in Table 9. The route selection to introduce a new aircraft will occur if the four parameters criteria are met. Table 10 presents the description of these parameters.

Table 9: Retiring and New Generation N+1 Aircraft Assumptions in Scenario 1.5.

Retiring Aircraft	Year of Retirement	Replacement Aircraft
Airbus 319	2030	Airbus 319neo
Airbus 320	2030	Airbus 320neo
Airbus 321	2030	Airbus 321neo
Airbus 330-200	2030	Airbus 330neo
Airbus 330-300	2030	Airbus 330neo
Boeing 737-700	2030	Boeing 737 MAX 7
Boeing 737-800	2030	Boeing 737 MAX 8
Boeing 737-900	2030	Boeing 737 MAX 9
Boeing 767-300	2025	Boeing 787-800
Boeing 767-400	2025	Boeing 787-800
Boeing 777-200	2030	Boeing 777-300
Bombardier Regional Jet CRJ-900	2035	Bombardier CS100
Embraer E170	2035	Bombardier CS100
Embraer E190	2035	Bombardier CS100

Table 10: Conditions to be meet for the Introduction of an Aircraft into any Given Route.

Condition	Description
1	A similar aircraft based on aircraft type and seating capacity must be part of the route.
2	For any given aircraft the annual production rate cannot be exceeded.
3	Check against the available flight-hours constraint
4	Routes with a growth greater than 50% between years 2016 and 2040

2.3.3 Description of Scenario 2

The scenario 2 follows the same procedure as scenario 1.5. It uses the demand distribution from the Fratar model (see section 2.2.1), distributes aircraft fleet mix based on OAG 2015 data, and assigned flights to satisfy the forecast demand. The retirement and replacement analysis (see section 2.2.2.1) and the introduction of new generation aircraft (see section 2.2.2.2) are analyzed. A further step is to analyze the introduction of NASA's N+2 aircraft into the network. The annual production rates and the year of introduction of aircraft are presented in Table 5. The route selection to introduce a new aircraft will occur if the four parameters criteria described in Table 10 are met.

2.3.4 Description of Scenario 3

The scenario 3 follows the same procedure and assumptions as scenario 2. However, it changes two parameters. First, the priority to routes with growth greater than 50% between years 2016 and 2040 is eliminated. This enlarges the potential candidate routes for the introduction of N+1 and NASA's N+2 aircraft. Second, the production rate for NASA's N+2 aircraft is increased. A high production rate, presented in Table 11 is assigned to NASA's N+2 aircraft in order to increase the total demand that can be

assigned to these aircraft. Table 11 shows both, baseline production rate and the new high production rate for a simpler comparison.

Table 11: Maximum Annual Production Rates for NASA's N+2 Aircraft for Scenarios 2 and 3.

NASA's N+2 Aircraft	Annual Production Rate Scenario 2	Annual Production Rate Scenario 3
T+W 98	60	200
T+W 160	240	600
T+W 216	108	128
T+W 301	96	126
T+W 400	36	66

It must be mention that the aircraft production rate does not indicate the total number of aircraft in the network. The GDM calculates the total possible number of aircraft that could potentially be part of the network based on these production rates. This applies to both N+1 and NASA's N+2 aircraft. The GDM combines the total number of aircraft that can be produced and the allowed number of flight-hours per aircraft (3,360 hours/year).

The available flight-hours decreases as the model start to assign flights to aircraft. If the available flight-hours reaches zero, then all the aircraft that could be produced in that particular year are used. If the flight-hours does not reach zero, then the GDM introduce into the network as many aircraft as aircraft needed to satisfy the forecast demand but not as many aircraft as aircraft that can be produced.

3 GDM Model Results

The GDM predicts a 5.14% annual growth in the number of commercial airline seats worldwide based on population and socio-economic data that were analyzed. Figure 6 shows the total number of seats that the Fratar model distribute in the worldwide network each year. The Fratar model distributes a total demand of 4.7 billion seats in 2016 and by 2040 it distributes 10.8 billion seats. Figure 7 presents the number of commercial flights worldwide to satisfy the future demand for commercial aviation. The model predicts an increase in annual flights from 37.4 million in 2016 to 83.8 million in year 2040. Figure 8 shows that around 17,600 aircraft by 2016 and around 39,000 aircraft are needed to satisfy the future demand for commercial aviation.

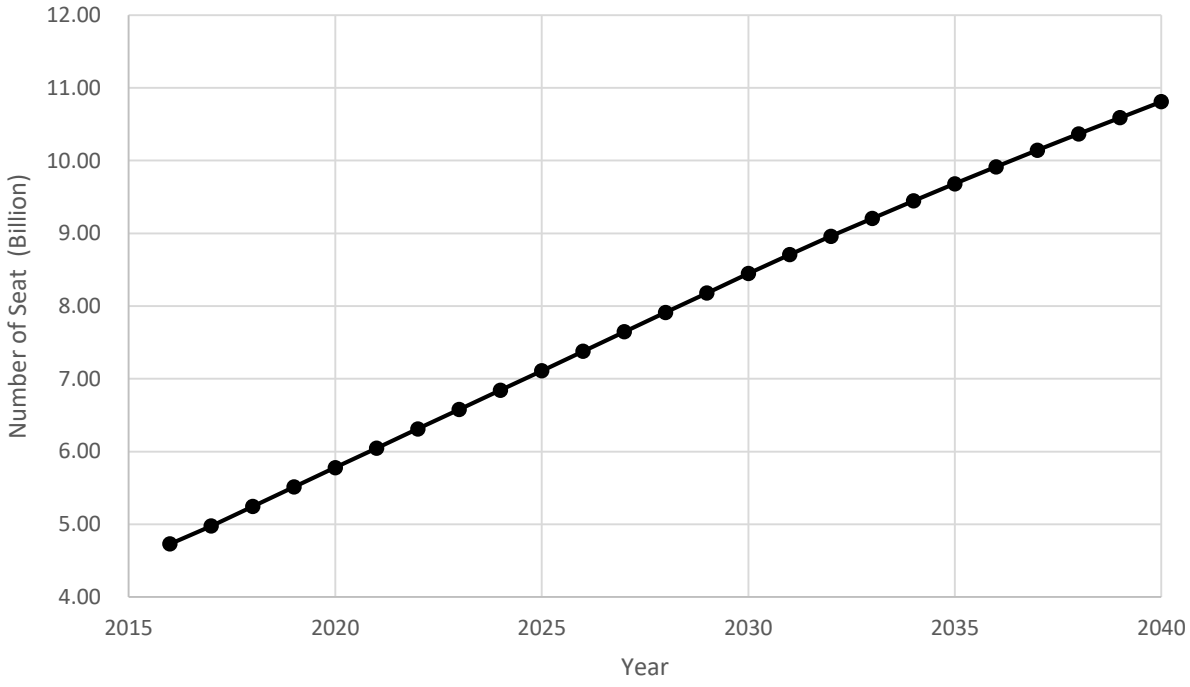


Figure 6: Total Number of Seats Distributed by the Fratar Model.

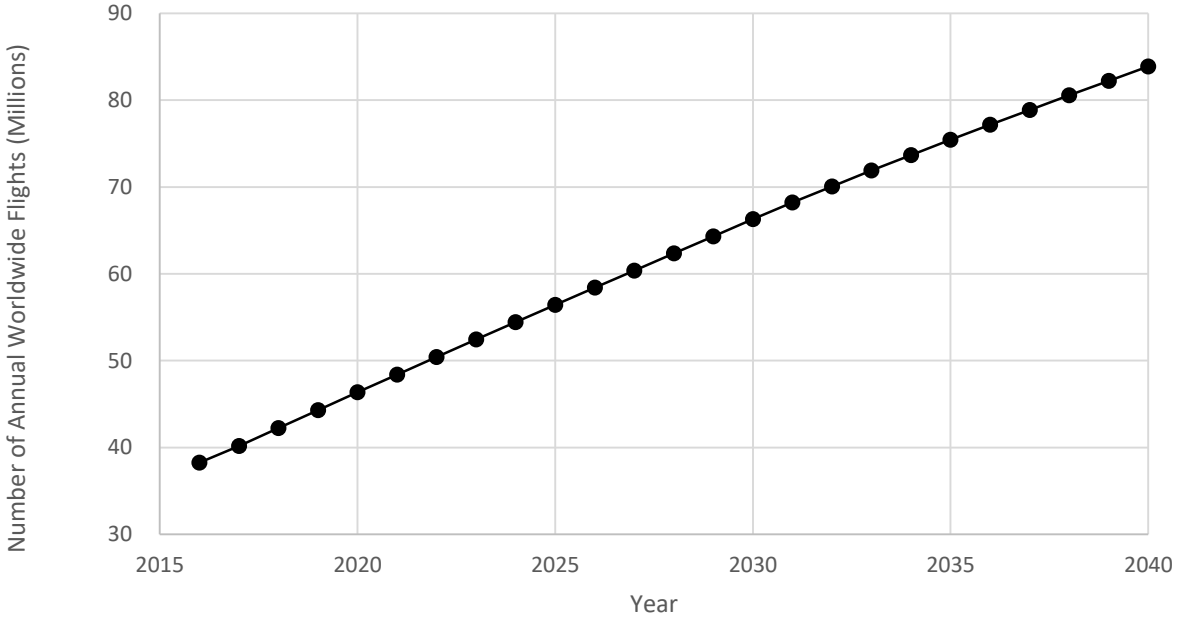


Figure 7: Estimated Total Number of Commercial Flights Worldwide.

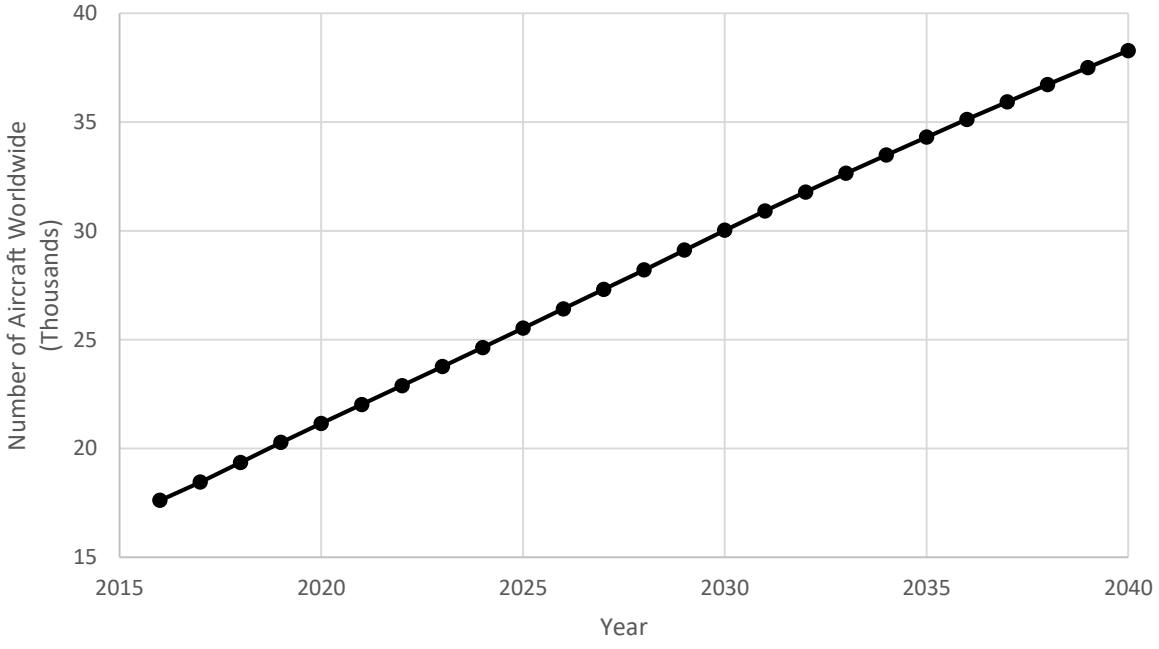


Figure 8: Estimated Total Number of Commercial Aircraft Worldwide.

3.1 Results for Scenario 1

Figure 9 and Figure 10 presents the 10 most used aircraft worldwide based on the calculation of scenario 1. The two most used aircraft were the Airbus 320 and the Boeing 737-800. The GDM indicates that 3,144 Airbus 320 and 3,101 Boeing 737-800 are needed to satisfy the demand assigned to these aircraft in the year 2016. For year 2040 the number of aircraft needed to satisfy the demand assigned to the Airbus 320 and the Boeing 737-800 is 7,594 aircraft and 7,422 aircraft respectively. In terms of the number of aircraft, the Airbus 320 and the Boeing 737-800 are followed by Airbus 319, Airbus 321, Boeing 737-700, Boeing 777-300, among others.

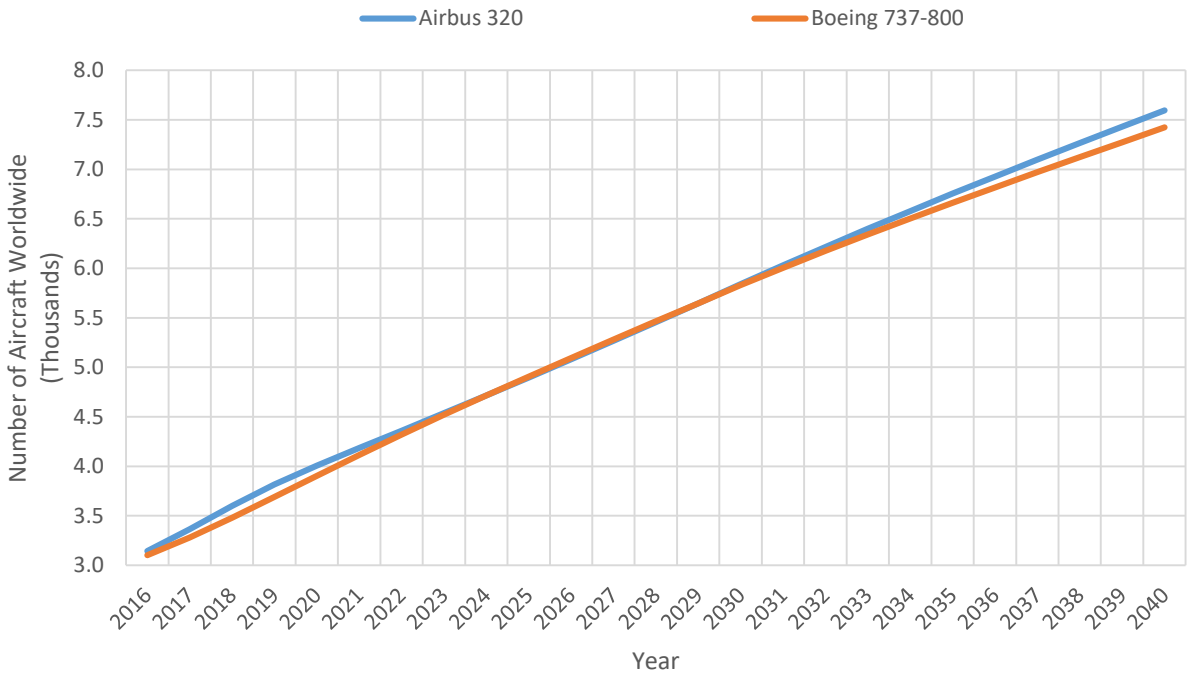


Figure 9: Estimated Number of Aircraft 2016 – 2040 for Scenario1.

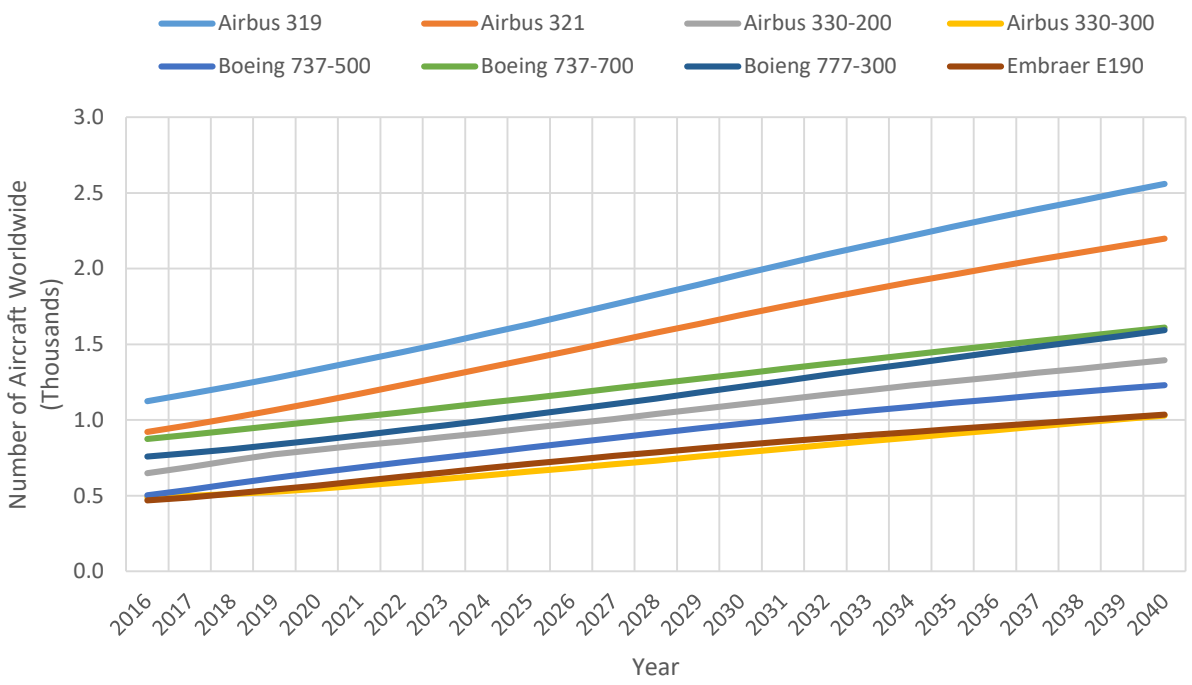


Figure 10: Estimated Number of Aircraft 2016 – 2040 for Scenario 1.

3.2 Results for Scenario 1.5

Figure 11 and Figure 12 presents the 10 most used N+1 aircraft worldwide. The different start point of each line indicates the year of introduction of a particular aircraft into the network. Airbus A320 and Boeing 737-800 were replaced by the two most used aircraft in scenario 1.5, Airbus 320neo and Boeing 737-8MAX respectively. In terms of the number of aircraft, the Airbus 320neo and the Boeing 737-8MAX are followed by Boeing 737-7MAX, Airbus 330neo, Airbus 321neo, Airbus 319neo, Boeing 737-9MAX, among others.

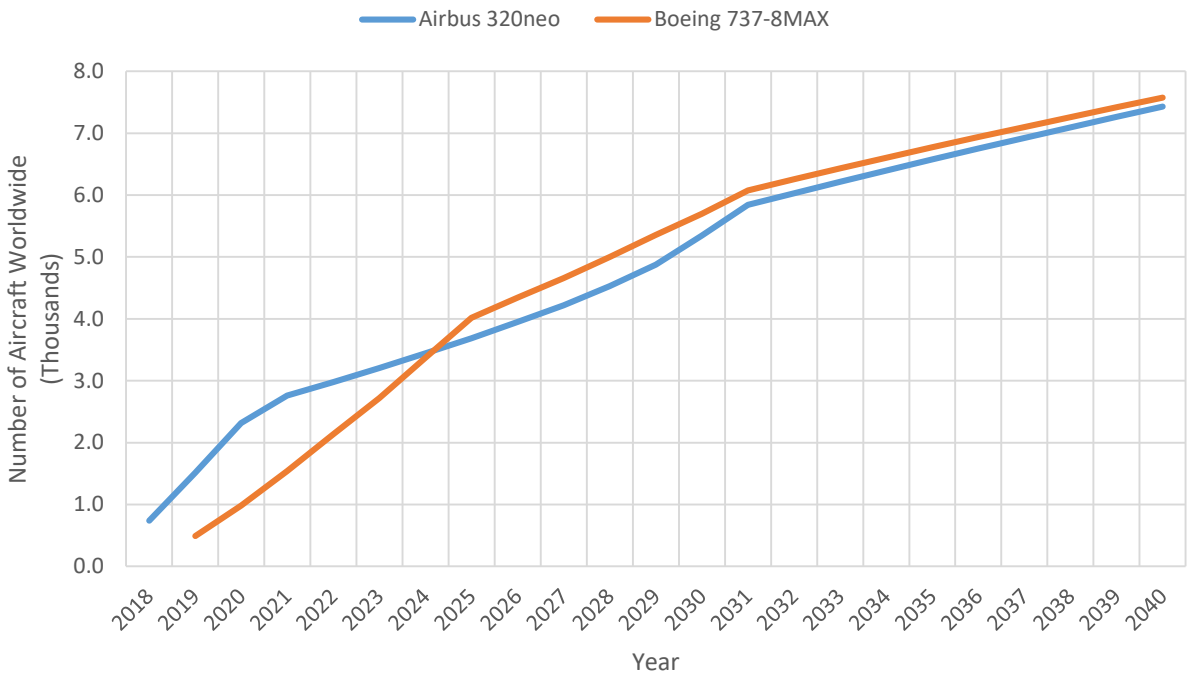


Figure 11: Estimated Number of Aircraft 2016 – 2040 for Scenario 1.5.

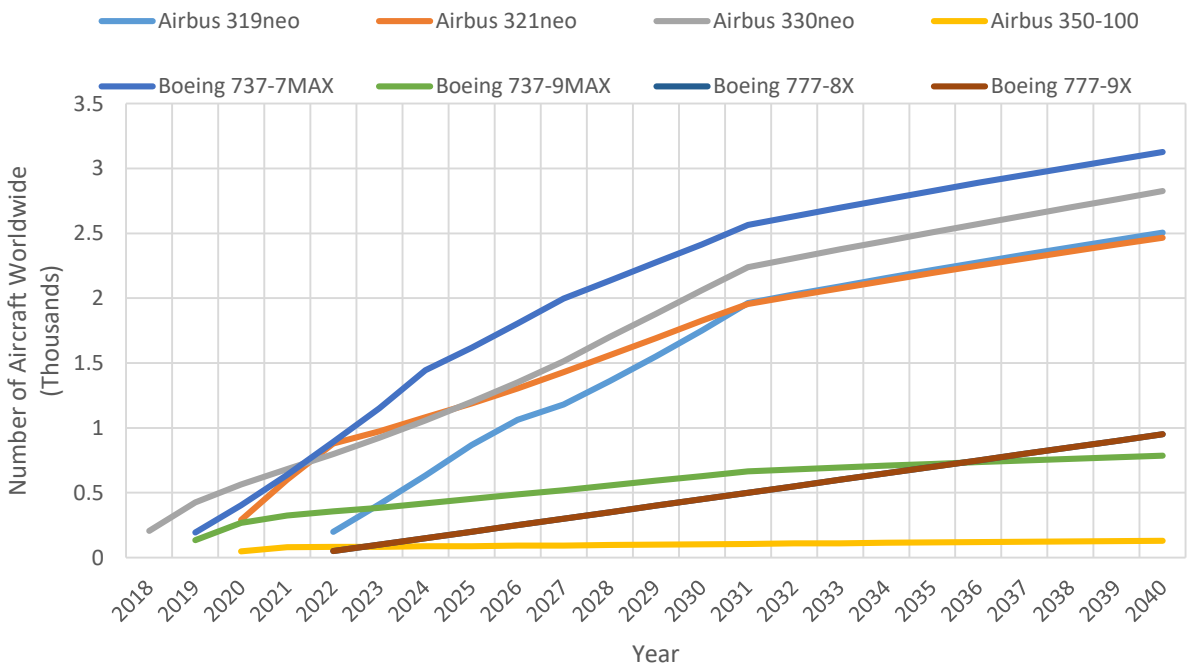


Figure 12: Estimated Number of Aircraft 2016 – 2040 for Scenario 1.5.

3.3 Results for Scenario 2 and 3

Figure 13 shows the number of Airbus A330neo in scenario 1.5 vs. scenario 3. As a reminder, scenario 3 is a case where large amounts of NASA's N+2 aircraft are used. It is clear how after year 2030, which is the introduction year for NASA's N+2 aircraft the number of Airbus A330neo decrease significantly. After 2030 more demand is assigned to NASA's N+2 aircraft and less demand is assigned to N+1 aircraft. Figure 14 shows the number of three N+1 aircraft for scenario 3 with a wide adoption of NASA's N+2 aircraft. It is evident that once NASA's N+2 aircraft are introduced into the network in the year 2030, the number of N+1 aircraft stops increasing. The three N+1 aircraft presented in Figure 14 are the Airbus 319neo, Airbus 330neo and Boeing 737-7MAX.

Figure 15 shows the number of NASA's N+2 aircraft for Scenario3. The figure shows the estimated number of T+W 98, T+W 160, and T+W 216 aircraft if aircraft are produced in large numbers (see Table 11, scenario 3). Figure 16 shows the number of NASA's N+2 aircraft (T+W 301 and T+W 400) according to scenario 3. From the results, it can be deduced that a high demand is being absorbed by the NASA's N+2 T+W 301 and the T+W 400. This is assumed since for this case the model is using as many aircraft as aircraft being produced.

Figure 17 illustrates how the aircraft fleet size from NASA's N+2 aircraft compares to the size of global aircraft fleet. If NASA's N+2 aircraft are first introduced into the network in the year 2030 and the standard production rate is used (see Table 11, scenario 2), NASA's N+2 aircraft could represent up to 12.3% of the total aircraft fleet worldwide. However, if a high production rate for these aircraft (see Table 11, scenario 3) is

adopted it could produce an end state with 31% of the global fleet in 2040 comprised of NASA's N+2 aircraft.

Figure 18 shows a comparison between the total number of flights assigned to NASA's N+2 aircraft in scenario 2 and scenario 3. The high production rate used for scenario 3 allows the GDM to assign NASA's N+2 aircraft three times the number of flights that were assigned in scenario 2. Figure 19 shows a comparison between the total number of NASA's N+2 aircraft in the network for scenario 2 and scenario 3. The high production rate for NASA's N+2 aircraft in scenario 3 increases the total number of aircraft in the network by a factor of 2 in 2030 and by a factor of 3 in the year 2040.

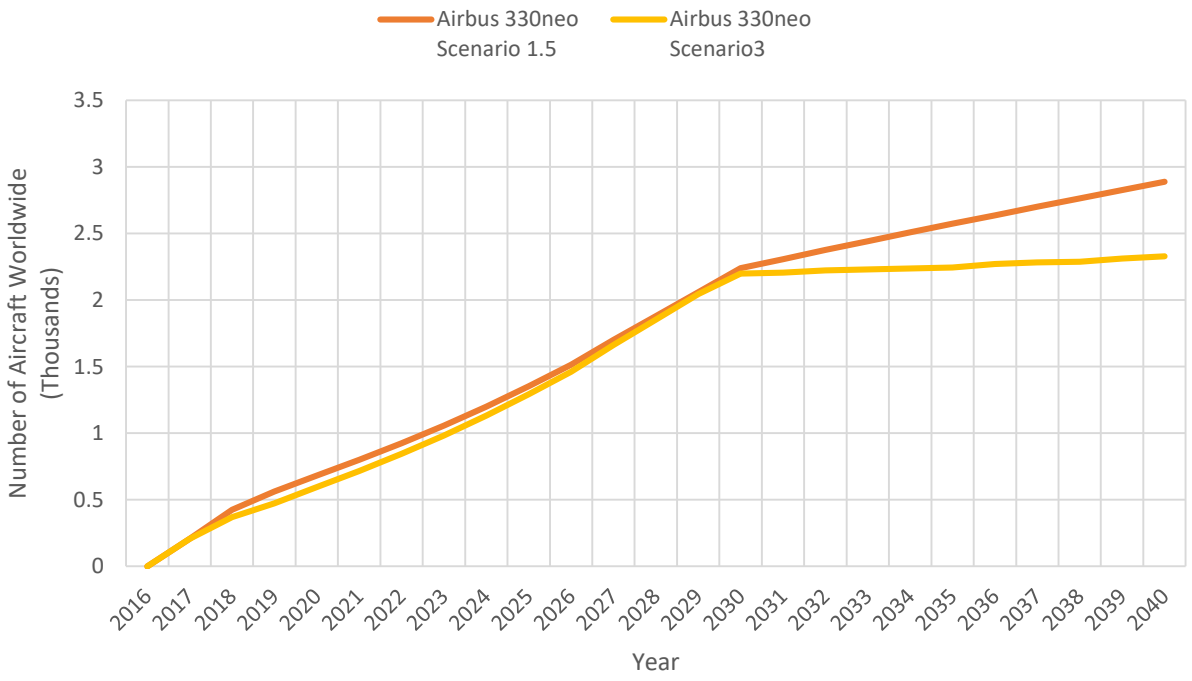


Figure 13: Estimated Number of Airbus A330neo Aircraft between 2016 – 2040. Comparison of Scenarios 1.5 and 3.

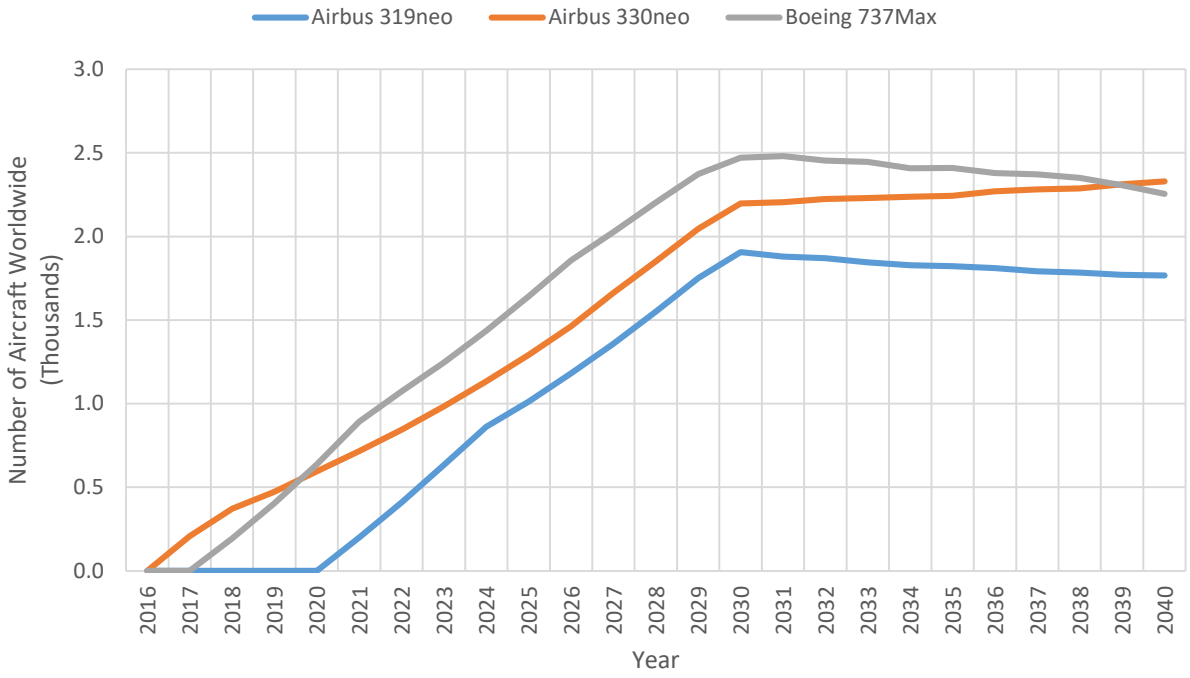


Figure 14: Number of Aircraft 2016 – 2040 for Selected N+1 Aircraft for Scenario 3 (with Wide Use of N+2 Aircraft).

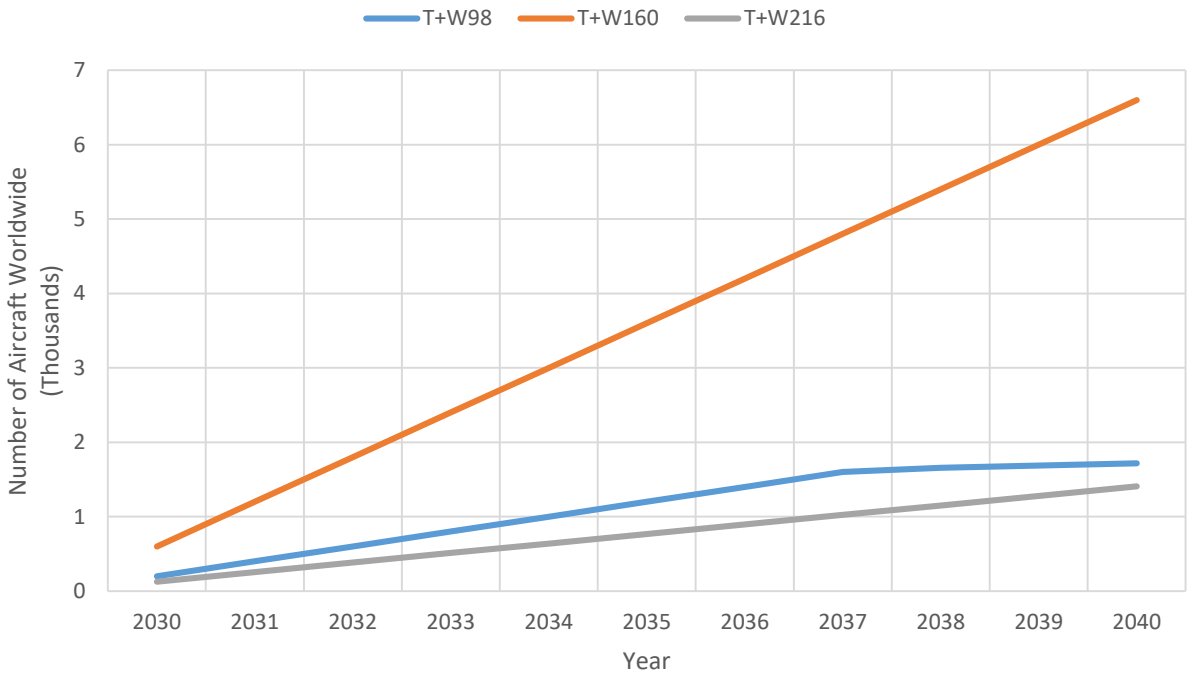


Figure 15: Estimated Number of NASA's N+2 Aircraft 2030 – 2040 for Scenario 3.

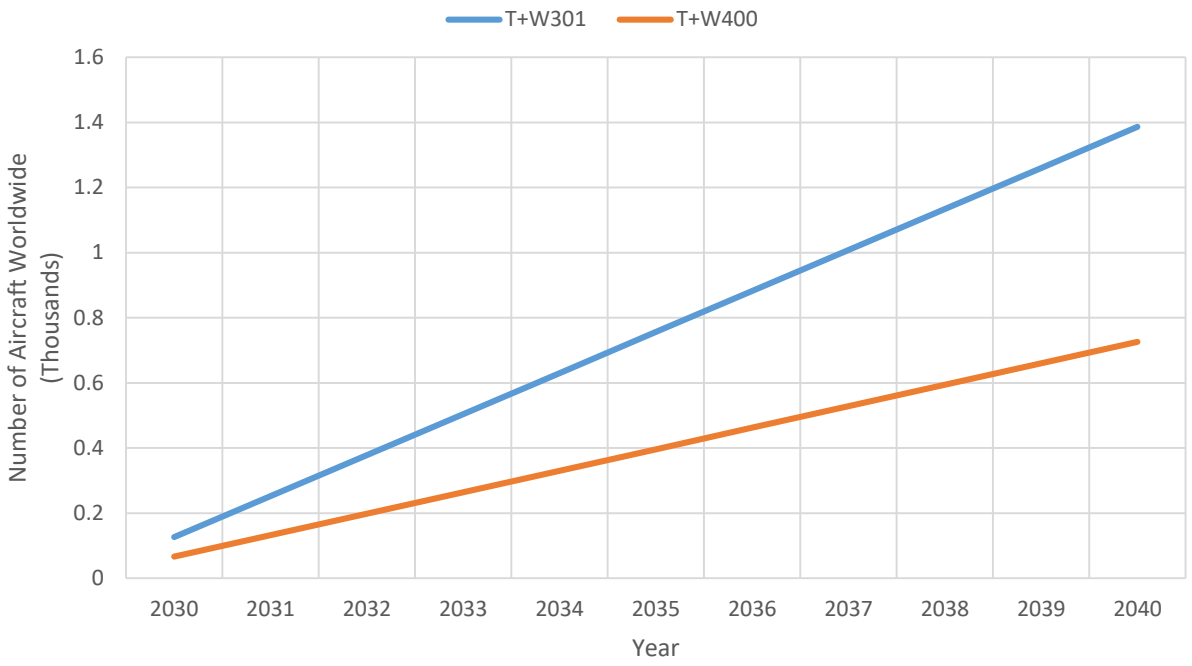


Figure 16: Estimated Number of NASA's N+2 Aircraft 2030-2040 for Scenario 3.

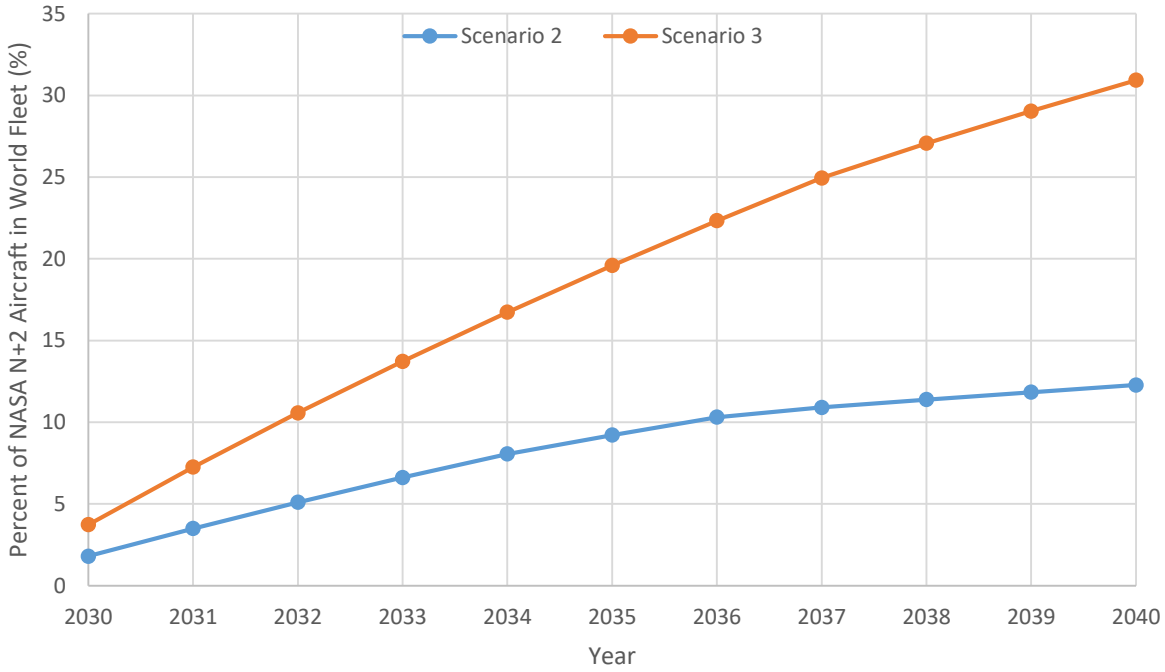


Figure 17: NASA's N+2 Aircraft in Comparison to the Worldwide Fleet (%) for Scenario 2 and 3.

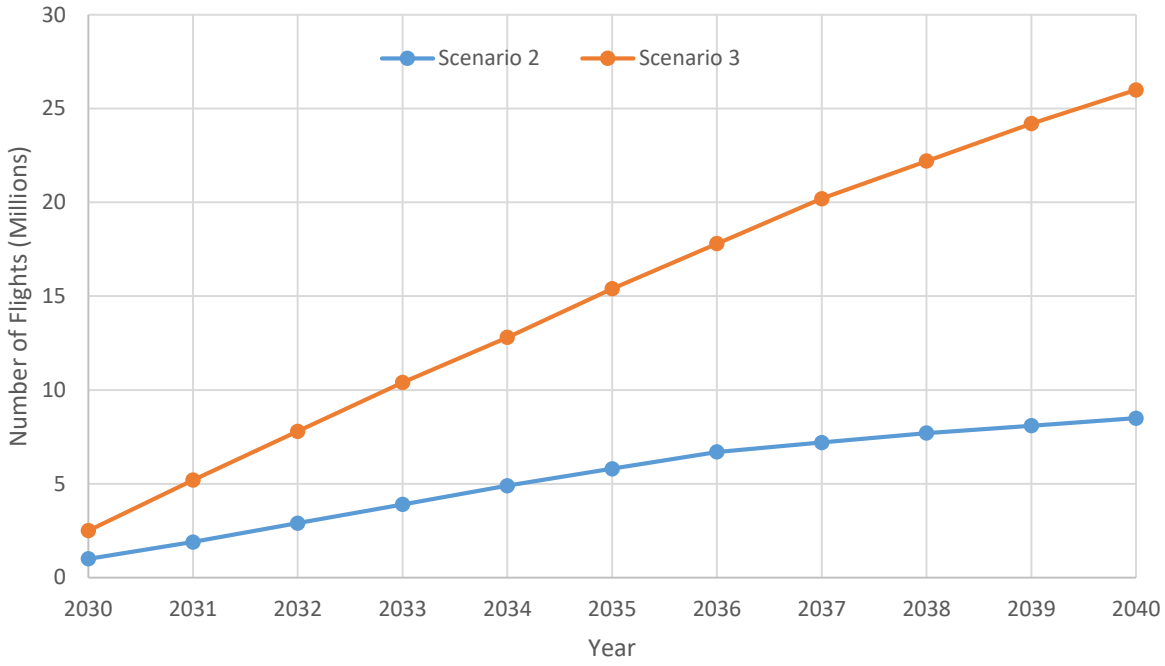


Figure 18: Annual Flights by NASA's N+2 Aircraft for Scenario 2 and 3.

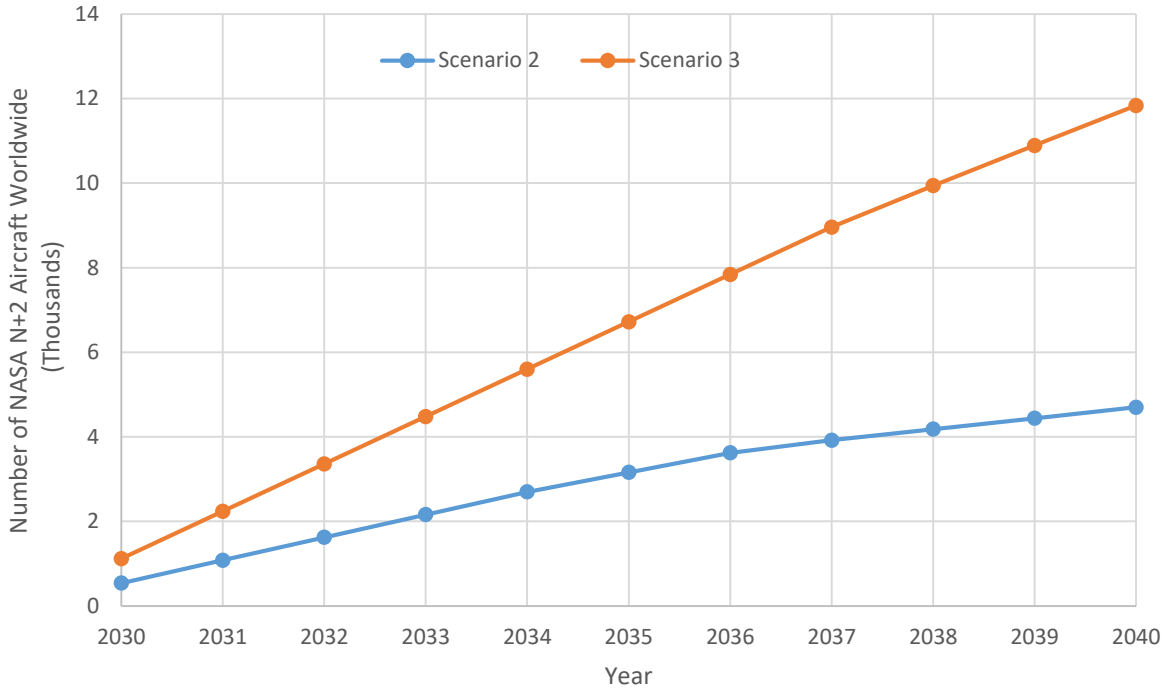


Figure 19: Total Number of NASA's N+2 Aircraft for Scenario 2 and 3.

4 Conclusions

This research project described enhancements to the Global Demand Model (GDM). The improvements made address the need to predict: a) number of seats (surrogate of demand) by origin-destination, b) worldwide aircraft fleet mix distribution by origin-destination pair, c) number of flights worldwide by origin-destination pair, d) aircraft fleet evolution over time, d) estimate of number of each aircraft type by origin-destination pair. The model results can be summarized as follows:

- Africa region has a growth of over 145%; with an increase in the number of flights to over 3.5 million.
- Asia region has a growth of over 170%; with an increase in the number of flights to over 28 million.
- Middle East region has the highest growth of over 200%; with an increase in the number of flights to over 3.5 million.
- Europe region has a growth of 119% with an increase in the number of flights to over 19 million.
- North America region has the least growth of 545; with an increase in the number of flights to over 17 million.
- Latin America region has a growth of 148%; with an increase in the number of flights to over 9 million.
- Oceania region has a growth of 69%; with an increase in the number of flights to over 2 million.

As for the total number of aircraft worldwide, it was mentioned that around 39,000 aircraft would be required to satisfy the global commercial aviation demand in the year 2016 (see Figure 7). At the end of the model part of the results are back engineered to estimate the number of aircraft in the network. The number of aircraft is estimated using the annual aircraft utilization of 3,630 hours. The average utilization is obtained from the MIT Airline Project (http://web.mit.edu/airlinedata/www/default.html) and is applicable to single-aisle aircraft in the US. The utilization of aircraft in other parts of the world is known to be lower than in the US and hence the fleet estimates in the GMD are lower than forecast by Airbus and Boeing. According to Boeing - Current Market Outlook 2017-2036, they predict a worldwide aircraft fleet size of 41,030 aircraft in the year 2036. The Airbus – Growing Horizons Global Market forecast 2017-2036 indicates that the worldwide aircraft fleet in the year 2016 and in the year 2036 is 18,890 aircraft and 40,120 aircraft respectively. The GDM estimates a worldwide aircraft fleet size of 17,616 aircraft by 2016; 7% lower than the Airbus prediction for the same year. The GDM estimates a worldwide aircraft fleet size of 35,119 aircraft by 2036, 14% and 16% lower than the prediction of Airbus and Boeing for the same year, respectively.

Depending on the aircraft type and size it would be feasible for an aircraft to fly more than 3,630 hours per year or even fewer hours. That would alter the number of aircraft in the fleet. Moreover, the GDM model does not include cargo aircraft at this time. The travel time estimates in the GDM model do not include airline “padding” (time added to account for re-current delays at airports). For these reasons, the fleet number estimates in the GDM tend to be lower than industry estimates.

5 Recommendations

The model described in this report can be improved in the following areas:

a) Include of origin-destination demand for cargo services. This step requires information about cargo operations worldwide that were not available during the study. Adding cargo service flights into the network would increase the size of the worldwide aircraft fleet.

b) Modeling more complex feedback effects of airline fleet renewal, airline fares, and aviation demand. The GDM model includes a demand elasticity factor to predict induces commercial air transportation demand if fares are reduced. The introduction of NASA's N+2 could, in theory, reduce airfares if the fuel savings associated with the operation of more fuel-efficient aircraft are passed on to consumers. If a high enough demand is assigned to NASA's N+2 aircraft, it could potentially lead to an indirect price drop in airfares. This could potentially increase the air travel demand for routes using NASA's N+2 aircraft and hence increasing the number of flights and aircraft fleet size. This feedback loop was not studied in this analysis.

c) Adopt Official Airline Guide published travel time data which includes padding times. This step will improve the estimates of the aircraft fleet worldwide. Airlines add travel time to flights to account for random schedule effects and to account for re-current airport delays. These additional travel times will require a larger global aircraft fleet to service the demand estimated in the GDM model.

d) Create a list containing allowed annual flight-hours for each individual aircraft. The number of hours that aircraft can perform annually varies by aircraft type and aircraft category. The fixed 3,630 hours allowed flight-time per year per aircraft should be

variable according to the aircraft type. This step will improve the estimates of the aircraft fleet worldwide.

e) Incorporate a cost analysis of the aircraft retirement process.

References

1) Airbus – Growing Horizons Global Market forecast 2017-2036. http://www.aircraft.airbus.com/market/global-market-forecast-2017-2036/
2) Alsalous, O., Global Demand Model, MS Thesis, Virginia Tech, Fall 2015.
3) Boeing - Current Market Outlook 2017-2036. www.boeing.com/cmo
4) Boyce, D. E. and H. C. W. L. Williams (2015). Forecasting Urban Travel: Past, Present and Future. Cheltenham, UNKNOWN, Edward Elgar Publishing.
5) BuchAir Aircraft Database, FlightGlobal, 2013
6) Ceha, R. and H. Ohta (1997). "Prediction of future origin destination matrix of air passengers by fratar and gravity models." Computers and Industrial Engineering 33(3-4): 845-848.
7) Grosche, T., et al. (2007). "Gravity models for airline passenger volume estimation." Journal of Air Transport Management 13(4): 175-183.
8) Heanue, K. E. and C. E. Pyers (1966). "Comparative evaluation of trip distribution procedures." Public Roads 34(2): 43-51.
9) Holmberg, K. and K. Jörnsten (1989). "Exact Methods for Gravity Trip-Distribution Models." Environment and Planning A 21(1): 81-97.
10) Nickol, C., Haller, W., “Assessment of the Performance Potential of Advanced Subsonic Transport Concepts for NASA’s Environmental Responsible Aviation Project”, 2016
11) Official Airline Guide, years 2000, 2005, 2010, and 2015
12) Tillema, F., et al. (2006). "Comparison of neural networks and gravity models in trip distribution." Computer-Aided Civil and Infrastructure Engineering 21(2): 104-119.

13) Viken, J., et al. (2006). Utilizing traveler demand modeling to predict future commercial flight schedules in the NAS. 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, September 6, 2006 - September 8, 2006, Portsmouth, VA, United states, American Institute of Aeronautics and Astronautics Inc.

14) www.flightaware.com

Appendix A: Flowchart

Figure 20 presetns a flowchart of the GDM Model. Figure 21 presents a summarized section of the GDM flowchart that waspart of this project.

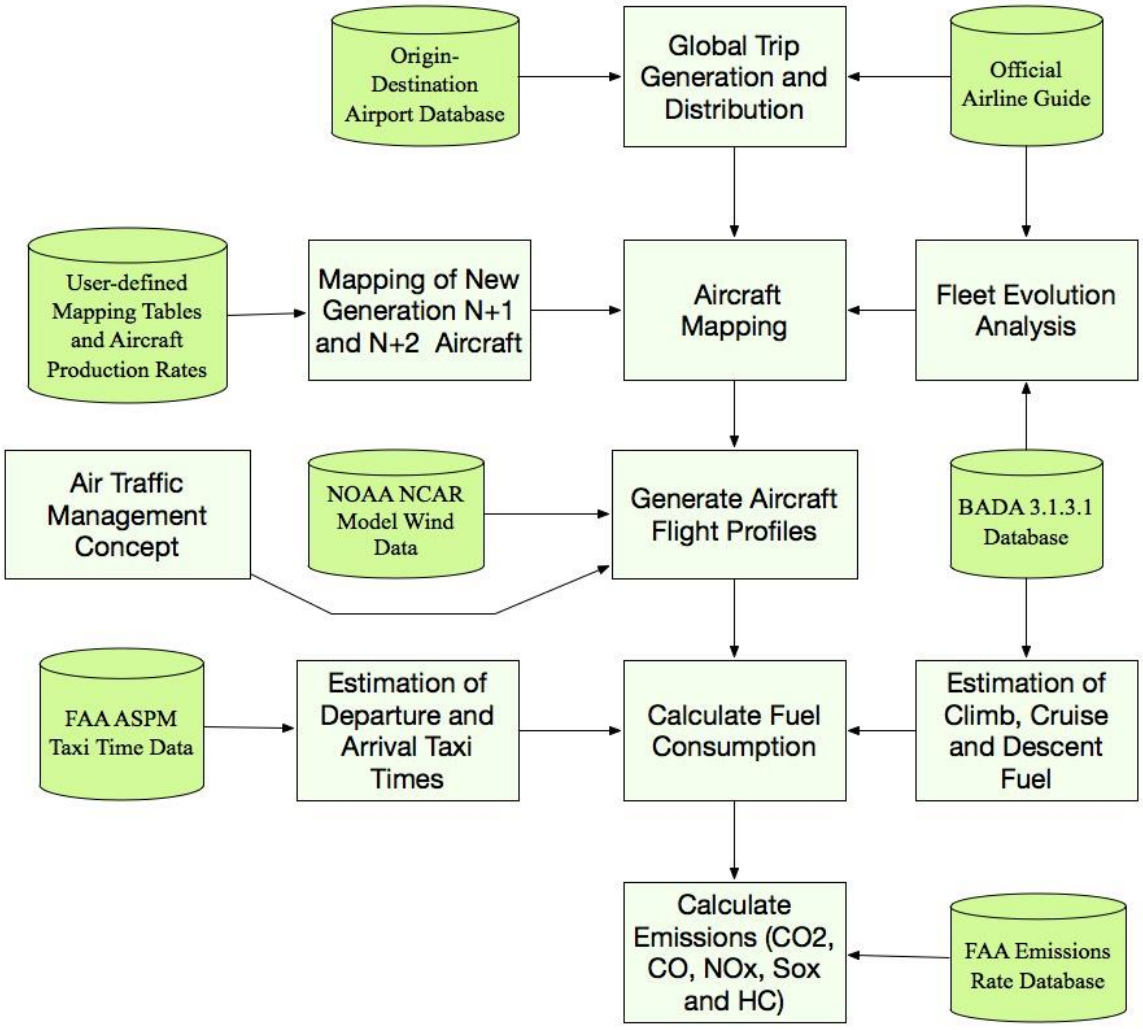


Figure 20: Flowchart of the GDM Model

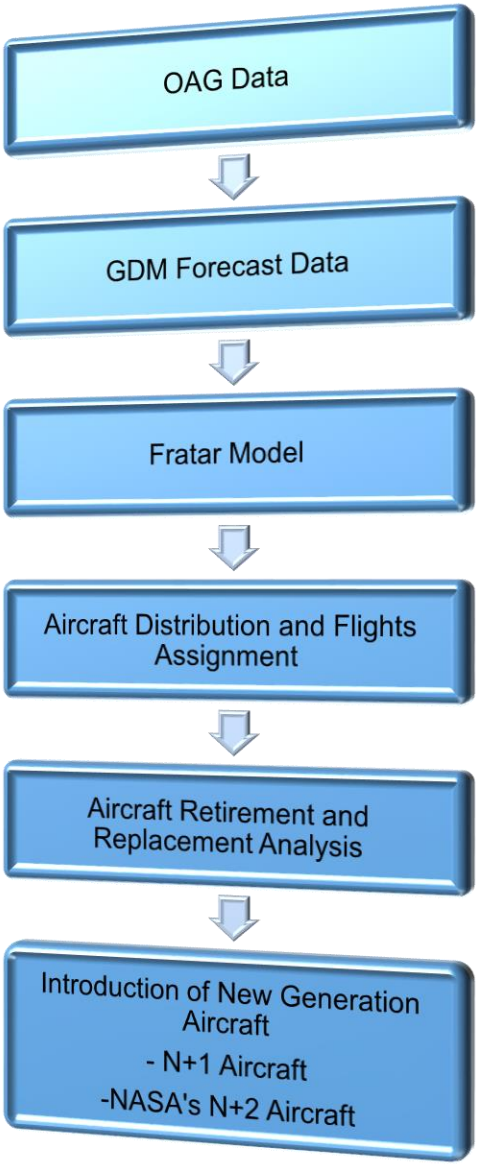


Figure 21: Flowchart Section of the GDM Model.

Appendix B: Source Code

The source codes that were improved or created over the past year and a half for the development and enhancement of the GDM is included bellow. The relevant code includes; OAG_main.m, OAG_Read.m, Import_OAG_Data.m, create_aircraft_list.m, OAG_Analysis.m, Reduction_AcftData_OAG.m, FRATAR_Model_Main_File.m, Forecast_Data_Table_Generator.m, Fratar_Model.m, Aircraft_and_Flights_Assignment.m, Fleet_Evolution.m, Retirment_Replacement_Analysis.m, Introduction_of_future_aricraft.m, Fleet_Evolution_Validation.m

OAG_main.m

```
% This script executes all OAG data analysis scripts

clc;

clear all;

close all;


%OAG_Data_Dir = 'D:\Datasets\OAG';

OAG_Data_Dir = 'C:\GDM\OAG';

Flight_Type = 'OPERATING';


Start_Year = 1996;

End_Year = 2016;


%% 1: Import the data by year


Read_OAG(OAG_Data_Dir, Flight_Type, Start_Year, End_Year); % This script loops through the years and calls Import_OAG_Data function

OAG_Parts = 3;


%% 2: Find OAG Totals by year by airport


OAG_Totals_zero_stops(start_year, end_year, Parts);


create_airport_list(end_year);


%% 3: OAG Airline analysis


OAG_Airlines_Analysis(start_year, end_year, Parts); % Find OAG total seats per airport per carrier by year


%% 4: Herfindahl index calculation


Herfindahl_index_calculation(start_year, end_year);


%% 5: Create OAG Summaries: Reformatting OAG data by airport
```

```
OAG_Summary(start_year, end_year);
OAG_Summary_Herfindahl(start_year, end_year);

%% 6: Generate aircraft list by year

create_aircraft_list(end_year);

%% 7: OD PAir Data for Trip Distrbution

%OAG_Analysis(Parts);
OAG_Analysis_All_Years(OAG_Parts, Start_Year, End_Year);

%% 8: Reduction_AcftData_OAG_2015(Install_Dir, Save_Dir);

Reduction_AcftData_OAG();

% **** END OF FILE **** %
```

OAG_Read.m

```
% Read and save OAG shcedule data for each year divided into 2 parts (to reduce file size)
function Parts = Read_OAG(OAG_Data_Dir, Flight_Type, Start_Year, End_Year)

Parts = 3;
Number_Of_Months = 12;
Number_Of_Months_Per_Part = Number_Of_Months / Parts;

Months = 0:Number_Of_Months_Per_Part:Number_Of_Months;

for Year = Start_Year : End_Year

   disp(['Year: ', num2str(Year), '/', num2str(End_Year)]);

   for Part = 1:Parts

      disp(['  Part: ', num2str(Part), '/', num2str(Parts)]);

      Start_Month = Months(Part) + 1;
      End_Month   = Months(Part) + Number_Of_Months_Per_Part;

      Counter = 0;
      for Month = Start_Month : End_Month
         disp(['    Month: ', num2str(Month), '/', num2str(End_Month)]);
         Counter = Counter + 1;

         OAG_Data(Counter) = Import_OAG_Data(OAG_Data_Dir, Year, Month, Flight_Type);
      end % for Month = Start_Month : End_Month

      % Serializing the struct to be able to save it compressed format.
      % Otherwise file size is huge. To Deserialize, use
      % getArrayFromByteStream() after loading.
      OAG_Data_Ser = getByteStreamFromArray(OAG_Data); clear OAG_Data;
      save ((['.\Output\OAG_Data_' , num2str(Year) , '_Part' , num2str(Part), '.mat']) , 'OAG_Data_Ser', '-v7.3');
```

```
        clear OAG_Data_Ser

    end % for Part = 1:Parts

end % for Year = Start_year : End_year

return;
```

Import_OAG_Data.m

```
% This m-file reads the OAG Schedule file and returns s truct array


%IMPORTANT NOTE:

%This function reads OAG files when each month is stored in

%a directory named as the example below

%E:\Global Demand Model\OAG data\Year_2005\OAG_2005_12_OPER

%It works this way since the OAG website only names the .ZIP files correctly and generates a numeric value for the CSV file.

%Therefore the code looks for the folder with the correct name then reads

%the *.CSV file inside it (There should be only 1 file there anyway)


%Example OAG_Data_Dir='E:\Global Demand Model\OAG data'


%******************************%

%Flight_Type='OPER' or 'NON_' %

%******************************%


function OAG_Schedule = Import_OAG_Data(OAG_Data_Dir, Year, Month, Flight_Type)


% disp('Importing OAG Schedule...');


% ------------

% Read OAG Schedule File

% ------------


% ------------

% Locate File

% ------------

% data_folder = [OAG_Data_Dir , '\Year_' , num2str(Year)]; %change directory

% directory = dir(data_folder);


%----------

%determine the length of the file name to be checked based on the

%number of digits o the month
```

```
% if Month < 10
%     end_filename_string = 15; %Number of characters in the filename string
% else
%     end_filename_string = 16; %Number of characters in the filename string
% end
% %-------------
%
% for file = 1: length(directory)
%     if directory(file).isdir == 1
%         if length(directory(file).name) >= end_filename_string %Check file name length
%             if strcmp(directory(file).name(5:end_filename_string),([ num2str(Year) ,'_' , num2str(Month) , '_' , Flight_Type])) %Compare the file name to the desired year, month, and flight type
%                 OAG_File_Dir = ([OAG_Data_Dir, '\Year_', num2str(Year), '\' , directory(file).name ]);
%             end
%         end
%     end
% end
% %OAG_File_Dir = ([OAG_Data_Dir, '\Year_', num2str(Year), '\OAG_', num2str(Year), '_', num2str(Month), '_', Flight_Type]);
% %cd(OAG_File_Dir); %change directory to the file path
% file_ext = '*.csv'; %File extension
% file_name = dir([OAG_File_Dir, file_ext]); %find files with the CSV extension in the current folder
%
% OAG_Filepath = ([OAG_File_Dir, '\', file_name.name]);


OAG_Filepath = ([OAG_Data_Dir, '\Y',  num2str(Year), '\OAG_', num2str(Year), '_', num2str(Month), '_', Flight_Type, '.csv']);


fid = fopen(OAG_Filepath, 'r');
FieldNames = regexp(fgetl(fid), ',', 'split');


% Field Names:
% Field 1:  Carrier1
% Field 2:  Carrier1Name
% Field 3:  FlightNo1
% Field 4:  Carrier1Alliance
```

% Field 5: DepAirport

% Field 6: DepState

% Field 7: DepIATACtry

% Field 8: DepReg

% Field 9: DepRegName

% Field 10: DepWACCountry

% Field 11: ArrAirport

% Field 12: ArrState

% Field 13: ArrIATACtry

% Field 14: ArrReg

% Field 15: ArrRegName

% Field 16: ArrWACCountry

% Field 17: LocalDepTime

% Field 18: LocalArrTime

% Field 19: InternationalDomestic

% Field 20: Service

% Field 21: Seats

% Field 22: FstSeats

% Field 23: BusSeats

% Field 24: EcoSeats

% Field 25: EffFrom

% Field 26: EffTo

% Field 27: ElapsedTime

% Field 28: FlyingTime

% Field 29: GroundTime

% Field 30: Stops

% Field 31: SpecificAcft

% Field 32: SpecificAcftName

% Field 33: EquipmentGroup

% Field 34: Routing

% Field 35: DistStMiles

% Field 36: RangeNautMiles

% Field 37: CruiseSpeed

% Field 38: MaxTakeOffWeight

% Field 39: HoldVolume

```
% Field 40: LocalDaysOfOp1
% Field 41: LocalDaysOfOp2
% Field 42: LocalDaysOfOp3
% Field 43: LocalDaysOfOp4
% Field 44: LocalDaysOfOp5
% Field 45: LocalDaysOfOp6
% Field 46: LocalDaysOfOp7
% Field 47: DupMarker
% Field 48: DupCar1
% Field 49: DupCar2
% Field 50: DupCar3
% Field 51: DupCar4
% Field 52: DupCar5
% Field 53: DupCar6
% Field 54: DupCar7
% Field 55: DupCar8
% Field 56: OpCar
% Field 57: FltDup
% Field 58: Frequency
% Field 59: ASMs
% Field 60: Timeseries

Imported_Fields = [1 2 3 5 7 8 11 13 14 19 21 22 23 24 27 28 30 31 35 58]; % matching the imported columns by
textscan

Format = '';

for f = 1:60
  if any(Imported_Fields == f) % Field to keep
     Format = [Format, '%q'];
  else
     Format = [Format, '%*q']; % Field to skip
  end
end

Output = textscan(fid, Format, 'delimiter', ',');
```

```
fclose(fid);

% ------------
% Create Struct File
% ------------
Number_Of_Fields = length(Output);

OAG_Schedule = struct;

for f = 1:Number_Of_Fields

    CurrentFieldName = FieldNames{Imported_Fields(f)};
    OAG_Schedule.(CurrentFieldName) = Output{f};

end % for f = 1:Number_Of_Fields

% ------------
% Convert to float
% ------------
OAG_Schedule.Frequency = str2double(OAG_Schedule.Frequency);
OAG_Schedule.Seats = str2double(OAG_Schedule.Seats);
OAG_Schedule.DistStMiles = str2double(OAG_Schedule.DistStMiles);
OAG_Schedule.Stops = str2double(OAG_Schedule.Stops);
OAG_Schedule.FstSeats = str2double(OAG_Schedule.FstSeats);
OAG_Schedule.BusSeats = str2double(OAG_Schedule.BusSeats);
OAG_Schedule.EcoSeats = str2double(OAG_Schedule.EcoSeats);
% ------------
% Go back to the previous directory
% ------------
% cd(old_folder)

clearvars Output

return;
```

create_aircraft_list.m

```
% This function creates the aircraft list for trip distribution
function create_aircraft_list(year)

% year = 2015;

Start_Date = ['1/1/', num2str(year)];
End_Date   = ['12/31/', num2str(year)];

Start_Date_Num = datenum(Start_Date, 'mm/dd/yyyy');
End_Date_Num   = datenum(End_Date, 'mm/dd/yyyy');

% Load OAG Aircraft List
[~,~,raw] = xlsread('..\OAG_Passenger_Processing\Input\IATA_OAG_aircraft_Jan_2017.xlsx');

% Extract data
IATA         = raw(2:end,1);
Manufacturer = raw(2:end,2);
Acft_Name    = raw(2:end,3);
Cat_Name     = raw(2:end,4);
Eff_From     = raw(2:end,5);
Eff_To       = raw(2:end,6);
BADA_Type    = raw(2:end,7);

Eff_From_Num = datenum(Eff_From, 'mm/dd/yyyy');
Eff_To_Num   = datenum(Eff_To, 'mm/dd/yyyy');

% Save relevant data
Number_Of_Aircraft = length(IATA);

Counter = 0;

for acft = 1:Number_Of_Aircraft
```

```
    % Pick only aircraft that have an effictive date for the year
    if Eff_From_Num(acft) > End_Date_Num
        continue;
    elseif Eff_To_Num(acft) < Start_Date_Num
        continue;
    end

    Counter = Counter + 1;

    aircraft_list(Counter).IATA         = IATA{acft};
    aircraft_list(Counter).Manufacturer = Manufacturer{acft};
    aircraft_list(Counter).Acft_Name    = Acft_Name{acft};
    aircraft_list(Counter).Cat_Name     = Cat_Name{acft};
    aircraft_list(Counter).Eff_From     = Eff_From{acft};
    aircraft_list(Counter).Eff_To       = Eff_To{acft};
    aircraft_list(Counter).Eff_From_Num = Eff_From_Num(acft);
    aircraft_list(Counter).Eff_To_Num   = Eff_To_Num(acft);
    aircraft_list(Counter).BADA_Type    = BADA_Type{acft};

end % for 1:Number_Of_Aircraft

% Check for duplicates
[Unique_Aircraft, ~, Unique_Aircraft_Index] = unique({aircraft_list.IATA});

% Pick latest Eff_To date
Aircraft_To_Keep = false(length(aircraft_list),1);

Number_Of_Unique_Aircraft = length(Unique_Aircraft);

for acft = 1:Number_Of_Unique_Aircraft

    Current_Aircraft_Indicies = find(Unique_Aircraft_Index == acft);

    % Skip if not duplicate
    if length(Current_Aircraft_Indicies) == 1
```

```
        Aircraft_To_Keep(Current_Aircraft_Indicies) = true;
        continue;
    end

    % Get latest effective to date
    Current_Eff_To_Num = [aircraft_list(Current_Aircraft_Indicies).Eff_To_Num];

    [~, Max_Index] = max(Current_Eff_To_Num);
    Aircraft_To_Keep(Current_Aircraft_Indicies(Max_Index)) = true;

end % for acft = 1:Number_Of_Unique_Aircraft

aircraft_list(Aircraft_To_Keep == false) = [];

% Saving
save ('.\Output\aircraft_list.mat', 'aircraft_list')

return;
```

OAG_Analysis.m

```
function OAG_Analysis(OAG_Parts)

%Fisrt, Run Read_OAG(OAG_Data_Dir,Flight_Type,start_year,end_year);
Number_Of_Months = 12;
Number_Of_Months_Per_Part = Number_Of_Months / OAG_Parts;

Months = 0:Number_Of_Months_Per_Part:Number_Of_Months;

OAG_Data2015.DepAirport   = [];
OAG_Data2015.ArrAirport   = [];
OAG_Data2015.Seats        = [];
OAG_Data2015.Stops        = [];
OAG_Data2015.SpecificAcft = [];
OAG_Data2015.Frequency    = [];

for Part = 1:OAG_Parts

    disp(['  Part: ', num2str(Part), '/', num2str(OAG_Parts)]);

    % Load OAG_Data
    load(['.\Output\OAG_Data_2015_Part', num2str(Part), '.mat'])
    OAG_Data = getArrayFromByteStream(OAG_Data_Ser);
    clear OAG_Data_Ser;

    Start_Month = Months(Part) + 1;
    End_Month   = Months(Part) + Number_Of_Months_Per_Part;

    % Filter the data for 0 stops only
    Counter = 0;
    for Month = Start_Month : End_Month

        disp(['    Month: ', num2str(Month), '/', num2str(End_Month)]);
        Counter = Counter + 1;
```

```
        Stops = [OAG_Data(Counter).Stops];

        Zero_Stops = Stops == 0;

        OAG_Data2015.DepAirport       = [OAG_Data2015.DepAirport;
[OAG_Data(Counter).DepAirport(Zero_Stops)]];
        OAG_Data2015.ArrAirport       = [OAG_Data2015.ArrAirport;       [OAG_Data(Counter).ArrAirport(Zero_Stops)]];
        OAG_Data2015.Seats            = [OAG_Data2015.Seats;            [OAG_Data(Counter).Seats(Zero_Stops)]];
        OAG_Data2015.Stops            = [OAG_Data2015.Stops;            [OAG_Data(Counter).Stops(Zero_Stops)]];
        OAG_Data2015.SpecificAcft     = [OAG_Data2015.SpecificAcft;
[OAG_Data(Counter).SpecificAcft(Zero_Stops)]];
        OAG_Data2015.Frequency        = [OAG_Data2015.Frequency;
[OAG_Data(Counter).Frequency(Zero_Stops)]];

    end % for Month = Start_Month : End_Month

    clear OAG_Data;

end % for Part = 1:OAG_Parts

%%
%Lines 114 to 140 Eliminate rows in OAG_Data_2015 that in the field of SpecificAcftName
%is not an aircraft. There were 5 helicopters and 2 others (Bus, Equipment
%Varies)

% Read OAG aircraft list to exclude certain types
load('.\Output\aircraft_list.mat');

[~, OAG_Data2015.SpecificAcftIndex] = ismember(OAG_Data2015.SpecificAcft, {aircraft_list.IATA});

% Find aircraft types to remove
Manufacturer = {'Other'};
Cat_Name     = {'Amphibious', 'Helicopter'};

[Manufacturer_Index_To_Be_Deleted, ~] = ismember({aircraft_list.Manufacturer}, Manufacturer);
[Cat_Name_Index_To_Be_Deleted, ~]     = ismember({aircraft_list.Cat_Name}, Cat_Name);
```

```
Index_To_Be_Deleted = find(Manufacturer_Index_To_Be_Deleted | Cat_Name_Index_To_Be_Deleted);

[Rows_to_be_deleted, ~] = ismember(OAG_Data2015.SpecificAcftIndex, Index_To_Be_Deleted);

disp('Deleting O-D pair with Non-aircraft vehicle')

OAG_Data2015.DepAirport(Rows_to_be_deleted)   = [];
OAG_Data2015.ArrAirport(Rows_to_be_deleted)   = [];
OAG_Data2015.Seats(Rows_to_be_deleted)        = [];
OAG_Data2015.Stops(Rows_to_be_deleted)        = [];
OAG_Data2015.SpecificAcft(Rows_to_be_deleted) = [];
OAG_Data2015.Frequency(Rows_to_be_deleted)    = [];

% save('.\Output\OAG_Data2015','OAG_Data2015')

%%

% % Lines 145 to 205 creates the OD_pairs_2015 file.
% tic;
%
% kk = 0; %m is a counter for the number of rows in OD_pairs_2015
%
% %Creates a unique list of DepAirports from OAG_Data_2015
% unique_departure_airports = unique([OAG_Data2015.DepAirport]);
%
% %Length of unique_departure_airports
% length_of_unique_departure_airports = length(unique_departure_airports);
%
%
% for origin = 1:length_of_unique_departure_airports
%
%     disp(['  Origin Airport: ', num2str(origin), '/', num2str(length_of_unique_departure_airports)]);
%
%     %Identify where a specific Airport from unique_departure_airports is
```

```
%     %located in OAG_Data2015.DepAirport
%     arr_cities_index = find(strcmp(unique_departure_airports(origin),[OAG_Data2015.DepAirport]));
%
%     %Using the arr_cities_index the following variables are assigned
%     %identified
%     if (isempty(arr_cities_index) == 0) %Indicates that the airport from unique_departure_airports is in
OAG_Data2015.DepAirport
%
%        arr_cities       = [OAG_Data2015.ArrAirport(arr_cities_index)];
%        OD_Seats         = [OAG_Data2015.Seats(arr_cities_index)];
%        OD_Frequency     = [OAG_Data2015.Frequency(arr_cities_index)];
%        Total_Seats      = [OD_Seats.*OD_Frequency];
%        SpecificAcftName = [OAG_Data2015.SpecificAcftName(arr_cities_index)];
%
%     end %if (isempty(arr_cities_index) == 0)
%
%     %Creates a unique list of the arrival airports
%     unique_arr_cities = unique(arr_cities);
%
%     %This for loop assigns the following fields to each OD pair
%     for destination = 1:length(unique_arr_cities)
%
%        if (length(find(strcmp(unique_arr_cities(destination),arr_cities) == 1)) >= 1)
%
%           entries = find(strcmp(unique_arr_cities(destination),arr_cities));
%
%           kk = kk + 1;
%           OD_pairs_2015.departure_airport(kk,1) = unique_departure_airports(origin);
%           OD_pairs_2015.arrival_airport(kk,1)   = unique_arr_cities(destination);
%           OD_pairs_2015.Total_Seats(kk,1)       = sum(Total_Seats(entries));
%           OD_pairs_2015.AcftData(kk).AcftSet    = SpecificAcftName(entries);
%           OD_pairs_2015.AcftData(kk).SeatSet    = Total_Seats(entries);
%           OD_pairs_2015.AcftData(kk).UniqueAcft = unique(OD_pairs_2015.AcftData(kk).AcftSet);
%
%
%           length_of_acft_set = length(OD_pairs_2015.AcftData(kk).UniqueAcft);
```

```
%

%            for acft = 1:length_of_acft_set

%               % %

%               acft_index =
find(strcmp(OD_pairs_2015.AcftData(kk).UniqueAcft(acft),OD_pairs_2015.AcftData(kk).AcftSet));

%               % %

%               OD_pairs_2015.AcftData(kk).Ratio(acft) =  sum(OD_pairs_2015.AcftData(kk).SeatSet(acft_index)) ./
sum(OD_pairs_2015.AcftData(kk).SeatSet);

%

%

%            end %for acft = 1:length_of_acft_set

%

%        end %if (length(find(strcmp(unique_arr_cities(destination),arr_cities) == 1)) >= 1)

%

%     end %for destination = 1:length(unique_arr_cities)

%

% end %for origin = 1:length_of_unique_departure_airports

%

% toc;

%

% clear unique_departure_airports unique_arr_cities Total_Seats SpecificAcftName row origin OD_Seats
OD_Frequency OAG_Data_Ser month m length_of_unique_departure_airports length_of_data length_of_acft_set
entries destination arr_cities_index arr_cities acft_index acft

%

% % Save the final output "OA_pairs_2015"

% save('.\Output\OD_pairs_2015.mat', 'OD_pairs_2015')


% New Solution --------------------


% Get OD Pair Data

OAG_Data_OD_Pairs = strcat(OAG_Data2015.DepAirport, '_', OAG_Data2015.ArrAirport);

[Unique_OAG_Data_OD_Pairs, OD_Pair_First_Index, OD_Pair_Index] = unique(OAG_Data_OD_Pairs);


Number_Of_Unqiue_OD_Pairs = length(Unique_OAG_Data_OD_Pairs);


% Pre-Allocation

OD_pairs_2015.departure_airport = cell(Number_Of_Unqiue_OD_Pairs,1);
```

```
OD_pairs_2015.arrival_airport   = cell(Number_Of_Unqiue_OD_Pairs,1);
OD_pairs_2015.Total_Seats       = zeros(Number_Of_Unqiue_OD_Pairs,1);

OD_pairs_2015.AcftData(Number_Of_Unqiue_OD_Pairs,1).AcftSet    = [];
OD_pairs_2015.AcftData(Number_Of_Unqiue_OD_Pairs,1).SeatSet    = [];
OD_pairs_2015.AcftData(Number_Of_Unqiue_OD_Pairs,1).UniqueAcft = [];

% Loop through each OD pair
for od = 1:Number_Of_Unqiue_OD_Pairs

    if mod(od, 1000) == 0
        disp(['  OD Pair: ', num2str(od), '/', num2str(Number_Of_Unqiue_OD_Pairs)]);
    end

    Current_OD_Pair_Indices = OD_Pair_Index == od;

    OD_pairs_2015.departure_airport(od,1) = OAG_Data2015.DepAirport(OD_Pair_First_Index(od));
    OD_pairs_2015.arrival_airport(od,1)   = OAG_Data2015.ArrAirport(OD_Pair_First_Index(od));

    OD_Seats     = OAG_Data2015.Seats(Current_OD_Pair_Indices);
    OD_Frequency = OAG_Data2015.Frequency(Current_OD_Pair_Indices);
    Total_Seats  = OD_Seats .* OD_Frequency;

    OD_pairs_2015.Total_Seats(od,1) = sum(Total_Seats);

    % Summarize by aircraft type
    OD_pairs_2015.AcftData(od,1).AcftSet    = OAG_Data2015.SpecificAcft(Current_OD_Pair_Indices);
    OD_pairs_2015.AcftData(od,1).SeatSet    = Total_Seats;

    [Unique_Aircraft, ~, Aircraft_Index] = unique(OD_pairs_2015.AcftData(od,1).AcftSet);

    Number_Of_Unique_Aircraft = length(Unique_Aircraft);

    OD_pairs_2015.AcftData(od,1).UniqueAcft = Unique_Aircraft;
    OD_pairs_2015.AcftData(od,1).Ratio      = zeros(1,Number_Of_Unique_Aircraft);
```

```
    for acft = 1:Number_Of_Unique_Aircraft

        Current_Aircraft_Indices = Aircraft_Index == acft;

        OD_pairs_2015.AcftData(od,1).Ratio(1,acft) = sum(Total_Seats(Current_Aircraft_Indices)) /
OD_pairs_2015.Total_Seats(od,1);

    end % for acft = 1:Number_Of_Unique_Aircraft

end % for od = 1:Number_Of_Unqiue_OD_Pairs

save('.\Output\OD_pairs_2015.mat', 'OD_pairs_2015');

% % Validate
% if all(OD_pairs_2015.Total_Seats == OD_pairs_2015.Total_Seats) == 0
%     error('New is not equal to old!');
% end
%
% for od = 1:Number_Of_Unqiue_OD_Pairs
%
%     % Seat set
%     New_SeatSet = OD_pairs_2015.AcftData(od).SeatSet;
%     Old_SeatSet = OD_pairs_2015.AcftData(od).SeatSet;
%
%     if all(New_SeatSet == Old_SeatSet) == 0
%         od
%         error('New is not equal to old!');
%     end
%
%     % Ratio
%     New_Ratio = OD_pairs_2015.AcftData(od).Ratio;
%     Old_Ratio = OD_pairs_2015.AcftData(od).Ratio;
%
%     if all(New_Ratio == Old_Ratio) == 0 && isnan(Old_Ratio) == 0
%         od
```

```
%         error('New is not equal to old!');
%     end
%
% end

return;
```

Reduction_AcftData_OAG.m

```
%Updating OAG_pairs_2015 AcftData. UniqueAcft&Ratios
function [OD_pairs_2015] = Reduction_AcftData_OAG()

load ('.\Output\OD_pairs_2015.mat')
load ('.\Output\aircraft_list.mat');

aircraft_list_IATA      = {aircraft_list.IATA}';
aircraft_list_BADA_Type = {aircraft_list.BADA_Type}';

length_OD_pair_OAG_2015 = length(OD_pairs_2015.AcftData);

for OD = 1:length_OD_pair_OAG_2015

   if mod(OD, 10000) == 0
      disp(OD)
   end

   Current_Aircraft = OD_pairs_2015.AcftData(OD).UniqueAcft;

   [~, Aircraft_Index] = ismember(Current_Aircraft, aircraft_list_IATA);

   OD_pairs_2015.AcftData(OD).Acft_Updated = aircraft_list_BADA_Type(Aircraft_Index);

%    length_acft_set = length(OD_pairs_2015.AcftData(OD).UniqueAcft);
%
%    for acft = 1:length_acft_set
%
%       if strcmp('Boeing 767-300 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%             || strcmp('Boeing 767-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%          OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B763__'};
%
%       elseif strcmp('Airbus Industrie A350',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
```

```
%            || strcmp('Airbus Industrie A350-900',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 777 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 777-200/200ER Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B772__'};
%
%    elseif strcmp('Boeing 747 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 747 (Passenger)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 747-400 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 747-400 (Passenger)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B744__'};
%
%    elseif strcmp('Boeing 727 (Passenger)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 727-200 (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 727-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 757 (Passenger)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 757-200 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 757-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 757-300 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 757-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Tupolev TU154',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Tupolev TU-204 /tu-214',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B752__'};
%
%    elseif strcmp('Airbus A330-300',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A333__'};
%
%    elseif strcmp('Airbus A340',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A340-200',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A340-300',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A340-500',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A340-600',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
```

```
%            || strcmp('Ilyushin Il-96 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Lockheed L1011 Tristar Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A346__'};
%
%    elseif strcmp('Airbus A380-800 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus Industrie A380 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A388__'};
%
%    elseif strcmp('Boeing 767-400 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B764__'};
%
%    elseif strcmp('Boeing 767 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 767-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) =  {'B762__'};
%
%    elseif strcmp('Boeing 737-400 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 737-400 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 737-700 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 737-700 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 737-700 Freighter',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 737-700 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B737__'};
%
%    elseif strcmp('Airbus A320',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A320 (sharklets)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A320__'};
%
%    elseif strcmp('Airbus A300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300-600 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
```

```
%            || strcmp('Airbus A300B2 /B4 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300 /600 /600C (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300 600 /600C (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300 all Series',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300-600 /600C (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A300B2 /B4 /C4 (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310 all Series',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310-200',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310-300',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Airbus A310-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A310__'};
%
%    elseif strcmp('Boeing 777-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 777-300ER Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B773__'};
%
%    elseif strcmp('Boeing 787',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B77W__'};
%
%    elseif strcmp('Boeing 777-200LR',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B77L__'};
%
%    elseif strcmp('Cessna Citation',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Hawker 400 Beechjet/400A/400XP/400T',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'CL60__'};
%
%    elseif strcmp('Boeing 787-8',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Boeing 787-9',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
```

```
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B788__'};
%
%       elseif strcmp('Boeing 747-8 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 747-8i Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 747SP Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B748__'};
%
%       elseif strcmp('Boeing 737-300 (Freighter)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-300 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-300 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B733__'};
%
%       elseif strcmp('Airbus A321',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Airbus Industrie A321 (Sharklets)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A321__'};
%
%       elseif  strcmp('Embraer RJ 135 /140 /145',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer RJ 135/140/145',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer RJ145',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'E145__'};
%
%       elseif strcmp('Embraer 175 (Enhanced Winglets)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer 190',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer 195',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Fokker 100',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Sukhoi Superjet 100-95',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'E190__'};
%
```

```
%       elseif strcmp('Embraer 170',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer 170 /195',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer 170/195',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer 175',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Yakovlev Yak-40',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Yakovlev Yak-42 /142',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'E170__'};
%
%       elseif strcmp('Airbus A318',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Airbus A318 /319 /320 /321',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Airbus A318 /319 /320 /321',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Airbus A319',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A319__'};
%
%       elseif strcmp('Embraer RJ135',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Embraer RJ140',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Fairchild Dornier 328jet',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'E135__'};
%
%       elseif strcmp('Antonov An148-100',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Antonov An-158',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Avro RJ100',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Avro RJ70',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Avro RJ85',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('BAe 146 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('BAe 146-100 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('BAe 146-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('BAe 146-300 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing (douglas) MD-88',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing (douglas) MD-90',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 717-200',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Tupolev TU134',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
```

```
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B712__'};
%
%       elseif strcmp('Boeing 737 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737 Advanced all Series',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-200 (Mixed Configuration)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-200 /200C Advanced (Pax)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('Boeing 737-200 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-500 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-500 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-600 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B735__'};
%
%       elseif strcmp('Boeing 737-800 (Scimitar Winglets) Pax',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('Boeing 737-800 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-800 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B738__'};
%
%       elseif strcmp('Boeing 737-900 (winglets) Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing 737-900 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'B739__'};
%
%       elseif strcmp('Airbus A330',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Airbus A330-200',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'A332__'};
%
%       elseif strcmp('Antonov AN-140',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('ATR 42-300 /320',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('ATR 42-500',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
```

```
%            || strcmp('BAe Jetstream',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('BAe Jetstream 31',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('BAe Jetstream 32',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Beechcraft (Lght Acft - Twin Turboprop)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%            || strcmp('Beechcraft C99 Airliner',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('DHvilld-Bombardier DHC4 Caribou',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('DHvilld-Bombardier DHC6 T/Otter',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Embraer 110 Bandeirante',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Embraer 120 Brasilia',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Fairchild Dornier 228',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Fairchild Dornier 328-100',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Fairchild Sa26 /Sa226 /Sa227 Merlin /Me',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%            || strcmp('Let 410',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR 42',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR 42 300 /400',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR 42-500',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR all Series',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Shorts 360 (sd3-60)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'AT45__'};
%
%    elseif strcmp('ATR 72',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR 72',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('ATR42 /ATR72',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'AT72__'};
%
%    elseif strcmp('Canadair CRJ Series 705',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Canadair Regional Jet',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Canadair Regional Jet 100',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%            || strcmp('Canadair Regional Jet 200',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%        OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'CRJ2__'};
%
```

```
%       elseif strcmp('Canadair (Bombardier) Regional Jet 1000',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%           || strcmp('Canadair Regional Jet 700',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Canadair Regional Jet 900',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%          OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'CRJ9__'};
%
%       elseif strcmp('Antonov AN-24',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Antonov An-26',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('BAe (HS) 748',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('BAe ATP',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Convair 580 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('DHvilld-Bombardier DHC8 Dsh 8',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('DHvilld-Bombardier DHC8-100 Dsh 8/8Q',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%           || strcmp('DHvilld-Bombardier DHC8-200 Dsh 8/8Q',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%           || strcmp('DHvilld-Bombardier DHC8-300 Dsh 8/8Q',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%           || strcmp('Fokker 50',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Fokker 70',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Fokker F28 Fellowship',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Harbin Yunshuji Y12',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Ilyushin IL114',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Saab 340',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Saab 340B',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Xian Yunshuji Ma-60',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Xian Yunshuji Y7 /Ma60',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%          OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'DH8C__'};
%
%       elseif strcmp('Antonov AN-26 /30 /32',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('Antonov An-32',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('DHvilld-Bombardier DHC7 Dsh 7',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%           || strcmp('DHvilld-Bombardier DHC8 Dsh 8-400/8Q',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%           || strcmp('Saab 2000',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
```

```
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'DH8D__'};
%
%       elseif strcmp('Antonov AN-38',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Beechcraft (Light Aircraft)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Beechcraft-Lght Acft-Twin Piston Engine',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('BN BN-2a /BN-2b Islander',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('BN BN-2a Mk.iii Trislander',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Boeing (Douglas) DC-3 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Cessna (Light Aircraft)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Cessna 208b Freighter',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Cessna Light Acft (Twin piston engines)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('Cessna Light Acft(single Piston Engine)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('Cessna Light Aircraft (Twin Turboprop',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Cessna Light Aircraft(Single Turboprop)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
...
%               || strcmp('DHvilld-Bombardier DHC2 Bvr',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('DHvilld-Bombardier DHC3 Tb Otter',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Equipment Varies',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Pilatus PC-12',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Piper (Light Aircraft - Single Piston)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Piper (Light Aircraft - Twin Piston)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Piper (Light Aircraft - Twin Turboprop)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Piper (Light Aircraft)',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'C208__'};
%
%       elseif strcmp('BAe Jetstream 41',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Beechcraft 1900 Airliner',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%               || strcmp('Beechcraft 1900D Airliner',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%           OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'BE99__'};
%
%       elseif strcmp('Boeing (douglas) DC-9 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
```

```
%                || strcmp('Boeing (Douglas) DC9-30 Passenger',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%                || strcmp('Boeing (douglas) MD-80',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%                || strcmp('Boeing (douglas) MD-82',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%                || strcmp('Boeing (douglas) MD-83',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1 ...
%                || strcmp('Boeing (douglas) MD-87',OD_pairs_2015.AcftData(OD).UniqueAcft(acft)) == 1
%
%            OD_pairs_2015.AcftData(OD).Acft_Updated(acft) = {'MD82__'};
%
%         end
%     end
end

%%
for OD = 1:length_OD_pair_OAG_2015

   try
      OD_pairs_2015.AcftData(OD).UniqueAcft_Updated = unique(OD_pairs_2015.AcftData(OD).Acft_Updated);
   catch
      disp(['ERROR: Unmatched aircraft for OD ', num2str(OD)]);

      Number_Of_Aircraft = length(OD_pairs_2015.AcftData(OD).Acft_Updated);

      for acft = 1:Number_Of_Aircraft

         if isnan(OD_pairs_2015.AcftData(OD).Acft_Updated{acft})

            disp([' - ', OD_pairs_2015.AcftData(OD).UniqueAcft{acft}]);

         end % if isnan(OD_pairs_2015.AcftData(OD).Acft_Updated{acft})

      end % for acft = Number_Of_Aircraft

   end % try

end % for OD = 1:length_OD_pair_OAG_2015
```

```
%%
for OD = 1:length_OD_pair_OAG_2015

    length_of_acft_set = length(OD_pairs_2015.AcftData(OD).UniqueAcft_Updated);

    for acft = 1:length_of_acft_set

        acft_match =
find(strcmp(OD_pairs_2015.AcftData(OD).UniqueAcft_Updated(acft),OD_pairs_2015.AcftData(OD).Acft_Updated));
        OD_pairs_2015.AcftData(OD).Ratio_Updated(acft) = sum(OD_pairs_2015.AcftData(OD).Ratio(acft_match));
    end
end

save('.\Output\OD_pairs_2015.mat', 'OD_pairs_2015');

return;
```

FRATAR_Model_Main_File.m

```
%Main File to run FRATAR Model

function FRATAR_Model_Main_File(Install_Dir, Project_Dir, Case_Folder, Case_Name, Start_Year, End_Year,
Price_Drop_Percent)


clc;

warning off;


% ----------------------

% Settings if run inside MATALB

% ----------------------

% IMPORTANT: All inputs are now organized in "..\GDM_DATA\" folder. This

% the Git project GDM_DATA and contains the official inputs for this model

% for the GDM_GUI.

% Model runs are now saved in the following directory structure:

% Save_Dir = Project_Dir\Output\Case_Folder\Case_Name (See below)


if isdeployed == 0

    Install_Dir = '..\GDM_DATA\DATA';

    Project_Dir = 'C:\GDM Project\Passenger';

    Case_Folder = 'Test_Folder';

    Case_Name   = 'Passenger Trip Distribution';


    % Trip Generation Output Folder

    TG_Case_Folder = 'Test_Folder';

    TG_Case_Name   = 'Passenger Trip Generation - 1996 - 2015';


    Start_Year = 2016;

    End_Year   = 2040;


    Price_Drop_Percent = 0;

end


% Create Output Dir
```

```
Save_Dir = [Project_Dir, '\Trip_Distribution\Output\', Case_Folder, '\', Case_Name];
mkdir(Save_Dir);

Trip_Generation_Dir = [Project_Dir, '\Trip_Generation\Output\', TG_Case_Folder, '\', TG_Case_Name];

%%
%3 This function generates a table with the forecast demand taking into
%%consideration the ticket price drop. Price drop is an input in this
%%function. Price drop would be 0, 1, 2, 5, or 10. This numbers represent
%%percentages
Forecast_Data_Table_Generator(Install_Dir, Save_Dir, Trip_Generation_Dir, Price_Drop_Percent, Start_Year,
End_Year);

%%
%4 FRATAR Model
Fratar_Model(Install_Dir, Save_Dir, Start_Year, End_Year);

%%
%5 Aircraft and Flights asignment
Aircraft_and_Flights_Assignment(Install_Dir, Save_Dir, Start_Year, End_Year);

return;
```

Forecast_Data_Table_Generator.m

```
function Forecast_Data_Table_Generator(Install_Dir, Save_Dir, Trip_Generation_Dir, Price_Drop_Percent,
Start_Year, End_Year)


%Load Forecast_Data

load([Trip_Generation_Dir, '\Forecast_Data.mat'])

%Load airport list

load([Install_Dir, '\Passenger\OAG_Passenger_Processing\airport_list.mat'])


%Length of airport list

length_of_airport_list = length(airport_list.Airport_IDs);


%Total number of year (25yrs in this case)

years = End_Year - Start_Year + 1;


%Create a table with zeros to be use later

future_departing_seats_each_year = zeros(length_of_airport_list,years);


for row = 1:length_of_airport_list


    for year = 1:years


        future_departing_seats_each_year(row,year) = Forecast_Data(row).Data(year);


    end

end


%Depending on the price_drop value inserted in the functino the demand

%decrease is defined in the following lines

if Price_Drop_Percent == 0


    demand_increase_by = 1.0;


elseif Price_Drop_Percent == 1

    demand_increase_by = 1.008;
```

```
elseif Price_Drop_Percent == 2
    demand_increase_by = 1.016;

elseif Price_Drop_Percent == 5
    demand_increase_by = 1.04;

    elseif Price_Drop_Percent == 10
    demand_increase_by = 1.085;

end

%Take the forecast data and multiply it by the corresponded
%demand_increase_by value
future_departing_seats_each_year = round(future_departing_seats_each_year .* demand_increase_by);

save([Save_Dir, '\future_departing_seats_each_year.mat'], 'future_departing_seats_each_year')

return;
```

Fratar_Model.m

```
function Fratar_Model(Install_Dir, Save_Dir, Start_Year, End_Year)

%Load the 2 inputs needed to run de model

%1) 2015 OD pairs from OAG_2015
%2) OAG_2015 Airport list
%clear all
%clc;

%close all;

%mkdir 'Output'

load([Install_Dir, '\Passenger\OAG_Passenger_Processing\OD_pairs_2015.mat'])
load([Save_Dir, '\future_departing_seats_each_year.mat'])
load([Install_Dir, '\Passenger\OAG_Passenger_Processing\airport_list.mat'])

%%
%Determine how many airportss the airport list has
length_of_airport_list = length(airport_list.Airport_IDs);

%This part populate the OD matrix according to OAG_2015. The result is
%a matrix with the number of seat going from each airport of origin to each
%one of its aiports of destination

[Lia_dep, departure_airport_index] = ismember([OD_pairs_2015.departure_airport], airport_list.Airport_IDs);
[Lia_arr, arrival_airport_index]   = ismember([OD_pairs_2015.arrival_airport], airport_list.Airport_IDs);

% Check if all airports indexed
% missing_departure_airport_index = find(Lia_dep == 0);
% missing_arrival_airport_index   = find(Lia_arr == 0);

% number_of_missing_departure_airports = length(missing_departure_airport_index);
```

```
% number_of_missing_arrival_airports   = length(missing_arrival_airport_index);


% % Display missing airports

% %for md = 1:number_of_missing_departure_airports

%  %   disp(['Missing departure airport for OD pair: ',
OD_pairs_2015(missing_departure_airport_index(md)).departure_airport{1}, ' -> ',
OD_pairs_2015(missing_departure_airport_index(md)).arrival_airport{1}]);

% end

% for ma = 1:number_of_missing_arrival_airports

%     disp(['Missing arrival airport for OD pair: ',
OD_pairs_2015(missing_arrival_airport_index(ma)).departure_airport{1}, ' -> ',
OD_pairs_2015(missing_arrival_airport_index(ma)).arrival_airport{1}]);

% end


% Get valid OD pairs (exclude the ones with missing airports)

Valid_OD_Pairs = Lia_dep == 1 & Lia_arr == 1;

seats_by_frequency = [OD_pairs_2015.Total_Seats];


% Convert from i,j indexing to linear

linearInd = sub2ind([length_of_airport_list, length_of_airport_list], departure_airport_index(Valid_OD_Pairs),
arrival_airport_index(Valid_OD_Pairs));


% Fill OD matrix

OD_matrix_1 = zeros(length_of_airport_list);

OD_matrix_1(linearInd) = seats_by_frequency(Valid_OD_Pairs);


% % Check that new code produces samre result

% Diff = OD_matrix_1 - OD_matrix_1;

%

% if min(min(Diff)) ~= 0 || max(max(Diff)) ~= 0

%     error('New and old code do not produce same results!');

% end


%toc

%%


%The FRATAR model establish the following 2 assumptions:
```

```
%1) The total number of seats (in this case)
%going from airport "A" to "B" is queal to the number of seats going
%from  "B" to "A"
%2) The number of seats departing airport "A" is equal to the number of
%seats arriving to airport "A"
%Hence, the OD_Matrix needed to run the model needs to be symmetric.
%The FRATAR model converts any OD_matrix into a symmetric matrix by
%calculatating the average of seats from "A" to "B" and "B" to "A" and
%making them equal.
%This for loop creates a symmetric OD_Matrix by calculating averages on
%the OD_matrix_1

%Creates a matrix with zeros to be use later
%symmetric_OD_matrix_original = zeros(length_of_airport_list);

symmetric_OD_matrix_original = (OD_matrix_1 + OD_matrix_1') / 2;

%Waitbar
%wb = waitbar(0,'Creating a Symmetric OD Matrix');

%Generates a symmetric OD matrix using "OD_matrix_1". Since is a symmetric
% %matrix I only need to calculate half of the values of the matrix.
% for row = 1:length_of_airport_list
%
%     %waitbar
%  %   waitbar(row/length_of_airport_list)
%
%     for column = row+1:length_of_airport_list
%
%         if (row ~= column)
%
%             symmetric_OD_matrix_original(row,column) = ((OD_matrix_1(row,column) +
OD_matrix_1(column,row))/2);
%
%             symmetric_OD_matrix_original(column,row) = symmetric_OD_matrix_original(row,column);
%
```

```
%         end
%     end
% end

%close (wb)
clear row column wb

%%

%From 2015-2040
%Start_Year = 2016;
%End_Year = 2040;

growth_factors_for_each_iteration = zeros(500,length_of_airport_list);

%Total number of year (25yrs in this case)
years = End_Year-Start_Year+1;

%for year = 1:years
for year = 1:years
    year_to_display = Start_Year + year - 1;

    disp(['Starting the analyis for year ',num2str(year_to_display)])

    %%

    %%

    %Using the symmetric_OD_matrix, the total number of seats departing
    %each airport of origin and destination is calculated. For now these 2
    %variables will have the same values. However, the values will change
    %once the iteration process of the FRATAR model starts

    %Creates an array with zeros to be use later
```

```
%total_seats_departing_origin = zeros(length_of_airport_list,1);
%total_seats_departing_destination = zeros(length_of_airport_list,1);

% ----------------------
% NEW CODE: You can replace the for loop above with the followign 2 lines
% Use vectoring when possible
% ----------------------
total_seats_departing_origin      = sum(symmetric_OD_matrix_original,2);
total_seats_departing_destination = sum(symmetric_OD_matrix_original,1)';
% ----------------------
% END NEW CODE
% ----------------------

%I had to use the round function or command to avoind floating numbers when
%calculation the Growth Factors. If I don't do this then Matlab will tell
%me that 1 is no equal to 1.0000
total_seats_departing_origin = round(total_seats_departing_origin);
total_seats_departing_destination = round(total_seats_departing_destination);

%%
  %calculate the initial Growth Factors

%Creates arrays with zeros to be used lated
airport_of_origin_GF = zeros(length_of_airport_list,1);
airport_of_destination_GF = zeros(length_of_airport_list,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Here is where the for loop to run the analysis from 2015-2040 could start.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The growth factors are calculated.
%The number 1 represents year=1. This will change when the analysis
%is run for all the years
for row = 1:length_of_airport_list
```

```
        if total_seats_departing_origin(row,1) == 0


            airport_of_origin_GF(row,1) = future_departing_seats_each_year(row,year);

        else

            airport_of_origin_GF(row,1) = future_departing_seats_each_year(row,year) ./
total_seats_departing_origin(row,1) ;

        end


        if total_seats_departing_destination(row,1) == 0

            airport_of_destination_GF(row,1) = future_departing_seats_each_year(row,year);

        else

            airport_of_destination_GF(row,1) = future_departing_seats_each_year(row,year) ./
total_seats_departing_destination(row,1);

        end


        %Solve issue with airport 770 on airport list. Airport #770 on the

        %airport list has a value for the total seats departing that airport.

        %However, in the Forecast_data this airport appears but there is no

        %data available.

        if airport_of_origin_GF(row,1) == 0

            airport_of_origin_GF(row,1) = 1;

        end


        if airport_of_destination_GF(row,1) == 0

            airport_of_destination_GF(row,1) = 1;

        end


    end


    clear row


    %%


    %The equation of the Fratar model has 4 terms.

    %1) Current seats ging from origin to destination

    %2) Growth factor of the airprot of origin
```

```
%3) Growth factor of the airprot of destinatin
%4) A factor that I called "K factor" and it takes into consideration the
%growth between the airport of origin and each one of its destination.

%This k_factor is a little hard to visualized or to understand it. Let me
%know if I need to explain this to you in a better way.

k_factor_denominator = zeros(length_of_airport_list,1);
k_factor_product = zeros(1,length_of_airport_list);
k_factor = zeros(length_of_airport_list,1);

for row = 1:length_of_airport_list

    for column = 1:length_of_airport_list

        %The number 1 represents year=1. This will change when the analysis
        %is run for all the years
        k_factor_product(1,column) = symmetric_OD_matrix_original(row,column) .*
airport_of_destination_GF(column,1);

    end

    %The number 1 represents year=1. This will change when the analysis
    %is run for all the years
    k_factor_denominator(row,1) = sum(k_factor_product);

end
clear row column

for row = 1:length_of_airport_list


    if k_factor_denominator(row,1) == 0
        k_factor(row,1) =  0;
    else
        k_factor(row,1) = total_seats_departing_origin(row,1) ./ k_factor_denominator(row,1);
```

```
    end

end

clear row

%%

symmetric_OD_matrix = symmetric_OD_matrix_original;

%Creates a matrix with zeros to be used later
% symmetric_OD_matrix_new = zeros(length_of_airport_list);

%growth_factors_for_each_iteration = zeros(200,length_of_airport_list);

%Here is where the iteration process of the FRATAR model starts.

%While any of the growth factor of origin or destination is bigger than
%1.005 or smaller than 0.995 the iteration process won't stop

%The iteration process can be divided in 5 steps
%while (max(max(airport_of_origin_GF(:,1)),max(airport_of_destination_GF(:,1))) > 1.005) &&
(min(min(airport_of_origin_GF(:,1)),min(airport_of_destination_GF(:,1))) < 0.995) || Iteration_Number == 500
for Iteration_Number = 1:500

    %Step 1
    if mod(Iteration_Number, 50) == 0
        disp(['Year ',num2str(year_to_display),': Iteration Number ',num2str(Iteration_Number)])
    end

    %%
        for column = 1:length_of_airport_list
            growth_factors_for_each_iteration(Iteration_Number,column) = airport_of_origin_GF(column,1);
        end
    %%
```

```
%This for loop use the Fratar equation to calculate the new value of the
%OD pair
%       for row = 1:length_of_airport_list
%
%           for column = 1:length_of_airport_list
%
%               symmetric_OD_matrix(row,column) = symmetric_OD_matrix(row,column) .*
airport_of_origin_GF(row,1) .* airport_of_destination_GF(column,1) .* k_factor(row,1);
%
%           end
%       end
%       clear row colum


symmetric_OD_matrix = symmetric_OD_matrix .* airport_of_origin_GF .* airport_of_destination_GF' .* k_factor;


%       if all(all(symmetric_OD_matrix_2 == symmetric_OD_matrix)) == 0
%           error('Not equal');
%       end


%Step 2


%This for loop creates the new symmetrix OD matrix according to the
%values obtain in the previous step


%    for row = 1:length_of_airport_list
%
%       for column = row+1:length_of_airport_list
%
%           if (row ~= column)
%
%               symmetric_OD_matrix_new(row,column) = ((symmetric_OD_matrix(row,column) +
symmetric_OD_matrix(column,row))/2);
%
%               symmetric_OD_matrix_new(column,row) = symmetric_OD_matrix_new(row,column);
%
%           end
```

```
%        end
%    end
%
%    clear row column

symmetric_OD_matrix_new = (symmetric_OD_matrix + symmetric_OD_matrix') / 2;

%Step 3

%This for loop re-calculates the total number of seats departing origin
%and destination. If the sum is equal to zeros then the variable gets a
%value of 1.

%        for row = 1:length_of_airport_list
%
%            if sum(symmetric_OD_matrix_new(row,:)) == 0
%                total_seats_departing_origin(row,1) = 1;
%            else
%                total_seats_departing_origin(row,1) = sum(symmetric_OD_matrix_new(row,:));
%            end
%            if sum(symmetric_OD_matrix_new(:,row)) == 0
%                total_seats_departing_destination(row,1) = 1;
%            else
%                total_seats_departing_destination(row,1) = sum(symmetric_OD_matrix_new(:,row));
%            end
%
%        end
%
%        clear row

total_seats_departing_origin = sum(symmetric_OD_matrix_new,2);
zero_seats_index = total_seats_departing_origin == 0;
total_seats_departing_origin(zero_seats_index) = 1;

total_seats_departing_destination = sum(symmetric_OD_matrix_new,1)';
```

```
zero_seats_index = total_seats_departing_destination == 0;
total_seats_departing_destination(zero_seats_index) = 1;

%       if all(total_seats_departing_origin_2 == total_seats_departing_origin) == 0
%           error('Not equal');
%       end
%       if all(total_seats_departing_destination_2 == total_seats_departing_destination) == 0
%           error('Not equal');
%       end

%I had to use the round function or command to avoind floating numbers when
%calculation the Growth Factors. If I don't do this then Matlab will tell
%me that 1 is no equal to 1.0000
total_seats_departing_origin = round(total_seats_departing_origin);
total_seats_departing_destination = round(total_seats_departing_destination);

%Step 4
%This for loop re-calculates the growth factor and check against the
%while condition

for row = 1:length_of_airport_list

    if total_seats_departing_origin(row,1) == 1

        airport_of_origin_GF(row,1) = 1;
    else
        airport_of_origin_GF(row,1) = future_departing_seats_each_year(row,year) ./
total_seats_departing_origin(row,1) ;
    end

    if total_seats_departing_destination(row,1) == 1
        airport_of_destination_GF(row,1) = 1;
    else
        airport_of_destination_GF(row,1) = future_departing_seats_each_year(row,year) ./
total_seats_departing_destination(row,1);
    end
```

```
end
clear row


%step 5
%Terms for the calculation of the k factor are re-calculated


%       for row = 1:length_of_airport_list
%
% %           for column = 1:length_of_airport_list
% %               % NOTE: According to the MATLAB profiler, this line of code
% %               % takes 99% of the execution time.
% %               k_factor_product(1,column) = symmetric_OD_matrix_new(row,column) .*
airport_of_destination_GF(column,1);
% %           end
%
%           % --------------------
%           % NEW CODE: To replace for loop above. This is about 5x faster
%           % Note the ' after airport_of_destination_GF. When using .* the
%           % vectors need to be the same size.
%           % --------------------
%           k_factor_product = symmetric_OD_matrix_new(row,:) .* airport_of_destination_GF';
%
% %           if all(k_factor_product_2 == k_factor_product) == 0
% %              error('Not equal');
% %           end
%           % ------------------------
%           % END NEW CODE
%           % ------------------------
%           k_factor_denominator(row,1) = sum(k_factor_product);
%
%       end
%       clear row column


k_factor_product = symmetric_OD_matrix_new .* airport_of_destination_GF';
k_factor_denominator = sum(k_factor_product,2);
```

```
    %       if all(k_factor_denominator_2 == k_factor_denominator) == 0
    %           error('Not equal');
    %       end

    clear k_factor_product

    for row = 1:length_of_airport_list

        if k_factor_denominator(row,1) == 0
            k_factor(row,1) = 0;
        else
            k_factor(row,1) = total_seats_departing_origin(row,1) ./ k_factor_denominator(row,1);
        end
    end

    clear row

    %change variable name from "symmetric_OD_matrix_new" to
    %"symmetric_OD_matrix" so it can be used again at the beginnig of the
    %iterration process.
    symmetric_OD_matrix = symmetric_OD_matrix_new;

end %ends the while loop

%After the iteration proces is completed; this for loop needs to be
%done in order to eliminate the creation of new od pairs. This needs to
%be that way since the Aircraft_and_Flight_Assignment is based on OAG
%OD matrix size. If the OD didn't happened in OAG 2015 then the model
%will not know what type of Aircraft should be assigned to the new OD
%pair. The new OD pairs are being created since there might be and
%airport with forecast data that didn't showed up in OAG 2015.
for row_origin = 1:length_of_airport_list

    for column_destination = 1:length_of_airport_list
```

```
            if OD_matrix_1(row_origin,column_destination) == 0

                symmetric_OD_matrix(row_origin,column_destination) = 0;
            end
        end
    end

    symmetric_OD_matrix = ceil(symmetric_OD_matrix);

    save([Save_Dir, '\New Trip Distribution for year ',num2str(year_to_display), '.mat'], 'symmetric_OD_matrix')

    disp(['End of analysis for year ',num2str(year_to_display)])

end

clear years year year_to_display year total_seats_departing_origin ...
    total_seats_departing_destination  seats_by_frequency ...
    OD_matrix_1 number_of_missing_departure_airports ...
    number_of_missing_arrival_airports missing_departure_airport_index ...
    missing_arrival_airport_index md ma m linearInd Lia_dep Lia_arr ...
    k_factor_product k_factor_denominator k_factor Iteration_Number i ...
    departure_airport_index column arrival_airport_index ...
    airport_of_origin_GF airport_of_destination_GF

%%

for year = Start_Year:End_Year
    m = 1;
    % year_to_disp = year_to_disp +1;

    disp(year)

    load([Save_Dir, '\New Trip Distribution for year ',num2str(year),'.mat'])
    for row = 1:length_of_airport_list
        for column = 1:length_of_airport_list
```

```
        if symmetric_OD_matrix(row,column) ~= 0

            Trip_Distribution_All.(['Year_', num2str(year)]).Departure_Airport(m,1) = airport_list.Airport_IDs(row);
            Trip_Distribution_All.(['Year_', num2str(year)]).Arrival_Airport(m,1) = airport_list.Airport_IDs(column);
            Trip_Distribution_All.(['Year_', num2str(year)]).Seats(m,1) = symmetric_OD_matrix(row,column);
            m = m+1;
        end
    end
  end
end

%The following lines of code identifies OD pairs with less than 3 nm or over
%7,500 nm of distance and eliminate them from the analysis.
OD_to_delete = zeros(1,1);
row_counter = 1;

for year = Start_Year

  for OD = 1:36249%length_of_airport_list



    origin_location =
find(ismember(airport_list.Airport_IDs,Trip_Distribution_All.Year_2016.Departure_Airport(OD)));
    destination_location =
find(ismember(airport_list.Airport_IDs,Trip_Distribution_All.Year_2016.Arrival_Airport(OD)));


    noWaypoints = 20;
    originLat = airport_list.Apt_Lat(origin_location);
    originLon = airport_list.Apt_Lon(origin_location);
    destLat   = airport_list.Apt_Lat(destination_location);
    destLon   = airport_list.Apt_Lon(destination_location);



    [distanceGCD] = generateGCD(originLat,originLon,destLat, destLon, noWaypoints);
```

```
        if distanceGCD < 100 || distanceGCD > 9000

            OD_to_delete(row_counter,1) = OD;
            row_counter = row_counter + 1;
        end
    end
end

if OD_to_delete(1,1) > 0

    for year = Start_Year:End_Year

        Trip_Distribution_All.(['Year_', num2str(year)]).Departure_Airport(OD_to_delete) = [];
        Trip_Distribution_All.(['Year_', num2str(year)]).Arrival_Airport(OD_to_delete) = [];
        Trip_Distribution_All.(['Year_', num2str(year)]).Seats(OD_to_delete) = [];

    end

end

save ([Save_Dir, '\Trip_Distribution_All_Years'], 'Trip_Distribution_All')

return;
```

Aircraft_and_Flights_Assignment.m

```
function Aircraft_and_Flights_Assignment(Install_Dir, Save_Dir, Start_Year, End_Year)

load([Install_Dir, '\Passenger\OAG_Passenger_Processing\OD_pairs_2015.mat'])
load([Save_Dir, '\Trip_Distribution_All_Years.mat']);
load([Install_Dir, '\Passenger\Fleet_Evolution\Acft_Seating_Capacity.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\Aircraft_list_updated.mat'])

OD_pairs_2015_Dep_Arr = strcat([OD_pairs_2015.departure_airport], '_', [OD_pairs_2015.arrival_airport]);

for year = Start_Year:End_Year

    disp(num2str(year));

    Trip_Distribution_All_Departure_Airport = Trip_Distribution_All.(['Year_', num2str(year)]).Departure_Airport;
    Trip_Distribution_All_Arrival_Airport   = Trip_Distribution_All.(['Year_', num2str(year)]).Arrival_Airport;

    length_of_od_pairs = length(Trip_Distribution_All_Departure_Airport);

    % wb1 = waitbar(0,'Analyzing OD Pairs...');

    % Matching OAG OD pairs with TD OD pairs
    Trip_Distribution_All_Dep_Arr = strcat(Trip_Distribution_All_Departure_Airport, '_',
Trip_Distribution_All_Arrival_Airport);
    [OD_pairs_2015_Index_Found, OD_pairs_2015_Index] = ismember(Trip_Distribution_All_Dep_Arr,
OD_pairs_2015_Dep_Arr);

    for row = 1:length_of_od_pairs
        %disp(row)
        %waitbar(row/length_of_od_pairs);

        % entries1 =
strcat('find(strcmp(Trip_Distribution_All.Year_',num2str(year),'.Departure_Airport(',num2str(row),'),[OD_pairs_2015.de
parture_airport]));');
        %entries_dep = eval(entries1);
```

```
    % entries_dep = find(strcmp(Trip_Distribution_All.(['Year_',
num2str(year)]).Departure_Airport(row),[OD_pairs_2015.departure_airport]));


    % entries_arr = find(strcmp(Trip_Distribution_All.(['Year_',
num2str(year)]).Arrival_Airport(row),[OD_pairs_2015.arrival_airport(entries_dep)]));

    % OD_pairs_2015_Index(row) indicates the specific row of each OD pair in the

    % OD_pairs_2015


    %entries2 =
strcat('find(strcmp(Trip_Distribution_All.Year_',num2str(year),'.Arrival_Airport(',num2str(row),'),[OD_pairs_2015(entrie
s_dep).arrival_airport]));');


    %entries_arr = eval(entries2);


%       if entries_dep(entries_arr) ~= OD_pairs_2015_Index(row)

%           1

%       end


    if OD_pairs_2015_Index_Found(row) %there's a match


        %evalstring = strcat('Aircraft_Distribution_All(row).Year_',num2str(year),' =
Acft_Distribution_2015(OD_pairs_2015_Index(row)).Unique_Acft_by_OD_pair;');

        %eval(evalstring)

        %Aircraft_Distribution_All(row).(['Year_',num2str(year)]) =
Acft_Distribution_2015(OD_pairs_2015_Index(row)).Unique_Acft_by_OD_pair;

        Aircraft_Distribution_All(row).Aircraft =
OD_pairs_2015.AcftData(OD_pairs_2015_Index(row)).UniqueAcft_Updated;


        %eval(strcat('acft_set = Aircraft_Distribution_All(row).Year_',num2str(year),';'));

        acft_set = Aircraft_Distribution_All(row).Aircraft;


        %evalstring2 = strcat('Seats_assigned_to_each_acft_all(row).Year_',num2str(year),' =
Trip_Distribution_All.Year_',num2str(year),'.Seats(row) .*
Acft_Distribution_2015(OD_pairs_2015_Index(row)).Acft_Utilization_Ratio;');

        %eval(evalstring2);

        %Seats_assigned_to_each_acft_all(row).(['Year_',num2str(year)]) =
Trip_Distribution_All.(['Year_',num2str(year)]).Seats(row) .*
Acft_Distribution_2015(OD_pairs_2015_Index(row)).Acft_Utilization_Ratio;

        Seats_assigned_to_each_acft_all(row).(['Year_',num2str(year)]) =
Trip_Distribution_All.(['Year_',num2str(year)]).Seats(row) .*
OD_pairs_2015.AcftData(OD_pairs_2015_Index(row)).Ratio_Updated;
```

```
        %eval(strcat('seats_set = Seats_assigned_to_each_acft_all(row).Year_',num2str(year),';'));
        seats_set = Seats_assigned_to_each_acft_all(row).(['Year_',num2str(year)]);

        %number_of_acft = length(Acft_Distribution_2015(OD_pairs_2015_Index(row)).Unique_Acft_by_OD_pair);

        number_of_acft = length(acft_set);

        % wb2 = waitbar(0,'Flights Assignment...');

        for  acft = 1:number_of_acft

            %   waitbar(acft / number_of_acft);

            acft_index = find(strcmp(acft_set(acft),Aircraft_list_updated));

            %evalstring2 = strcat('Flights_by_each_acft_all(row).Year_',num2str(year),' = seats_set ./
Seats_by_Acft(acft_index);');
            %eval(evalstring2);
            Flights_by_each_acft_all(row).(['Year_',num2str(year)]) = seats_set ./ Seats_by_Acft(acft_index);

        end

        %  close (wb2)

      else %there's no od match

        %evalstring = strcat('Aircraft_Distribution_All(row).Year_',num2str(year),' = [];');
        %eval(evalstring)
        Aircraft_Distribution_All(row).Aircraft = [];

        %evalstring3 = strcat('Seats_assigned_to_each_acft_all(row).Year_',num2str(year),' =
Trip_Distribution_All.Year_',num2str(year),'.Seats(row);');
        %eval(evalstring3);
        Seats_assigned_to_each_acft_all(row).(['Year_',num2str(year)]) =
Trip_Distribution_All.(['Year_',num2str(year)]).Seats(row);
```

```
        %strcat('Flights_by_each_acft_all(row).Year_',num2str(year),' = 0;');
        Flights_by_each_acft_all(row).(['Year_',num2str(year)]) = 0;
    end
  end
  %close (wb1)
end

clear year row entries1 entries_dep entries2 entries_arr
%%

%
length_of_aircraft_distribution_all = length(Aircraft_Distribution_All);
%
m = 1;
for row = 1:length_of_aircraft_distribution_all
  if isempty(Aircraft_Distribution_All(row).Aircraft) == 1
    rows_to_delete(m,1) = row;
    m = m+1;
  end
end
disp('end')
%
check_if_variable_exist = exist('rows_to_delete');

if check_if_variable_exist ~= 0

  Aircraft_Distribution_All(rows_to_delete)       = [];
  Flights_by_each_acft_all(rows_to_delete)        = [];
  Seats_assigned_to_each_acft_all(rows_to_delete) = [];
  %
  for year = Start_Year:End_Year
    %
    disp(year)
    Trip_Distribution_All.(['Year_', num2str(year)]).Departure_Airport(rows_to_delete) = [];
    Trip_Distribution_All.(['Year_', num2str(year)]).Arrival_Airport (rows_to_delete)  = [];
```

```
        Trip_Distribution_All.(['Year_',num2str(year)]).Seats (rows_to_delete)            = [];
        %
    end
end

clear m year rows_to_delete length_of_aircraft_distribution_all

%%

save([Save_Dir, '\Trip_Distribution_All_Years.mat', 'Trip_Distribution_All']);

disp('Saving Aircraft_Distribution_All')
save([Save_Dir, '\Aircraft_Distribution_All.mat'], 'Aircraft_Distribution_All', '-v7.3');
disp('SAVED')

disp('Saving Seats_assigned_to_each_acft_all')
save([Save_Dir, '\Seats_assigned_to_each_acft_all.mat'], 'Seats_assigned_to_each_acft_all', '-v7.3');
disp('SAVED')

disp('Saving Flights_by_each_acft_all')
save([Save_Dir, '\Flights_by_each_acft_all.mat'], 'Flights_by_each_acft_all', '-v7.3');
disp('SAVED')

%%
% save('Trip_Distribution_All','Trip_Distribution_All')
%
%    disp('Saving Aircraft_Distribution_All')
%    save('Aircraft_Distribution_All','Aircraft_Distribution_All','-v7.3')
%    disp('SAVED')
%
%    disp('Saving Seats_assigned_to_each_acft_all')
%    save('Seats_assigned_to_each_acft_all','Seats_assigned_to_each_acft_all','-v7.3')
%    disp('SAVED')
%
%    disp('Saving Flights_by_each_acft_all')
```

```
%    save('Flights_by_each_acft_all','Flights_by_each_acft_all','-v7.3')
%    disp('SAVED')

return;
```

Fleet_Evolution.m

```
%Window%Main File to run FRATAR Model
function Fleet_Evolution(Install_Dir, Project_Dir, Case_Folder, Case_Name, Start_Year, End_Year)

clc;
warning off;

% ----------------------
% Settings if run inside MATALB
% ----------------------
% IMPORTANT: All inputs are now organized in "..\GDM_DATA\" folder. This
% the Git project GDM_DATA and contains the official inputs for this model
% for the GDM_GUI.
% Model runs are now saved in the following directory structure:
% Save_Dir = Project_Dir\Output\Case_Folder\Case_Name (See below)
%Scenario 1.5, 2 or 3 should be identify. Scenario 2 will use a base production
%rate for NASA's aircraft. Scenario 3 will use a higher production rate for
%NASA's aircraft. Scenario 1.5 will take into consideration only the
%introduction of N+1 vehicles
if isdeployed == 0
    Install_Dir = '..\GDM_DATA\DATA';
    Project_Dir = 'C:\GDM Project\Passenger';
    Case_Folder = 'Test_Folder';
    Case_Name   = 'Passenger Fleet Evolution';
    Scenario    = 2;
    % Trip Generation Output Folder
    TD_Case_Folder = 'Test_Folder';
    TD_Case_Name   = 'Passenger Trip Distribution';

    Start_Year = 2016;
    End_Year   = 2040;
end

% Create Output Dir
```

```
Save_Dir = [Project_Dir, '\Fleet_Evolution\Output\', Case_Folder, '\', Case_Name];
mkdir(Save_Dir);

Trip_Distribution_Dir = [Project_Dir, '\Trip_Distribution\Output\', TD_Case_Folder, '\', TD_Case_Name];

% ----------------------
% Retirement Replacemente Analysis
% ----------------------
Retirement_Replacement_Analysis(Install_Dir, Save_Dir, Trip_Distribution_Dir, Start_Year, End_Year)

% ----------------------
% Introduction of future aircraft to fleet
% ----------------------
Introduction_of_future_aircraft_fleet(Install_Dir, Save_Dir, Trip_Distribution_Dir, Start_Year, End_Year,Scenario)

% ----------------------
% Demand_Increase for NASA Aircraft
% ----------------------

%This function should be run only if additional demand wants to be added to
%NASA's aircraft only. The output won't affect previous results since the
%output of this function would be saved with a different variable name.
%How much additional per route demand would be assigned to NASA's Aircraft
%Route_demand_increase_by = 20; %Percent
%First_NASA_Aircraft_on_Aircraft_Under_Production_List = 15; %15th

%Demand_Increase_for_NASA_Aircraft(Install_Dir, Save_Dir, Route_demand_increase_by,
First_NASA_Aircraft_on_Aircraft_Under_Production_List,Start_Year, End_Year)


% ----------------------
% Fleet_Evolution_Validation
% ----------------------

%Fleet_Evolution_Validation(Install_Dir, Trip_Distribution_Dir, Save_Dir, Start_Year, End_Year)
return;
```

Retirment_Replacement_Analysis.m

```
function Retirement_Replacement_Analysis(Install_Dir, Save_Dir, Trip_Distribution_Dir, Start_Year, End_Year)

%Input files are located on server at: /Volumes/ATSL_partition1/ATSLData/Global
Demand/Input_Files_Feet_Evolution

%Create a Output folder for the results of the analysis
%mkdir ('C:\GDM_Git\Fleet_Evolution\Output')

 disp('Loading Files')
% %Loads all the inputs necessary to run the analysis
 load([Trip_Distribution_Dir, '\Aircraft_Distribution_All.mat'])
 load([Trip_Distribution_Dir, '\Flights_by_each_acft_all.mat'])
 load([Install_Dir, '\Passenger\Fleet_Evolution\Replacement_Aircraft.mat'])
 load([Install_Dir, '\Passenger\Fleet_Evolution\Replacement_Percentage.mat'])
 load([Install_Dir, '\Passenger\Fleet_Evolution\Retiring_Aircraft.mat'])
 load([Install_Dir, '\Passenger\Fleet_Evolution\Retiring_Percentage.mat'])
 disp('End loading files')

%%
%Calculates the length of Aircraft_Distribution_All which represents the...
%total number of OD pairs in the analysis
length_of_Aircraft_Distribution_All = length(Aircraft_Distribution_All);

disp('Generating Flight_by_year and Aircraft_by_year')

Flight_by_year = Flights_by_each_acft_all;

for year = Start_Year:End_Year

    for row = 1:length_of_Aircraft_Distribution_All

        Aircraft_by_year(row).(['Year_',num2str(year)])= Aircraft_Distribution_All(row).Aircraft;

    end %for row = 1:length_of_Aircraft_Distribution_All
```

```
end  %for year = Start_Year:End_Year

disp('Start Retirement/Replacement Analysis')

%Retiring and replacement starts in 2017 and end in 2040.
Replacement_Start_Year = Start_Year + 1;

%Number of aircraft that will be retiring
length_retiring_aircraft = length(Retiring_Aircraft);

%This for loop run the retirement/replacement analysis for each one
%of the aircraft that will be retiring.
for retiring_aircraft = 1:length_retiring_aircraft

    disp(['  Retiring Aircraft: ', num2str(retiring_aircraft), '/', num2str(length_retiring_aircraft)]);

    %An aircraft could be replace by one or more aircraft. max_replacement
    %tells whether there will be one or two replacement for a specific

    max_replacement =
length(Replacement_Percentage(retiring_aircraft).(['Year_',num2str(Replacement_Start_Year)]));

    %This for loop run the analysis for each OD pair in the netwokrk
    for OD = 1:length_of_Aircraft_Distribution_All

        %This for loop run the analysis from 2017 to 2040

        for year = Replacement_Start_Year:End_Year

            %For each OD the if function will check if the retiring aircraft is being used in the OD pairs that is being
analyzed
            %the results of the sum could be either 1 or 0.
            if sum(ismember(Aircraft_by_year(OD).(['Year_',num2str(year)]),Retiring_Aircraft(retiring_aircraft))) > 0

                %If the retiring aircraft is being used in the OD pair
                %being analysed the acft_match value will indicate the
```

```
            %position in which the retiring aircraft is located in the
            %Aircraft_by_year. The could be one or more aircraft for each OD pair
            acft_match =
find(ismember(Aircraft_by_year(OD).(['Year_',num2str(year)]),Retiring_Aircraft(retiring_aircraft)));

            %Indicates how many aircraft are being used in the OD pair
            %that is being analyzed
            lenght_of_aircraft_set = length(Aircraft_by_year(OD).(['Year_',num2str(year)]));

            %checks if the number of flights for a specific aircraft is
            %higher than 0
            if Flight_by_year(OD).(['Year_',num2str(year)])(acft_match) > 0

               %This for loop execute the analyzis for the total number of
               %replacements
               for set_of_replacement = 1:max_replacement %there could be one or two replacement

                  %The replacement aircraft is introduce to the network
                  Aircraft_by_year(OD).(['Year_',num2str(year)])(lenght_of_aircraft_set+set_of_replacement) =
Replacement_Aircraft(retiring_aircraft,set_of_replacement);

                  %The correspondign number of flights is assigned to the
                  %aircraft that is being used for replacement
                  Flight_by_year(OD).(['Year_',num2str(year)])(lenght_of_aircraft_set+set_of_replacement) =
sum(Flight_by_year(OD).(['Year_',num2str(year)])(acft_match)) *
Replacement_Percentage(retiring_aircraft).(['Year_',num2str(year)])(set_of_replacement);

               end %for set_of_replacement = 1:max_replacement

               %The number of flights of the retiring aircraft is reduced.
               Flight_by_year(OD).(['Year_',num2str(year)])(acft_match) =
Flight_by_year(OD).(['Year_',num2str(year)])(acft_match) *
Retiring_Percentage(retiring_aircraft).(['Year_',num2str(year)]);

            end % if Flight_by_year(OD).(['Year_',num2str(year)])(acft_match) > 0

         end %if sum(ismember(Aircraft_by_year(OD).(['Year_',num2str(year)]),Retiring_Aircraft(retiring_aircraft))) > 0
```

```
        end  %for year = Replacement_Start_Year:End_Year


    end  %for OD = 1:length_of_Aircraft_Distribution_All
end  %for retiring_aircraft = 1:length_retiring_aircraft
%%
for row = 1:length(Aircraft_by_year)
    if mod(row, 1000) == 0
        disp(row)
    end


    for year = Start_Year:End_Year


        Aircraft_by_year_Updated(row).(['Year_',num2str(year)]) =
unique(Aircraft_by_year(row).(['Year_',num2str(year)]));


    end %for year = Start_Year:End_Year
end %for row = 1:length_of_Aircraft_Distribution_All
%%
for row = 1:length(Aircraft_by_year)
    if mod(row, 1000) == 0
        disp(row)
    end


    for year = Start_Year:End_Year


        length_acft_set = length(Aircraft_by_year_Updated(row).(['Year_',num2str(year)]));


        for acft = 1:length_acft_set


            acft_match =
find(strcmp(Aircraft_by_year_Updated(row).(['Year_',num2str(year)])(acft),Aircraft_by_year(row).(['Year_',num2str(ye
ar)])));
            Flight_by_year_Updated(row).(['Year_',num2str(year)])(acft) =
sum(Flight_by_year(row).(['Year_',num2str(year)])(acft_match));
        end %for acft = 1:length_acft_set
    end %for year = Start_Year:End_Year
end %for row = 1:length_of_Aircraft_Distribution_All
```

```
%%
%Save the 2 outputs of the analysis%
save([Save_Dir, '\Aircraft_by_year_Updated.mat'], 'Aircraft_by_year_Updated');
save([Save_Dir, '\Flight_by_year_Updated.mat'],   'Flight_by_year_Updated');


return;
```

Introduction_of_future_aricraft.m

```
function Introduction_of_future_aircraft_fleet(Install_Dir, Save_Dir, Trip_Distribution_Dir, Start_Year,
End_Year,Scenario)

%Loads all the inputs necessary to run the analysis
disp('Loading Files')
load([Save_Dir, '\Aircraft_by_year_Updated.mat'])
load([Save_Dir, '\Flight_by_year_Updated.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\Aircraft_Under_Production.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\Similar_to_Aircraft_Under_Production.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\Introduction_Year.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\OD_Travel_Time.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\Aircraft_Production_Rate_Scenario_',num2str(Scenario),'.mat'])
load([Trip_Distribution_Dir, '\Seats_assigned_to_each_acft_all.mat'])
disp('End loading files')

%%
%Scenario 2 will assign new aircraft only to routes showed a growth greater
%than 1.5 between 2016 and 2040. Scenario 3 will neglate the growth of the
%route and will assign aircraft base only in the other 2 factors which are;
%the route must have a similar aircraft to the one being introduced and
%that the aircrft eing introduced has enough flight-hours availble to make
%the flight
if Scenario  == 1.5 || Scenario == 2

    selected_routes_with_agrowth_factor_greater_than = 1.5;

elseif Scenario == 3

    selected_routes_with_agrowth_factor_greater_than = 0;
end

%Identify the total number of aircraft to be introduce into the network
if Scenario == 2 || Scenario == 3
```

```
    length_of_aircraft_under_production = length(Aircraft_Under_Production);
elseif Scenario == 1.5

    length_of_aircraft_under_production = 14;
end

%Average number of hours an aircraft fly annually.
%Assumuing 10.35 hours per day and 351 days (365 days - 2 weeks for
%maintenance)
Aicraft_annual_flight_hours = 3630; %hours

%This for loop calculate the total number of hours an aircraft could fly
%per year by taking into consideration Aicraft_annual_flight_hours and Aircraft_Production_Rate
for aircraft_being_analyzed = 1:length_of_aircraft_under_production

    %Initial counter
    years = 0;

    for year = Introduction_Year(aircraft_being_analyzed):End_Year

        years = years + 1;

        Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) = Aicraft_annual_flight_hours .*
(Aircraft_Production_Rate(aircraft_being_analyzed)*years);

    end %for year = Introduction_Year(aircraft_being_analyzed):End_Year
end %for aircraft_being_analyzed = 1:length_of_aircraft_under_production
clear year

%Calculates the length of Aircraft_by_year which represents the...
%total number of OD pairs in the analysis
length_of_Aircraft_by_year_Updated = length(Aircraft_by_year_Updated);

%Pre-allocation for route-growth
route_growth = zeros(length_of_Aircraft_by_year_Updated,1);
```

```
%This for loop calculates the growth for each OD pair between 2016 and 2040
for row = 1:length_of_Aircraft_by_year_Updated

    route_growth(row) = sum(Seats_assigned_to_each_acft_all(row).(['Year_',num2str(End_Year)])) /
sum(Seats_assigned_to_each_acft_all(row).(['Year_',num2str(Start_Year)]));

end %for row = 1:length_of_Aircraft_by_year_Updated

%The for loop introduce each one of the aircraft in
%aircraft_under_prduction into the network
for aircraft_being_analyzed = 1:length_of_aircraft_under_production

    %Percentage that the new aircraft fleet will be replaceming from the
    %existing one (up to)
    if aircraft_being_analyzed >= 15

        Flight_replacement_percentage = 60; %60 percent

    else

        Flight_replacement_percentage = 40; %40 percent
    end

    disp(['  Analyzing Aircraft: ', num2str(aircraft_being_analyzed), '/', num2str(length_of_aircraft_under_production)]);

    %The for loop run the analysis for all the years

    for year = Introduction_Year(aircraft_being_analyzed):End_Year

        disp(['  Analyzing Year: ', num2str(year), '/', num2str(End_Year)]);

        %The for loop run the analysis for each OD pair
        for OD = 1:length_of_Aircraft_by_year_Updated

            %Check if the route_growth is higher than 1.5
            %New aircraft are introduce to heavily use routes only
```

```
if route_growth(OD,1) > selected_routes_with_agrowth_factor_greater_than

    %For each "aircraft_being_analyzed" identify how many
    %similar aircraft are
    if aircraft_being_analyzed < 15

        number_of_similar_aircraft = 1;

    elseif aircraft_being_analyzed == 15 || aircraft_being_analyzed == 19

        number_of_similar_aircraft = 7;

    elseif aircraft_being_analyzed == 16

        number_of_similar_aircraft = 4;

    elseif aircraft_being_analyzed == 17

        number_of_similar_aircraft = 2;

    elseif aircraft_being_analyzed == 18

        number_of_similar_aircraft = 5;

    end


    for similiar_aircraft = 1:number_of_similar_aircraft

        %For each OD the 'if' function will check if there is a similar
        %aircraft to the one beign produced for that
        %particular OD pair.
        %The results of the sum could be either 1 or 0.
        if
sum(ismember(Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]),Similar_to_Aircraft_Under_Production(aircraf
t_being_analyzed,similiar_aircraft))) > 0
```

```
            %indentify the location of the aircraft on the

            %aircraft_set

            location_on_aircraft_set =
find(strcmp(Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]),Similar_to_Aircraft_Under_Production(aircraft_b
eing_analyzed,similiar_aircraft)));


            %Indicates how many aircraft are being used in the OD pair

            %that is being analyzed

            lenght_of_aircraft_set = length(Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]));



            %Will inticate # of row

            counter = 1;


            %Will verify if the aircraft being analyzed has enough

            %filght-hours available

            if (Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)])) > 0


               %Preallocation

               check_Flights_assigned   = zeros(Flight_replacement_percentage,1);

               check_Hours_flew         = zeros(Flight_replacement_percentage,1);

               check_hours_to_substract = zeros(Flight_replacement_percentage,1);


               %The foor loop calculates flights and flight-hours

               %for 1% to Flight_replacement_percentage%

               for replacement_percentage = 1:Flight_replacement_percentage


                  check_Flights_assigned(counter,1) =
Flight_by_year_Updated(OD).(['Year_',num2str(year)])(location_on_aircraft_set) * (replacement_percentage/100);


                  check_Hours_flew(counter,1) = check_Flights_assigned(counter,1) * OD_Travel_Time(OD);


                  check_hours_to_substract(counter,1) =
check_Hours_flew(counter,1);%Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) -
Hours_flew;
```

```
            counter = counter+1;
        end %for replacement_percentage = 1:Flight_replacement_percentage

        %The for loop determine the maximum number of
        %flights that can be assigned
        for check_row = Flight_replacement_percentage:-1:1

            %Check if it's possible assign a specific
            %number of flight-hours to the aircraft being
            %analyzed
            if Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) -
check_hours_to_substract(check_row) > 0

                Flights_assigned = check_Flights_assigned(check_row);

                %Check if there is enough flights to be
                %deducted from the similar aircraft
                if (Flight_by_year_Updated(OD).(['Year_',num2str(year)])(location_on_aircraft_set) -
Flights_assigned) > 0

                    Hours_flew =  check_Hours_flew(check_row);

                    %Number of flights is assigned to
                    %aircraft, number of flights is
                    %deducted from similar aircraft and the
                    %new aircraft is added to the OD
                    Flight_by_year_Updated(OD).(['Year_',num2str(year)])(lenght_of_aircraft_set+1) =
Flights_assigned;
                    Flight_by_year_Updated(OD).(['Year_',num2str(year)])(location_on_aircraft_set) =
Flight_by_year_Updated(OD).(['Year_',num2str(year)])(location_on_aircraft_set) - Flights_assigned;
                    Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) =
Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) - Hours_flew;
                    Aircraft_by_year_Updated(OD).(['Year_',num2str(year)])(lenght_of_aircraft_set+1) =
Aircraft_Under_Production(aircraft_being_analyzed);

                    break %exit for check_row = Flight_replacement_percentage:-1:1
                end %if (Flight_by_year_Updated(OD).(['Year_',num2str(year)])(location_on_aircraft_set) -
Flights_assigned) > 0
```

```
                        end %if Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)]) -
check_hours_to_substract(check_row) > 0

                    end %for check_row = Flight_replacement_percentage:-1:1

                end %if (Total_flight_hours_by_aircraft(aircraft_being_analyzed).(['Year_',num2str(year)])) > 0

            end %if
sum(ismember(Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]),Similar_to_Aircraft_Under_Production(aircraf
t_being_analyzed))) > 0

        end %for similiar_aircraft = 1:number_of_similar_aircraft

    end %if route_growth(OD,1) > 2

end %for OD = 1:length_of_Aircraft_by_year_Updated

end %for year = Introduction_Year(aircraft_being_analyzed):End_Year


end %for aircraft_being_analyzed = 1:length_of_aircraft_under_production


%%

%The following for loops creates a unique aircraft set for each OD pair.

lenght_Aircraft_by_year_Updated = length(Aircraft_by_year_Updated);

for row = 1:lenght_Aircraft_by_year_Updated

    if mod(row, 1000) == 0

        disp(row)

    end


    for year = Start_Year:End_Year


        Aircraft_by_year_Updated_NASA(row).(['Year_',num2str(year)]) =
unique(Aircraft_by_year_Updated(row).(['Year_',num2str(year)]));


    end %for year = Start_Year:End_Year

end %for row = 1:length_of_Aircraft_Distribution_All

%%

length_Aircraft_by_year_Updated_NASA = length(Aircraft_by_year_Updated_NASA);

for row = 1:length_Aircraft_by_year_Updated_NASA

    if mod(row, 1000) == 0

        disp(row)

    end


    for year = Start_Year:End_Year
```

```
        length_acft_set = length(Aircraft_by_year_Updated_NASA(row).(['Year_',num2str(year)]));


        for acft = 1:length_acft_set


            acft_match =
find(strcmp(Aircraft_by_year_Updated_NASA(row).(['Year_',num2str(year)])(acft),Aircraft_by_year_Updated(row).(['Y
ear_',num2str(year)])));

            Flight_by_year_Updated_NASA(row).(['Year_',num2str(year)])(acft) =
sum(Flight_by_year_Updated(row).(['Year_',num2str(year)])(acft_match));

        end %for acft = 1:length_acft_set

    end %for year = Start_Year:End_Year

end %for row = 1:length_of_Aircraft_Distribution_All


%%

%Save the 3 outputs of the analysis%

save([Save_Dir, '\Aircraft_by_year_Updated_NASA_Scenario_',num2str(Scenario), '.mat'],
'Aircraft_by_year_Updated_NASA')

save([Save_Dir, '\Flight_by_year_Updated_NASA_Scenario_',num2str(Scenario), '.mat'],
'Flight_by_year_Updated_NASA')

save([Save_Dir, '\Total_flight_hours_by_aircraft_NASA_Scenario_',num2str(Scenario), '.mat'],
'Total_flight_hours_by_aircraft')

return;
```

Fleet_Evolution_Validation.m

```
function Fleet_Evolution_Validation(Install_Dir, Trip_Distribution_Dir, Save_Dir, Start_Year, End_Year)

%Loads all the inputs necessary to run the analysi
disp('Loading Files for Validation')
load([Save_Dir, '\Aircraft_by_year_Updated.mat'])
load([Save_Dir, '\Flight_by_year_Updated.mat'])
load([Save_Dir, '\Aircraft_by_year_Updated_NASA.mat'])
load([Save_Dir, '\Flight_by_year_Updated_NASA.mat'])
load([Trip_Distribution_Dir, '\Aircraft_Distribution_All.mat'])
load([Trip_Distribution_Dir, '\Flights_by_each_acft_all.mat'])
load([Install_Dir, '\Passenger\Fleet_Evolution\OD_Travel_Time.mat'])
disp('End loading files')

length_of_Aircraft_by_year_Updated = length(Aircraft_by_year_Updated);

disp('Generating Aircraft List')

row_counter = 1;

for year = Start_Year:End_Year

   disp(['Analyzing Year: ',num2str(year), '/' ,num2str(End_Year)])

   for OD = 1:length_of_Aircraft_by_year_Updated

      length_acft_set = length(Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]));

      for acft = 1:length_acft_set

         aircraft_list_all(row_counter,1) = Aircraft_by_year_Updated(OD).(['Year_',num2str(year)])(acft);
         row_counter = row_counter + 1;
      end
   end
```

```
end

unique_aircraft_list = unique(aircraft_list_all);
clear aircraft_list_all
%%

length_unique_aircraft_list = length(unique_aircraft_list);
Total_Number_of_Aircraft_After_Retirement_Replacement_Analysis = unique_aircraft_list;
Aicraft_annual_flight_hours = 3630; %hours

disp('Calculating NUmber of Flight and Aircraft per Year for each Aircraft')

for acft = 1:length_unique_aircraft_list

    disp(['  Analyzing Aircraft: ', num2str(acft), '/', num2str(length_unique_aircraft_list)]);

    for year = Start_Year:End_Year

        disp(['  Analyzing Year: ', num2str(year), '/', num2str(End_Year)]);

        flight_counter = zeros(1,1);
        row_counter = 1;
        for OD = 1:length_of_Aircraft_by_year_Updated
            if OD_Travel_Time(OD) == 9999999999

            else
                if sum(ismember(unique_aircraft_list(acft),Aircraft_by_year_Updated(OD).(['Year_',num2str(year)]))) > 0

                    acft_set_location =
find(strcmp(unique_aircraft_list(acft),Aircraft_by_year_Updated(OD).(['Year_',num2str(year)])));

                    flight_counter(row_counter,1) =
Flight_by_year_Updated(OD).(['Year_',num2str(year)])(acft_set_location);
                    flight_counter(row_counter,2) = OD_Travel_Time(OD);
                    row_counter = row_counter + 1;
                end
```

```
        end
    end

    if flight_counter(1,1) > 0

        Aicraft_Validation_Data_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights =
sum(flight_counter(:,1));
        Aicraft_Validation_Data_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft =
ceil(sum(flight_counter(:,1) .* flight_counter(:,2)) ./  Aicraft_annual_flight_hours);

    else

        Aicraft_Validation_Data_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights = 0;
        Aicraft_Validation_Data_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft = 0;

    end

  end
end

Aircraft_Table_RR  = zeros(length_unique_aircraft_list,(End_Year-Start_Year+1));

for acft = 1:length_unique_aircraft_list

  for year = Start_Year:End_Year

    years = year - (Start_Year-1);

    Aircraft_Table_RR(acft,years) =
Aicraft_Validation_Data_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft;
  end
end

save([Save_Dir, '\Aicraft_Validation_Data_RR.mat'], 'Aicraft_Validation_Data_RR');
save([Save_Dir, '\Total_Number_of_Aircraft_After_Retirement_Replacement_Analysis.mat'],
'Total_Number_of_Aircraft_After_Retirement_Replacement_Analysis');
save([Save_Dir, '\Aircraft_Table_RR.mat'], 'Aircraft_Table_RR');
```

```
%%
%

length_of_Aircraft_Distribution_All = length(Aircraft_Distribution_All);

disp('Generating Aircraft List')

row_counter = 1;

   for OD = 1:length_of_Aircraft_Distribution_All

      length_acft_set = length(Aircraft_Distribution_All(OD).Aircraft);

      for acft = 1:length_acft_set

         aircraft_list_all(row_counter,1) = Aircraft_Distribution_All(OD).Aircraft(acft);
         row_counter = row_counter + 1;
      end
   end


unique_aircraft_list = unique(aircraft_list_all);
%%

length_unique_aircraft_list = length(unique_aircraft_list);
Total_Number_of_Aircraft_Before_Retirement_Replacement_Analysis = unique_aircraft_list;

disp('Calculating NUmber of Flight and Aircraft per Year for each Aircraft')

for acft = 1:length_unique_aircraft_list

   disp(['  Analyzing Aircraft: ', num2str(acft), '/', num2str(length_unique_aircraft_list)]);

   for year = Start_Year:End_Year
```

```
    disp(['  Analyzing Year: ', num2str(year), '/', num2str(End_Year)]);


    flight_counter = zeros(1,1);
    row_counter = 1;
    for OD = 1:length_of_Aircraft_Distribution_All
       if OD_Travel_Time(OD) == 9999999999


       else
          if sum(ismember(unique_aircraft_list(acft),Aircraft_Distribution_All(OD).Aircraft)) > 0


             acft_set_location = find(strcmp(unique_aircraft_list(acft),Aircraft_Distribution_All(OD).Aircraft));


             flight_counter(row_counter,1) =
Flights_by_each_acft_all(OD).(['Year_',num2str(year)])(acft_set_location);
             flight_counter(row_counter,2) = OD_Travel_Time(OD);
             row_counter = row_counter + 1;
          end
       end
    end



    if flight_counter(1,1) > 0


       Aicraft_Validation_Data_Before_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights =
sum(flight_counter(:,1));
       Aicraft_Validation_Data_Before_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft =
ceil(sum(flight_counter(:,1) .* flight_counter(:,2)) ./  Aicraft_annual_flight_hours);


    else


       Aicraft_Validation_Data_Before_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights =
0;
       Aicraft_Validation_Data_Before_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft =
0;


    end
```

```
    end
end

Aircraft_Table_Before_RR  = zeros(length_unique_aircraft_list,(End_Year-Start_Year+1));

for acft = 1:length_unique_aircraft_list

    for year = Start_Year:End_Year

        years = year - (Start_Year-1);

        Aircraft_Table_Before_RR(acft,years) =
Aicraft_Validation_Data_Before_RR.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft;
    end
end

save([Save_Dir, '\Aicraft_Validation_Data_Before_RR.mat'], 'Aicraft_Validation_Data_Before_RR');
save([Save_Dir, '\Total_Number_of_Aircraft_Before_Retirement_Replacement_Analysis.mat'],
'Total_Number_of_Aircraft_Before_Retirement_Replacement_Analysis');
save([Save_Dir, '\Aircraft_Table_Before_RR.mat'], 'Aircraft_Table_Before_RR');

%%

length_of_Aircraft_by_year_Updated_NASA = length(Aircraft_by_year_Updated_NASA);

disp('Generating Aircraft List')

row_counter = 1;

for year = Start_Year:End_Year

    disp(['Analyzing Year: ',num2str(year), '/' ,num2str(End_Year)])

    for OD = 1:length_of_Aircraft_by_year_Updated_NASA
```

```
        length_acft_set = length(Aircraft_by_year_Updated_NASA(OD).(['Year_',num2str(year)]));

        for acft = 1:length_acft_set

            aircraft_list_all(row_counter,1) = Aircraft_by_year_Updated_NASA(OD).(['Year_',num2str(year)])(acft);
            row_counter = row_counter + 1;
        end
    end
end

unique_aircraft_list = unique(aircraft_list_all);
clear aircraft_list_all
%%

length_unique_aircraft_list = length(unique_aircraft_list);

Total_Number_of_Aircraft_NASA = unique_aircraft_list;

disp('Calculating NUmber of Flight and Aircraft per Year for each Aircraft')

for acft = 1:length_unique_aircraft_list

    disp(['  Analyzing Aircraft: ', num2str(acft), '/', num2str(length_unique_aircraft_list)]);

    for year = Start_Year:End_Year

        disp(['  Analyzing Year: ', num2str(year), '/', num2str(End_Year)]);

        flight_counter = zeros(1,1);
        row_counter = 1;
        for OD = 1:length_of_Aircraft_by_year_Updated_NASA
            if OD_Travel_Time(OD) == 9999999999

            else
                if sum(ismember(unique_aircraft_list(acft),Aircraft_by_year_Updated_NASA(OD).(['Year_',num2str(year)])))
> 0
```

```
                acft_set_location =
find(strcmp(unique_aircraft_list(acft),Aircraft_by_year_Updated_NASA(OD).(['Year_',num2str(year)])));


                flight_counter(row_counter,1) =
Flight_by_year_Updated_NASA(OD).(['Year_',num2str(year)])(acft_set_location);
                flight_counter(row_counter,2) = OD_Travel_Time(OD);
                row_counter = row_counter + 1;


            end
          end
       end



       if flight_counter(1,1) > 0


          Aicraft_Validation_Data_NASA.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights =
sum(flight_counter(:,1));
          Aicraft_Validation_Data_NASA.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft =
ceil(sum(flight_counter(:,1) .* flight_counter(:,2)) ./  Aicraft_annual_flight_hours);


       else


          Aicraft_Validation_Data_NASA.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Flights = 0;
          Aicraft_Validation_Data_NASA.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft = 0;


       end



   end
end



Aircraft_Table_NASA  = zeros(length_unique_aircraft_list,(End_Year-Start_Year+1));


for acft = 1:length_unique_aircraft_list
```

```
    for year = Start_Year:End_Year

        years = year - (Start_Year-1);

        Aircraft_Table_NASA(acft,years) =
Aicraft_Validation_Data_NASA.(['Aircraft_',num2str(acft)]).(['Year_',num2str(year)]).Number_of_Aircraft;
    end
end

save([Save_Dir, '\Aicraft_Validation_Data_NASA.mat'], 'Aicraft_Validation_Data_NASA');
save([Save_Dir, '\Total_Number_of_Aircraft_NASA.mat'], 'Total_Number_of_Aircraft_NASA');
save([Save_Dir, '\Aircraft_Table_NASA.mat'], 'Aircraft_Table_NASA');

return
```