Estimation of Runway Throughput with Reduced Wake Vortex Separation, Technical Buffer, and Runway Occupancy Time Considerations

Junqi Hu

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

July 31, 2018
Blacksburg, Virginia

Keywords: Technical Buffer, RECAT Separation, Simulation Model, Runway Occupancy Time, Go Around, Runway Capacity

Estimation of Runway Throughput with Reduced Wake Vortex Separation, Technical Buffer, and Runway Occupancy Time Considerations

Junqi Hu

ABSTRACT

This thesis evaluates the potential recovery of the runway throughputs under Wake Turbulence Re-categorization (RECAT) Phase II and Time-based Separation (TBS) with a ROT constraint comparing with RECAT Phase I. This research uses aircraft performance parameters (runway occupancy time, approach speed, etc.) from the Airport Surface Detection Equipment, Model X (ASDE-X) data set. The analysis uses a modified version of the Quick Response Runway Capacity Model (RUNSIM). The main contributions of the study are: 1) identifying the technical buffer between in-trail arrivals and regenerate them in RUNSIM; 2) estimate the percentage of the arrival pairs that have wake mitigation separation times in excess of ROT; 3) developed an additional in-trail arrival separation rule based on ROT; 4) measure the risk of potential go-arounds with and without the additional 95 ROT separation rules. 5) generate a sample equivalent time-based RECAT II

The study results show that the distributions of technical buffers have significant differences for different in-trail groups and strong connectivity to airport elevations. This is critical to estimate runway capacities and safety issues especially when advanced wake mitigation separation rules are applied. Also, with decreasing of wake separations, ROT will become a limiting factor in runway throughput in the future. This study shows that by considering a 95 percentile ROT constrain, one single runway can still obtain 4 or 5 more arrivals per hour under RECAT II but keep the same level of potential go-arounds compared with current operation rules (RECAT I). TBS rules seem to benefit more under strong wind conditions compared to RECAT I, and RECAT II. TBS rules need to be tailored to every airport.

Estimation of Runway Throughput with Reduced Wake Vortex Separation, Technical Buffer, and Runway Occupancy Time Considerations

Junqi Hu

GENERAL AUDIENCE ABSTRACT

This thesis evaluates the potential recovery of the runway throughputs by re-defining the minimum distance or time separations between successive arrivals. The minimum separation criteria between in-trrail arrivals is defined by Federal Aviation Administraiton to avoid the wake vortex influence produced by the leading aircraft. The main contribution of this thesis lies in estimation of throughput capacity with the reduced minimum separation between arrivals.

The study results show that the distributions of buffers added to the minimum separations have significant differences for different in-trail groups and strong connectivity to airport elevations. This is critical to estimate runway capacities and safety issues especially when advanced wake mitigation separation rules are applied. Also, with decreasing of wake separations, runway occupancy time will become a limiting factor in runway throughput in the future. This study shows that by considering a 95 percentile ROT constrain, one single runway can still obtain 4 or 5 more arrivals per hour under reduced minimum separation but keep the same level of potential go-arounds compared with current operation rules.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor, Dr. Antonio A. Trani, for his guidance and support during my M.S. degree study. It is a great honor to work with him in the Air Transportation Systems Laboratory. His knowledge and easy-going personality benefited me a lot. His great passion for aviation and a willingness to learn new things will influence and motivate me profoundly for the rest of my life.

My sincere gratitude to my committee members: Dr. Montasir M. Abbas and Dr. Susan Hotle for their patience and feedback on my thesis.

I am particularly thankful to Howard Swingle and Nicolas Hinze for their assistance in the research task. It has been a great experience working with these talented people in Wake Vortex project.

I also want to thank Navid Mirmohammadsadeghi for his support during the research especially. The success of the research could not be possible without a clean set of ASDE-X data provided by Mr. Mirmohammadsadeghi.

I also want to thank all my colleagues, Navid Mirmohammadsadeghi, Julio Roa, Arman Izadi, Edwin Freire Burgos, Bollempalli, Mani Bhargava Reddy, Rimjha, Mihir, Jungmin Seo, Stephanie Atallah, Yanqi Liang, Xi Duan, Kangyang Chen and Qichao Wang. It was a great pleasure working with all of you. Thank you for all the support and all the good times as well.

Last and most importantly, I must express my profound gratitude to my family for their unconditional support and continuous encouragement through my years of study. To my parents, thank you for trusting me all the time and letting me choose my career.

TABLE OF CONTENTS

1. Introduction 1

1.1 The objective of the study 1

1.2 NextGen and Re-Categorization 2

1.2.1 Problem Background 2

1.2.2 RE-Categorization Separation Applied at Different Airports 4

1.2.3 Time-based Separation 7

1.3 Thesis outline 8

2. Literature Review 10

2.1 In-trail Technical buffers 10

2.2 Runway Occupancy Time 11

2.3 Quick Response Runway Capacity Model (RUNSIM) 15

3. Determination of Current Arrival Separation Technical Buffers 16

3.1 Definition of technical buffers 16

3.2 Methodology to obtain technical buffers 17

3.2.1 Visibility Condition Factor 20

3.2.2 Identifying the Peak Period 20

3.2.3 Identifying the Final Approach Length 21

3.3 Buffer time Result for Several Main Airports 23

3.3.1 Buffer Time Comparison between Different Airports 23

3.3.2 Buffer Time Comparison between Different In-trail Groups 26

3.4 Re-generation of Buffer Time Distribution in RUNSIM 30

3.4.1 Statistic Tests 32

3.5 Converting factor between IMC and VMC 34

4. Runway Occupancy Time Constraint and Runway Throughput Estimation under RECAT II 38

4.1 Improvements to Quick Response Runway Capacity Model 38

4.1.1 Methodology to Estimate Single Runway Capacity in RUNSIM 38

4.2 Basic Inputs from ASDE-X Data for RUNSIM Simulation 41

4.2.1 Average Approach Speed per Aircraft Type 41

4.2.2 Runway Occupancy Time by Each Aircraft Type per Each Runway 43

4.3 ROT Operations Limited by ROT under RECAT I at ORD for IMC and VMC 44

4.4 RECAT II Runway Throughput With ROT Constraints at ORD and CLT 46

4.4.1 ROT limitations under RECAT II at ORD under IMC 46
4.4.2 The Suggestion of an Additional ROT-based Separation Rule....................... 47
4.4.3 ORD 28C, 27R, 27L 50
4.4.4 ORD 09L, 09R, 10C, 10R.. 53
4.4.5 ORD 22R... 56
4.4.6 ORD 28R, 10L... 58
4.4.7 CLT 18R, 18L, 18C 60
4.4.8 CLT 36L, 36R, 36C 63
4.4.9 CLT 23 66
4.5 Runway Throughput with Additional 95 Percentile ROT Separation Rule under RECAT I and RECAT II 68
5. Potential Go-around Estimation under RECAT II... 71
5.1 Methodology to Estimate Potential Go-Around under RECAT II.................. 71
5.2 Potential Go-arounds Estimation with and without Additional ROT Separation Rule under RECAT II for Several Hub Airports............................... 72
6. The Incremental Benefits of RECAT II and TBS under Wind Conditions at ORD 78
6.1 Wind Data at ORD 78
6.2 Potential Gains from TBS and RECAT II under Wind Conditions................ 79
6.3 Sample time-based RECAT II separation at ORD 82
7. Conclusions and Recommendations 84
REFERENCE... 86
Appendex 1 88

LIST OF FIGURES

Figure 1 Boeing 737-800 Runway Occupancy Time Cumulative Density Function plot (Trani, 2017). 3
Figure 2 Potential Runway Throughput Benefits of Time-Based Separation Standards Applied to a Busy Airport in the U.S. (Trani, 2017). 4
Figure 3 Buffer Added to the Minimum Distance Separation (C.L.V.Driessen 2015). 10
Figure 4 Elements of Runway Occupancy during the Landing Procedure {Trani, 1992 #35} 12
Figure 5 Three main definitions of Runway Occupancy Time (Trani, 2017). 14
Figure 6 B738 Runway Occupancy Times for Tree Methods (Trani, 2017). 14
Figure 7 Graphic User Interface in RUNSIM Version 2.0.1 (Trani et al. 2018). 16
Figure 8 Flowchart to Obtain Buffer Time-based on ASDE-X Data. 18
Figure 9 Methodology to Identify Minimum Distance between In-trail Arrivals, Buffer Distance, and Buffer Time. 19
Figure 10 Sample Histogram Plot of Arrivals per 15 Minutes on Runway 10C in ORD. 21
Figure 11 Sample Cumulative Distance – Course Plot for Sample Arrival Operation at ORD. 22
Figure 12 Sample Approach Length Distribution for Runway 10C at ORD. 22
Figure 13 Buffer Time Distribution during IMC at ORD, CLT, ATL, and DEN. 24
Figure 14 Cumulative Density Plot of All Buffer Times at ORD, CLT, ATL, and DEN. 25
Figure 15 Cumulative Density Plot of Buffer Time for DD, DE, ED, EE groups during IMC at ORD. 27
Figure 16 Cumulative Density Plot of Buffer Times for DD, DE, ED, EE groups during IMC at CLT. 27
Figure 17 Box-plot of Buffer Time Distribution for ED, EE, DD, and DE In-trail Groups at ORD. 28
Figure 18 Cumulative Density Distribution of Buffer Time for All In-trail Groups with More Than 100 Data Sets during IMC at ORD. 29
Figure 19 Comparison of Kernel distribution and normal distribution with real data for in-trail arrivals D-D group at ORD. 31
Figure 20 Cumulative distribution function plot for 10,000 random number from the simulation for each in-trail arrival group at ORD 31
Figure 21 Comparison of box plot between real data and simulated data for D-D group at ORD 33
Figure 22 Comparison of box plot between real data and simulated data for E-F group at ORD 34
Figure 23 In-trail Time Interval Kernel Distribution during VMC and IMC at ORD 35
Figure 24 In-trail Time Interval Kernel Distribution during VMC and IMC at CLT 35
Figure 25 In-trail Time Interval Kernel Distribution during VMC and IMC at DEN 35

Figure 26 In-trail Time Interval Kernel Distribution during VMC and IMC at ATL 36
Figure 27 In-trail Time Interval Kernel Distribution during VMC and IMC at LGA 36
Figure 28 In-trail Time Interval Kernel Distribution during VMC and IMC at MCO 36
Figure 29 In-trail Time Interval Kernel Distribution during VMC and IMC at IAD 37
Figure 30 In-trail Time Interval Kernel Distribution during VMC and IMC at DCA 37
Figure 31 "Opening" and "Closing" Cases during the Final Approach 39
Figure 32 Linear Regression between Average Approach Speed for Final Approach and Last 2.5 nm for 33 Aircraft Types at ORD 42
Figure 33 Linear Regression between Average Approach Speed for Final Approach and Last 2.5 nm for 33 Aircraft Types at CLT 42
Figure 34 Normalized Runway Occupancy Time Histogram plot for Aircraft A320 at Runway 9L, 10C, 10R, and 10L 43
Figure 35 Runway Configurations at ORD (Source: Airnav.com). 44
Figure 36 Flag Indicator is introduced to Record the Cases when ROT Limits the Runway Throughput in RUNSIM 45
Figure 37 The ROT Limiting Percentage under RECAT I and RECAT II with the Kernel and Normal Distributed Buffer Time for ORD 47
Figure 38 The Methodology of the Suggested Separation Rule with the Additional ROT Separation Rule Consideration 48
Figure 39 Sample 90 and 95 Percentile ROT value for A320 at ORD Runway 10C Based on the Year 2016 ASDE-X Data 49
Figure 40 Modified Flag Indicator under the Suggested Separation Rule 49
Figure 41 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 28C 50
Figure 42 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 27R 51
Figure 43 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 27L 52
Figure 44 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 09L 53
Figure 45 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10C 54
Figure 46 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10R 55
Figure 47 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 09R 56
Figure 48 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 22R 57
Figure 49 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 28R 58

Figure 50 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10L 59
Figure 51 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18R. 60
Figure 52 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18L. 61
Figure 53 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18C. 62
Figure 54 Runway Configurations at CLT (Source: Airnav.com.) 63
Figure 55 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36L. 64
Figure 56 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36R. 65
Figure 57 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36C. 66
Figure 58 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 23. 67
Figure 59 Runway Throughputs With and Without 95 Percentile ROT Separation Rule at ORD. 68
Figure 60 Runway Throughputs With and Without 95 Percentile ROT Separation Rule at CLT. 69
Figure 61 Runway Throughput Comparison between RECAT II and RECAT I with Additional 95 Percentile ROT Separation Rule at ORD. 70
Figure 62 Runway Throughput Comparison between RECAT II and RECAT I with Additional 95 Percentile ROT Separation Rule at CLT. 70
Figure 63 Methodology for Estimating the Time Interval between In-trail Arrivals under RECAT II. 71
Figure 64 Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT I with only Wake Vortex Separation Rule at ORD. 73
Figure 65 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with Only Wake Vortex Separation Rule at ORD. 74
Figure 66 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with Only Wake Vortex Separation Rule at ORD 27L. 74
Figure 67 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with an Additional 95 Percentile ROT Separation Rule at ORD 27L. 75
Figure 68 Sample Headwinds for Runway End 27R at ORD 78
Figure 69 Potential Benefits from RECAT II and TBS under Wind Conditions for West Flow (Runway 28C, 27R, and 27L) at ORD. 80
Figure 70 Potential Benefits from RECAT II and TBS under Wind Conditions for East Flow (Runway 09L, 28R, and 10L) at ORD. 81
Figure 71 Potential Benefits from RECAT II and TBS under Wind Conditions for Runway end 22R at ORD. 81

LIST OF TABLES

Table 1 Separation Matrix for Legacy Aircraft Groups (at the threshold) (FAA 2013b). 5
Table 2 Separation Matrix for RECAT I (FAA 2016). 5
Table 3 RECAT II Aircraft Groups for South California TRACON (FAA 2016). 6
Table 4 Airports that have Implemented Wake Re-Categorization Phase I (FAA 2016) 6
Table 5 RECAT Phase II Locations/Dates (FAA 2016). 7
Table 6 Heathrow Reference Time-based Separations (values in seconds) (Morris and Choroba 2013). 8
Table 7 Approach Speed for Different Wake Group Aircrafts at ORD, CLT, ATL, and DEN. 26
Table 8 Bartlett's Statistic Result Table between Buffer Time and Critical In-trail Groups at ORD. 28
Table 9 Similar Mean and Standard Deviation for Buffer Times between DD, ED, EE, DE in-trail groups at ORD, CLT and ATL (airports below 2.500 feet). 30
Table 10 Kruskal-Wallis ANOVA Table for real data and simulation data for D-D in-trail arrival group at ORD 33
Table 11 Kruskal-Wallis ANOVA Table for real data and simulation data for E-F in-trail arrival group at ORD 33
Table 12 Sample Inputs for Runway 10C at ORD 41
Table 13 RUNSIM Simulation Results with RECAT I Separation for the Main Runway Ends at ORD 45
Table 14 Inputs from ASDE-X data for ORD 28C 50
Table 15 Inputs from ASDE-X data for ORD 27R 51
Table 16 Inputs from ASDE-X Data for ORD 27L 51
Table 17 Inputs from ASDE-X Data for ORD 09L 53
Table 18 Inputs from ASDE-X Data for ORD 10C 54
Table 19 Inputs from ASDE-X Data for ORD 10R 54
Table 20 Inputs from ASDE-X Data for ORD 09R 55
Table 21 Inputs from ASDE-X Data for Runway 22R 57
Table 22 Inputs from ASDE-X Data for ORD 28R 58
Table 23 Inputs from ASDE-X Data for ORD 10L 59
Table 24 Inputs from ASDE-X Data for CLT 18R. 60
Table 25 Inputs from ASDE-X Data for CLT 18L. 61
Table 26 Inputs from ASDE-X Data for CLT 18C. 61
Table 27 Inputs from ASDE-X Data for CLT 36L. 63
Table 28 Inputs from ASDE-X Data for CLT 36R. 64
Table 29 Inputs from ASDE-X Data for CLT 36C. 65
Table 30 Inputs from ASDE-X Data for CLT 23. 67
Table 31 Samples of In-trail Arrival Information and Estimated in-trail time under RECAT II. 72
Table 32 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at ORD. 75

Table 33 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at CLT. 76
Table 34 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at DEN. 76
Table 35 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at ATL. 77
Table 36 Wind Conditions for West Flow (Runway 27L, 27R, 28C, and 28R) at ORD. 79
Table 37 Wind Conditions for East Flow (Runway 09L, 09R, 10L and 10R) at ORD. 79
Table 38 Wind Conditions for Runway 22R at ORD. 79
Table 39 Typical Aircrafts for Each Wake Group Operates at ORD. 82
Table 40 Distance-based RECAT II Separation for Typical Aircrafts Operates at ORD. 82
Table 41 Sample Time-based RECAT II Separation Matrix for Typical Aircraft Types at ORD. 83
Table 42 Sample TBS Matrix for Typical Aircraft Types at ORD. 83

1. Introduction

1.1 The objective of the study

The objective of this study is to assess the potential airport landing capacity improvement that can be achieved by the implementation of wake turbulence Re-categorization separation Phase II (RECAT II) and Time-based Separation with technical buffer and runway occupancy time considerations. This study uses detailed analyzing aircraft landing data (approach speed, the time interval between arrivals, runway occupancy time, etc.) from the ASDE-X (Airport Surface Detection Equipment Model X) equipment installed at 37 U.S. airports. ASDE-X data is used as input for estimating runway capacity and potential go-arounds under the different separation rules and weather conditions. This study is conducted using a Monte Carlo computer simulation. The analysis results provide an evaluation of the airport throughput gains with new separation criteria called RECAT II and Time-based Separation. The analysis considers newly derived technical buffers and runway occupancy times as constraints.

ASDE-X data is used to obtain technical buffer times between in-trail arrivals, runway occupancy time for each aircraft type for different runways and aircraft approach speeds. The study uses a Monte Carlo simulation model named Quick Response Runway Capacity Model (RUNSIM). RUNSIM is an airport capacity estimation tool developed at the Virginia Tech Air Transportation Systems Laboratory. RUNSIM can estimate the runway and airport capacity based on different separation rules, airport configurations, different fleet mixes and aircraft trajectory performance parameters (mean value and standard deviation of runway occupancy time and approach speed, minimum arrival, and departure runway length, etc.) for different visibilities and weather conditions.

RUNSIM was expended during this work to track the number of times runway occupancy time would limit the runway throughput during peak period with RECAT II wake separation criteria. An additional separation rule based on runway occupancy time is suggested to reduce the potential conflicts between the aircraft runway occupancy time and the reduced RECAT II wake separation plus buffer, which is expected to lead to fewer go-arounds.

The potential go-arounds with and without the additional runway occupancy time separation rule for ORD, CLT and ATL under RECAT II are analyzed by comparing the runway occupancy times and estimated in-trail time intervals between arrivals under RECAT II. The aircraft performance data derived from ASDE-X can be used in future capacity improvement analysis.

1.2 NextGen and Re-Categorization

The Next Generation Air Transportation System, commonly known as NextGen, "is the transformation of how airplanes traverse the sky," aims to modernize the nation's air transportation system (Boehlert and Calvert 2006). According to the NextGen plan, the satellite-based technology surveillance system will replace radar technology in the future. The different types of navigation equipment will enhance the aircraft separation assurance and thus improve the separations between in-trail aircraft and wake performance capacities (FAA 2016).

1.2.1 Problem Background

The FAA NextGen development initiatives plan to increase the capacity of the National Airspace System (NAS). With the increasing demand in air traffic and limited land at current airports, NextGen addresses the need to increase the capacity for more flights through air corridors and into an airport without having to fly excessive distance or to build more runways. The FAA Wake Turbulence Research Program (WTRP) has been and is continuing to develop reduced wake vortex separations that safely allow greater throughput in the NAS. The application of wake turbulence Re-categorization separation Phase I (RECAT I) standards at Memphis International Airport in November 2012 eliminates the delays in FedEx departures and continues to provide benefits. Since that time, RECAT Phase I, RECAT Phase II, and their derivative distance-based wake separation standards have been implemented at ATC operations at 16 TRACONS and 29 airports, resulting in increased runway throughput as those airports (Cheng, Tittsworth et al. 2016).

The throughput gains from Wake Turbulence Categorization Phase I/II development efforts was predicted by earlier NEXTOR II modeling work (Mirmohammadsadeghi and Trani 2017). These gains in throughput were achieved with wake risk mitigation solutions that only required changes in the wake separation distance minima (RECAT I vs. RECAT II) that ATC applied between arrival aircraft. However, the research shows that the ROT will exceed the equivalent in-trail time separations when applying RECAT II. For example, when a Boeing 737-800 following a Boeing 737-800, from the runway occupancy time Cumulative Density Function (CDF) plot as shown in Figure 1, 20% of time ROT could exceed in-trail separation. (Mirmohammadsadeghi and Trani 2017)

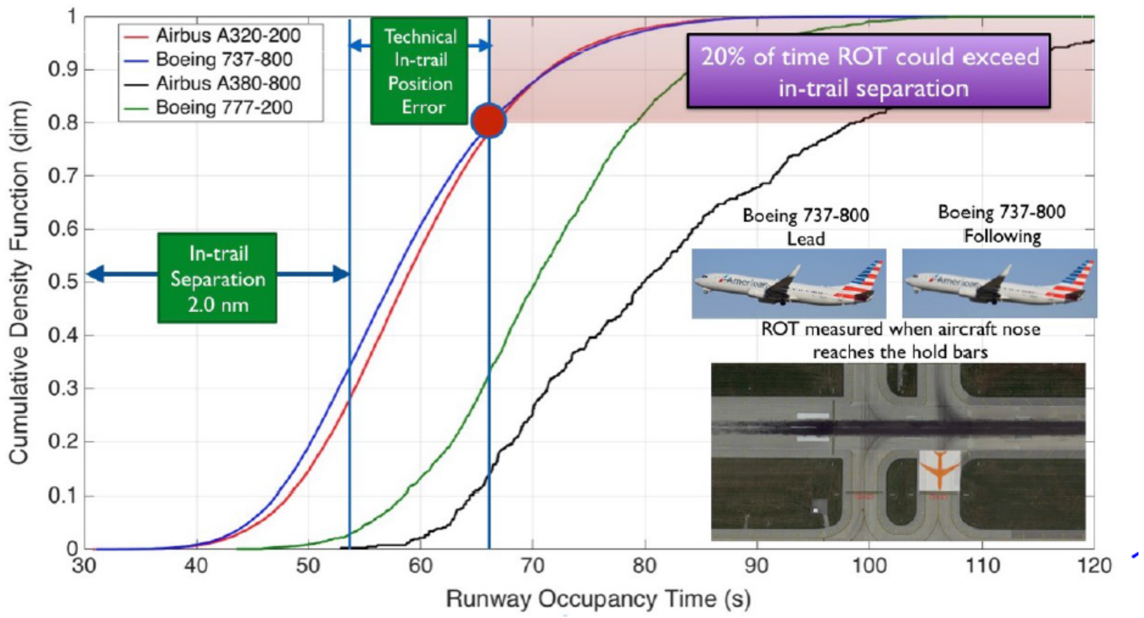


Figure 1 Boeing 737-800 Runway Occupancy Time Cumulative Density Function plot (Trani, 2017).

The London Heathrow (LHR) implementation of time-based wake mitigation separations allow ATC to manage the separations based on the actual time between in-trail aircraft. If headwind is present during the approach, the actual distance between aircraft is less than if there was no headwind. Figure 2 illustrates some early results of the time-based separation analysis performed using NEXTOR II tools for one large airport in the US. (Mirmohammadsadeghi and Trani 2017)

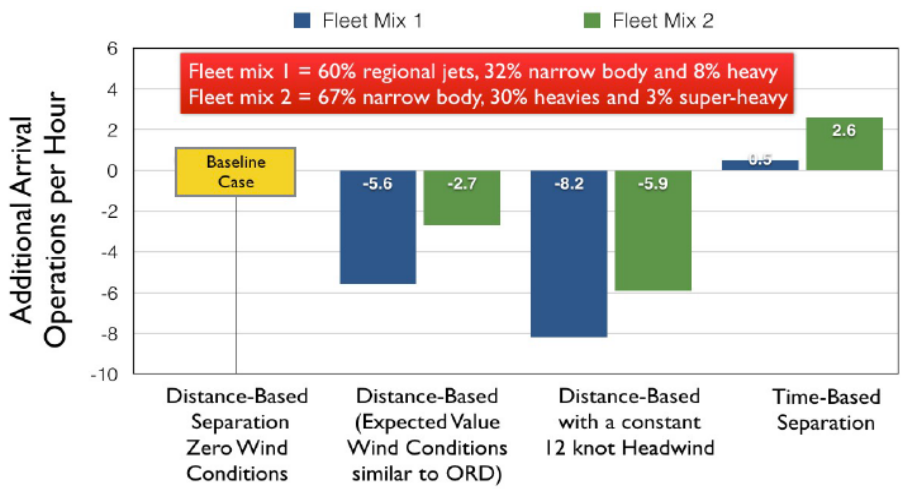


Figure 2 Potential Runway Throughput Benefits of Time-Based Separation Standards Applied to a Busy Airport in the U.S. (Trani, 2017).

1.2.2 RE-Categorization Separation Applied at Different Airports

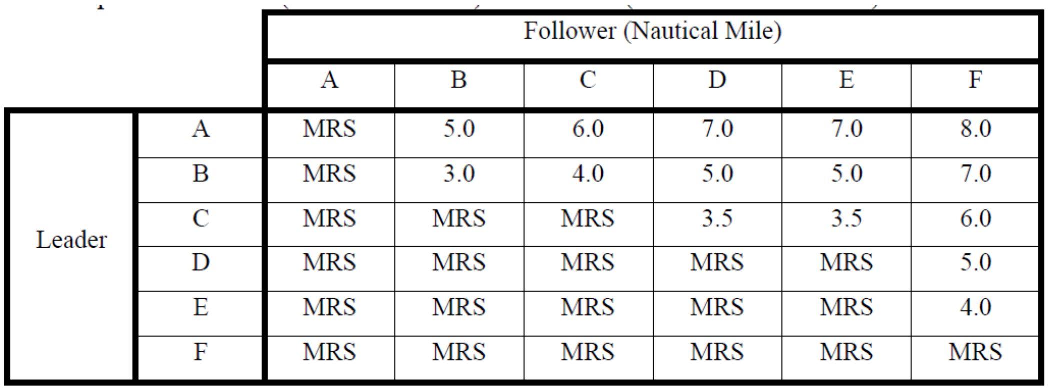
Wake Turbulence Re-Categorization, or commonly referred as RECAT, is an ongoing three-phase effort by the FAA to revise the current aircraft in-trail separation standards. Legacy in-trail separation standards are based primarily on weight with five wake categories published by FAA (FAA 2013b) as shown in Table 1. RECAT Phase I standards have been published and improved since 2011, replacing the old five wake groups with six aircraft groups (FAA 2016), as shown in Table 2. RECAT Phase II is a location-specific, static pairwise categorization based on 123 aircraft type verified by International Civil Aviation Organization and representing 99% of the U.S. commercial traffic from 32 major U.S. airports (FAA 2016) as shown in Table 3. Table 4 shows the list of airports that are applying the RECAT Phase I according to the NextGen Portfolio (FAA 2016). RECAT Phase II is TRACON-centric with participating major and satellite airports (FAA 2016) as shown in Table 5. Each TRACON has a specific aircraft fleet mix and associated separation matrix and may consist of more than the six (6) fixed RECAT Phase I categories. The practical implementation of complete RECAT II matrix (123 aircraft by 123 aircraft) to other airports requires further study. Air traffic controllers need a decision-support tool to help them recall individual separation rules among a large number of aircraft.

Table 1 Separation Matrix for Legacy Aircraft Groups (at the threshold) (FAA 2013b).

		Follower (Nautical Mile)				
		Superheavy	Heavy	B757	Large	Small
Leader	Superheavy	2.5*	6	7	7	8
	Heavy	2.5*	4	5	5	6
	B757	2.5*	4	4	4	5
	Large	2.5*	2.5*	2.5*	2.5*	4
	Small	2.5*	2.5*	2.5*	2.5*	2.5*

* MRS: Minimum Radar Separation (3 NM, or 2.5 NM when existing requirements are met)

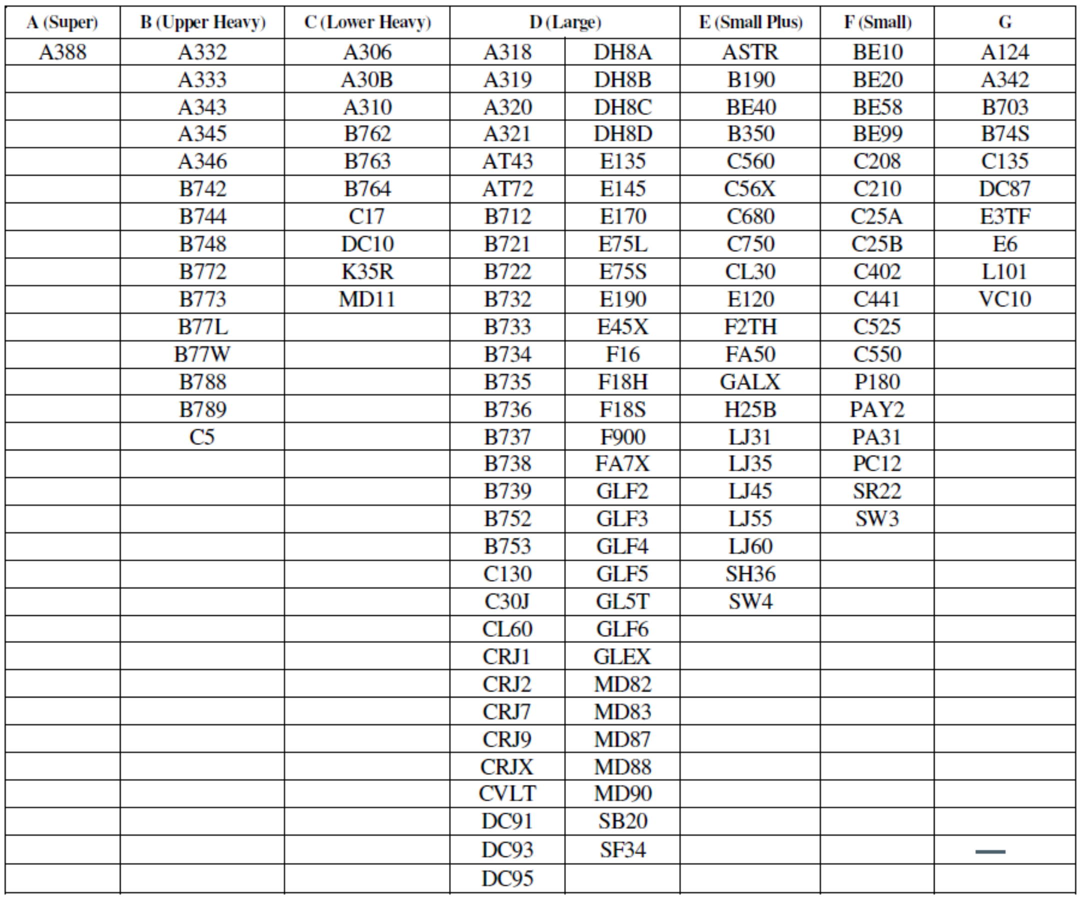
Table 2 Separation Matrix for RECAT I (FAA 2016).



		Follower (Nautical Mile)					
		A	B	C	D	E	F
Leader	A	MRS	5.0	6.0	7.0	7.0	8.0
	B	MRS	3.0	4.0	5.0	5.0	7.0
	C	MRS	MRS	MRS	3.5	3.5	6.0
	D	MRS	MRS	MRS	MRS	MRS	5.0
	E	MRS	MRS	MRS	MRS	MRS	4.0
	F	MRS	MRS	MRS	MRS	MRS	MRS

* MRS: Minimum Radar Separation (3 NM, or 2.5 NM when existing requirements are met)

Table 3 RECAT II Aircraft Groups for South California TRACON (FAA 2016).



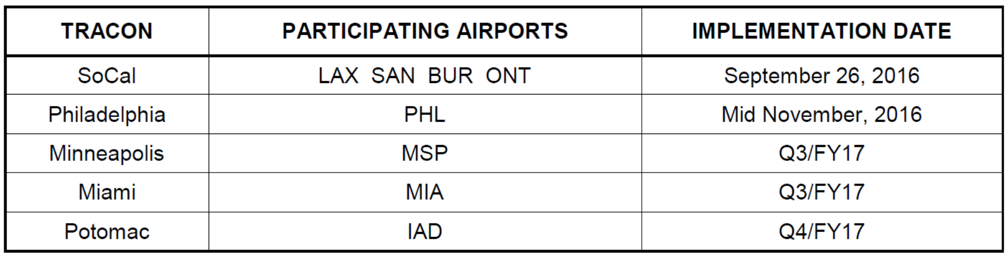
A (Super)	B (Upper Heavy)	C (Lower Heavy)	D (Large)		E (Small Plus)	F (Small)	G
A388	A332	A306	A318	DH8A	ASTR	BE10	A124
	A333	A30B	A319	DH8B	B190	BE20	A342
	A343	A310	A320	DH8C	BE40	BE58	B703
	A345	B762	A321	DH8D	B350	BE99	B74S
	A346	B763	AT43	E135	C560	C208	C135
	B742	B764	AT72	E145	C56X	C210	DC87
	B744	C17	B712	E170	C680	C25A	E3TF
	B748	DC10	B721	E75L	C750	C25B	E6
	B772	K35R	B722	E75S	CL30	C402	L101
	B773	MD11	B732	E190	E120	C441	VC10
	B77L		B733	E45X	F2TH	C525	
	B77W		B734	F16	FA50	C550	
	B788		B735	F18H	GALX	P180	
	B789		B736	F18S	H25B	PAY2	
	C5		B737	F900	LJ31	PA31	
			B738	FA7X	LJ35	PC12	
			B739	GLF2	LJ45	SR22	
			B752	GLF3	LJ55	SW3	
			B753	GLF4	LJ60		
			C130	GLF5	SH36		
			C30J	GL5T	SW4		
			CL60	GLF6			
			CRJ1	GLEX			
			CRJ2	MD82			
			CRJ7	MD83			
			CRJ9	MD87			
			CRJX	MD88			
			CVLT	MD90			
			DC91	SB20			
			DC93	SF34			—
			DC95				

Table 4 Airports that have Implemented Wake Re-Categorization Phase I (FAA 2016).

Location Abbreviation	Location Name
ANC	Ted Stevens Anchorage International Airport
ATL	Hartsfield-Jackson Atlanta International Airport
CLT	Charlotte Douglas International Airport
CVG	Cincinnati/Northern Kentucky International Airport
DEN	Denver International Airport
EWR	Newark Liberty International Airport
HOU	William P. Hobby Airport
HPN	Westchester County Airport
IAH	Houston - George Bush Intercontinental Airport

IND	Indianapolis International Airport
ISP	Long Island MacArthur Airport
JFK	New York - John F. Kennedy International Airport
LGA	New York - LaGuardia Airport
MDW	Chicago Midway International Airport
MEM	Memphis International Airport
OAK	Oakland International Airport
ORD	Chicago O'Hare International Airport
SDF	Louisville International-Standiford Field
SFO	San Francisco International Airport
SJC	Norman Y. Mineta San Jose International Airport
SMF	Sacramento International Airport
TEB	Teterboro Airport
Information as of May 11.2016	

Table 5 RECAT Phase II Locations/Dates (FAA 2016).



TRACON	PARTICIPATING AIRPORTS	IMPLEMENTATION DATE
SoCal	LAX SAN BUR ONT	September 26, 2016
Philadelphia	PHL	Mid November, 2016
Minneapolis	MSP	Q3/FY17
Miami	MIA	Q3/FY17
Potomac	IAD	Q4/FY17

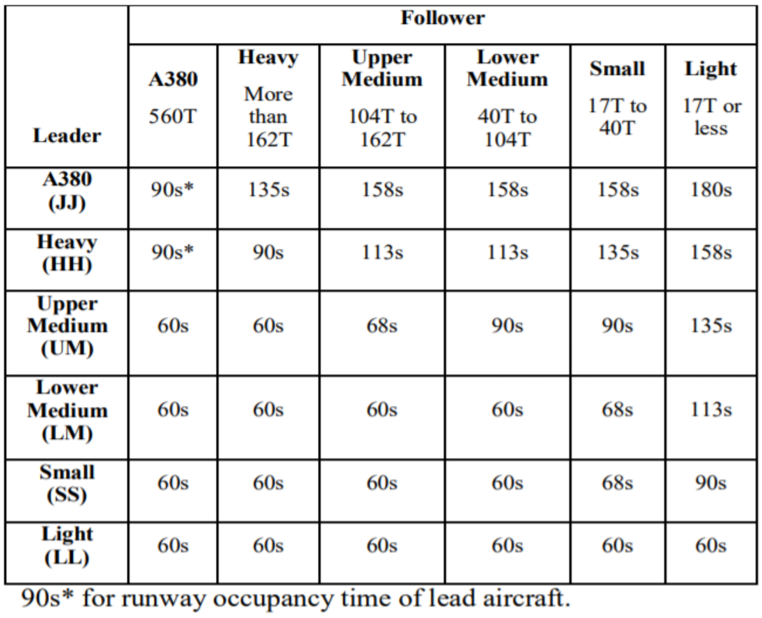
1.2.3 Time-based Separation

When there are strong headwinds, the aircraft ground speed is reduced on final approach, which leads to a reduced landing rate. A Time-based Separation (TBS) was developed by EUROCONTROL aiming at reducing the gap between arrival aircraft to maintain the airport capacity in light and strong wind conditions (EUROCONTROL 2014).

London Heathrow (LHR) started applying TBS in the year 2015 and achieved 40 to 45 landings per hour in calm winds, depending on the mix of arrivals. Before applying

TBS, the landing rate fell between 32 to 36 landings per hour in strong headwinds, which resulted in delays, increased holding and even cancellations. This research uses the time-based separation matrix to study the potential runway throughput gains with TBS under wind conditions at some major airports in the U.S. The LHR-TBS matrix is shown in Table 6 (Morris and Choroba 2013).

Table 6 Heathrow Reference Time-based Separations (values in seconds) (Morris and Choroba 2013).



	Follower					
Leader	**A380** 560T	**Heavy** More than 162T	**Upper Medium** 104T to 162T	**Lower Medium** 40T to 104T	**Small** 17T to 40T	**Light** 17T or less
A380 (JJ)	90s*	135s	158s	158s	158s	180s
Heavy (HH)	90s*	90s	113s	113s	135s	158s
Upper Medium (UM)	60s	60s	68s	90s	90s	135s
Lower Medium (LM)	60s	60s	60s	60s	68s	113s
Small (SS)	60s	60s	60s	60s	68s	90s
Light (LL)	60s	60s	60s	60s	60s	60s

90s* for runway occupancy time of lead aircraft.

1.3 Thesis outline

The thesis begins with an overview of the objective of the study and the background and challenges of RE-Categorization. Since the target for this thesis is to assess the feasibility of applying RE-Categorization in the future, some key factors are identified such as buffer time and runway occupancy time. Chapter 2 summarizes previous studies on buffer time and runway occupancy time and introduces the Quick Response Runway Capacity Model (RUNSIM), which is used to estimate the potential benefits of re-categorization.

Chapter 3 is a detailed study of the technical buffers added to the minimum separations between arrivals during the peak period. The chapter looks at buffer times for each in-trail arrival groups by analyzing the Airport Surface Detection Equipment Model

– X (ASDE-X) data. The research evaluates multiple runways and the influence of buffer times in detail.

Chapter 4 evaluates the runway throughputs considering runway occupancy time constraints under different separation rules by using RUNSIM. The results of the simulation quantify the potential gains in runway throughput, but also identify problems under reduced separation rules. Some recommendations are also given to apply the reduced separation to deal with the potential runway occupancy time limits.

Chapter 5 provides an essential method to estimate the potential go-arounds with reduced separations at Chicago O'Hare Airport, Charlotte Douglas International Airport, and Hartsfield–Jackson Atlanta International Airport. The analysis shows a feasible way to decrease the potential number of go-arounds caused by runway occupancy time with reduced separations.

Chapter 6 estimates the potential gains by applying a time-based reduced separation considering headwind effects. A sample equivalent time-based RECAT II matrix is also generated and compared with the time-based separation applied at London Heathrow Airport.

Chapter 7 summarizes the findings of the study and makes recommendations for future research.

2. Literature Review

2.1 In-trail Technical buffers

Wake turbulence is one of the main reasons that limit runway capacity for most of the airports in both the U.S. and Europe. The NASA and FAA first realized wake turbulence as an operational problem back in the early 1970s when Boeing 747-100 came to service (Gerz, Holzäpfel et al. 2002). Since then, NASA and FAA developed sensors to collect operational data to measure wake behavior and started wake mitigation projects (Fan and Trani 2014). The minimum separation between in-trail arrivals is applied to avoid wake-vortex encounters. To deal with system error caused by Air Traffic Control (ATC) facilities and human behavior error caused by pilots, ATC controllers consider a buffer above the minimum separation based on experience to ensure minimum separation as shown in Figure 3.

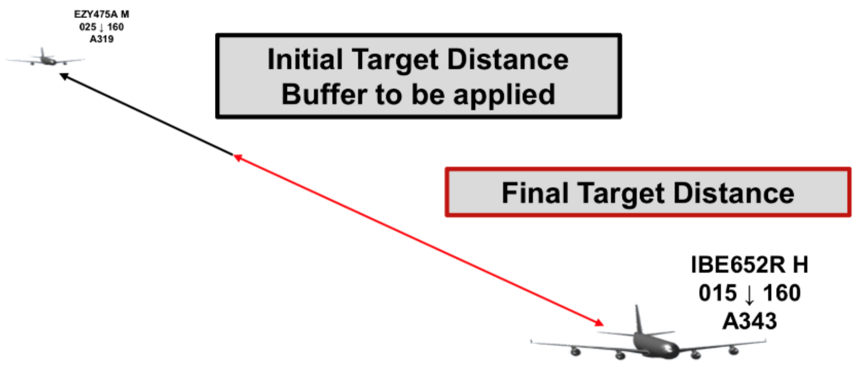


Figure 3 Buffer Added to the Minimum Distance Separation (C.L.V.Driessen 2015).

Results from Delft University of Technology and EUROCONTROL's study shows that the technical buffers between in-trail arrivals differ for different in-trail groups and different wind conditions based on RECAT-EU separation rules at Vienna Airport (C.L.V.Driessen 2015). The results show that the buffer distance increases when a relative "small" aircraft follows a relative "large" aircraft. However, only one airport was studied, and there were differences between separation rules in the U.S and Europe. Tamas Kolos-Lakatos studied the separation buffer at PHL, LGA, BOS, and EWR using the ASDE-X data collected in 2013. The study shows the buffer distance differs between different in-trail groups, but most of the airports were using the legacy separation rules that contains only five categories (Kolos-Lakatos 2013). Today, many U.S. airports are applying the reduced separation rules called RECAT I and will apply a more detailed aircraft based separation called RECAT II in future. This study presents an analysis of technical buffers under these separation rules. The study performs detailed technical analysis of in-trail buffers under RECAT I separation rules based on the ASDE-X dataset collected by a companion study of runway exit design (Trani and Mirmohammadsadeghi 2017). Runway Occupancy Time will be a limiting factor to the runway throughput in the future. The technical buffer is capital to compare the minimum wake separation with the leading aircraft ROT.

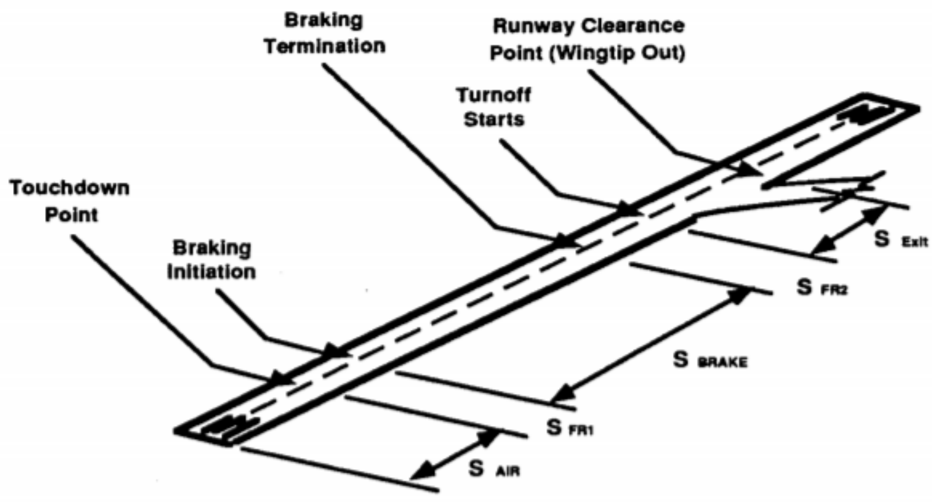
2.2 Runway Occupancy Time

Among the wide spectrum of runway capacity elements, the time spent by aircraft on the runway is one of the most important elements. To be able to define ROT, we will use the definitions provided by FAA:

Threshold: "The beginning of the portion of the runway usable for landing" (FAA 2008).

Runway-holding position: "A designated position intended to protect a runway, an obstacle limitation surface, or an ILS/MLS critical/sensitive area at which taxiing aircraft and vehicles shall stop and hold, unless otherwise authorized by the airport control tower" (FAA 2008).

The occupancy of time for arrival aircraft starts with threshold crossing and ends with the tail-off runway. Figure 4 shows the elements of runway occupancy during the landing procedure.



S_{AIR}	Air Distance to Touchdown
S_{FR1}	Free Roll Distance Between Touchdown and Application of Brakes
S_{BRAKE}	Braking Distance
S_{FR2}	Free Roll Distance Between Brake Termination and Start of Turn
S_{Exit}	Longitudinal Exit Distance to Clear the Runway

where

S_{air} = air distance (m)

h_{th} = threshold crossing altitude (m)

γ = tangent value of descending angle

v_{fl} = flare speed (m/s)

g = acceleration of gravity (m/sec^2)

Figure 4 Elements of Runway Occupancy during the Landing Procedure (Trani, Hobeika et al. 1992).

There have been several pieces of research focusing on analyzing and collecting Runway Occupancy Time for different runways and airports. In 1984, the MITRE Corporation collected data on the runway occupancy time at La Guardia, Boston, and Newark Airports using a digital watch from the control tower. The ROTs were defined as the time interval between crossing the threshold and clearing the runway. The result showed that runway condition and visibility had no significant influence on the runway occupancy time. The average ROT for small and large aircraft was 46 seconds at La

Guardia and Newark, and 52.1 seconds for Boston (Weiss and J.N.Barrer 1984). However, since the data was collected by observation, it was hard to define the time to cross the threshold and clear runway. In 1992, Sherali found the ROT was influenced by the geometry and location of high-speed exits on a runway and developed an integrated simulation program to determine optimal exit locations with minimum ROT (Sherali, Hobeika et al. 1992). In 1993, Hobeika and Trani had developed the Runway Exit Design Interactive Model (REDIM) to locate and design high-speed runway exit at airports optimally and applied to lots of airports in the U.S. to minimize the ROT (Hobeika, Trani et al. 1993). In 1995, Gu found that the gate location could also influence the ROT, and developed a choice model to predict the taxiing path and landing procedures (Gu, Trani et al. 1995). The model was tested and validated by operations at Washington National Airport. In 1996, Kim developed a Monte Carlo simulation model based on heuristics derived from observations to estimate landing-roll trajectories (Kim, Trani et al. 1996). In 2001, Martinez applied activity-based simulation techniques to model runway capacity, delays, and double runway occupancy time instance. The simulation results showed the heavy aircrafts had an average ROT of 55 seconds with a standard deviation of 6 seconds. The medium and light aircraft types had an average ROT of 50 and 45 seconds respectively with a standard deviation of 10 seconds (Martinez, Trani et al. 2001). In 2008, Baik and Trani had developed a new microscopic simulation model called Virginia Tech airfield simulation model (VTASIM) (Baik and Trani 2008). The model employs a vehicle-following model to portray aircraft movements on taxiways in details and had a higher accuracy comparing to traditional discrete-event based airport simulation models. In 2010, V. Kumar and R. Kicinger had analyzed the ROT at DFW through surface radar for different runways by redefining ROT as “the time between the instant an aircraft crosses the imaginary plane of the threshold and the instant it is 25 feet clear of the runway boundary”. The result showed that ROT was influenced by different aircraft types and runway configurations (Kumar 2010). However, for wide-body aircraft, the tail would still occupy the runway 25 feet from the edge of the runway. In the year 2013, Tamas had analyzed the ROT at Philadelphia, Newark, and Boston and La Guardia airports by using the precise ASDE-X data. The ROT was defined as the time interval between the crossing threshold and exiting a holding point to ensure safety. The results showed that for some aircraft pairs, there was very little room to reduce separation minimum under RECAT I, and the ROTs would be the limiting factor for runway throughput in the future (Kolos-Lakatos 2013). In 2017, Navid had compared the ROTs values for three different main definitions as shown in Figure 5 by analyzing the ASDE-X data at ORD airport (Mirmohammadsadeghi and Trani 2017). The results in Figure 6 showed there was a significant difference between runway occupancy time to runway edge, runway occupancy time when aircraft was fully out, and runway occupancy time until hold bar.

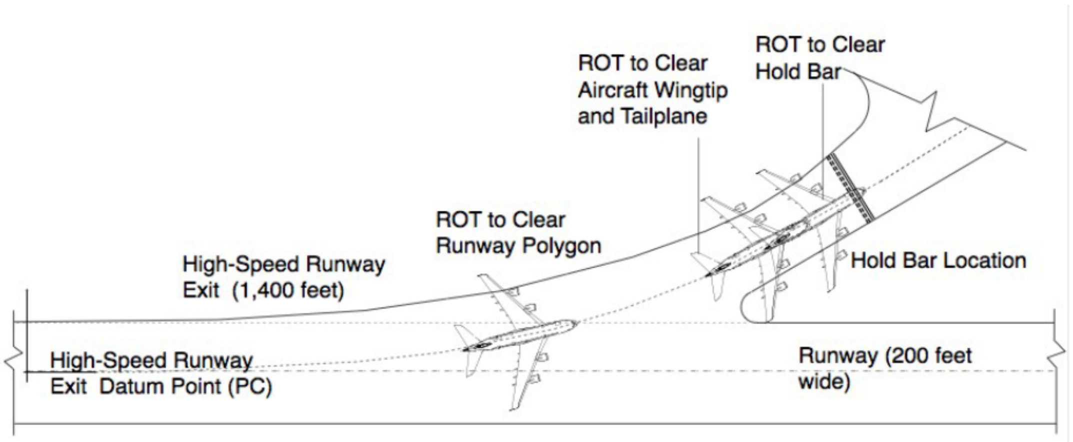


Figure 5 Three main definitions of Runway Occupancy Time (Trani, 2017).

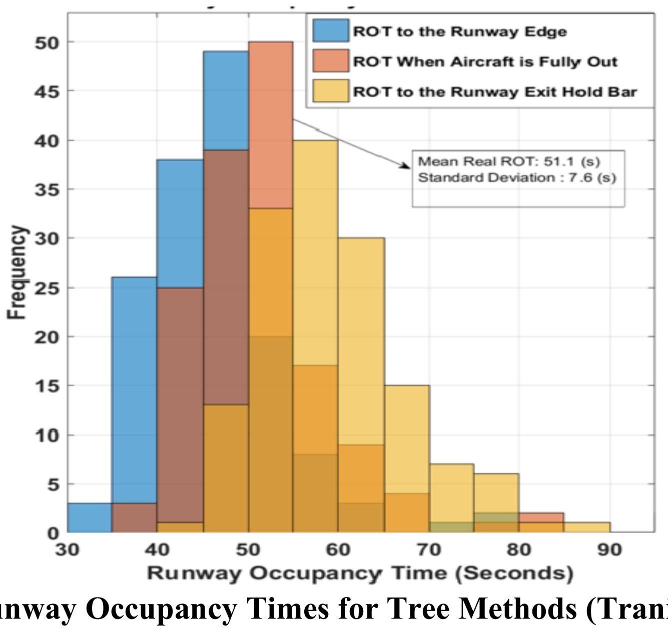


Figure 6 B738 Runway Occupancy Times for Tree Methods (Trani, 2017).

2.3 Quick Response Runway Capacity Model (RUNSIM)

The Quick Response Runway Capacity Model (RUNSIM) is developed by Air Transportation System Laboratory (ATSL). The model blends analytical and microscopic simulation techniques. The model uses the Monte Carlo simulation to estimate the airport throughput by generating an ordered sequence of flights based on the fleet mix at the airport. Figure 7 shows the Graphic User Interface in RUNSIM enables users to change input parameters and default values to analyze different scenarios. The capacity model developed the capability to estimate for 77 airports in the US contained in the NASA ACES simulation model (Pu and Trani 2014).

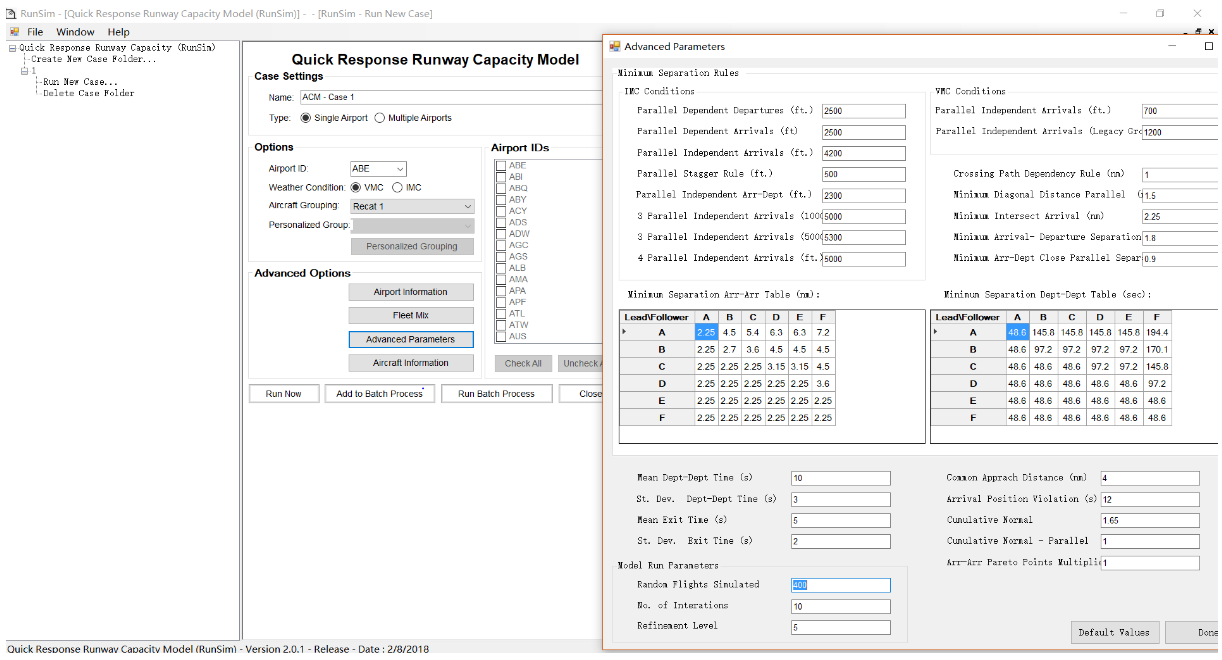
The inputs parameters can be classified into four categories:

Fleet mix: RUNSIM is using the fleet mix estimating from FAA Air Traffic Activity System (ATADS) database for each airport

Separation rules: the RUNSIM is capable of Legacy Separation, RECAT I, RECAT II and TBS for both departures and arrivals

Aircraft characteristics: the RUNSIM records the mean arrival runway occupancy time (seconds), standard deviation of runway occupancy time, approach speed (knots), mean departure runway occupancy time (seconds), standard deviation of departure runway occupancy time, minimum arrival runway length (feet) and minimum departure runway length (feet) for each aircraft type from the Performance Data Analysis and Reporting System (PDARS).

Environment information: RUNSIM can consider weather conditions by applying 0.9 converting factor between Visual Meteorological Conditions (VMC) and Instrumental Meteorological Conditions (IMC); RUNSIM use considerable large numbers for non-towered airports according to FAA Order 7110.


RunSim - [Quick Response Runway Capacity Model (RunSim)] - - [RunSim - Run New Case]

Quick Response Runway Capacity Model

Case Settings

Name: ACM - Case 1

Type: [✓ Single Airport] Multiple Airports

Options

Airport ID: ABE

Weather Condition: [✓ VMC] IMC

Aircraft Grouping: Recat 1

Personalized Group:

Airport IDs: ABE, ABI, ABQ, ABY, ACY, ADS, ADW, AGC, AGS, ALB, AMA, APA, APF, ATL, ATW, AUS

Advanced Parameters

Minimum Separation Rules

IMC Conditions

Parallel Dependent Departures (ft.)	2500
Parallel Dependent Arrivals (ft)	2500
Parallel Independent Arrivals (ft.)	4200
Parallel Stagger Rule (ft.)	500
Parallel Independent Arr-Dept (ft.)	2300
3 Parallel Independent Arrivals (100(	5000
3 Parallel Independent Arrivals (500(	5300
4 Parallel Independent Arrivals (ft.)	5000

VMC Conditions

Parallel Independent Arrivals (ft.)	700
Parallel Independent Arrivals (Legacy Gr(	1200

Crossing Path Dependency Rule (nm)	1
Minimum Diagonal Distance Parallel (	1.5
Minimum Intersect Arrival (nm)	2.25
Minimum Arrival- Departure Separation	1.8
Minimum Arr-Dept Close Parallel Separ	0.9

Minimum Separation Arr-Arr Table (nm):

Lead\Follower	A	B	C	D	E	F
A	2.25	4.5	5.4	6.3	6.3	7.2
B	2.25	2.7	3.6	4.5	4.5	4.5
C	2.25	2.25	2.25	3.15	3.15	4.5
D	2.25	2.25	2.25	2.25	2.25	3.6
E	2.25	2.25	2.25	2.25	2.25	2.25
F	2.25	2.25	2.25	2.25	2.25	2.25

Minimum Separation Dept-Dept Table (sec):

Lead\Follower	A	B	C	D	E	F
A	48.6	145.8	145.8	145.8	145.8	194.4
B	48.6	97.2	97.2	97.2	97.2	170.1
C	48.6	48.6	48.6	97.2	97.2	145.8
D	48.6	48.6	48.6	48.6	48.6	97.2
E	48.6	48.6	48.6	48.6	48.6	48.6
F	48.6	48.6	48.6	48.6	48.6	48.6

Mean Dept-Dept Time (s)	10
St. Dev. Dept-Dept Time (s)	3
Mean Exit Time (s)	5
St. Dev. Exit Time (s)	2
Common Apprach Distance (nm)	4
Arrival Position Violation (s)	12
Cumulative Normal	1.65
Cumulative Normal - Parallel	1
Arr-Arr Pareto Points Multipli(	1

Model Run Parameters

Random Flights Simulated	400
No. of Interations	10
Refinement Level	5

Quick Response Runway Capacity Model (RunSim) - Version 2.0.1 - Release - Date : 2/8/2018

Figure 7 Graphic User Interface in RUNSIM Version 2.0.1 (Trani et al. 2018).

3. Determination of Current Arrival Separation Technical Buffers

3.1 Definition of technical buffers

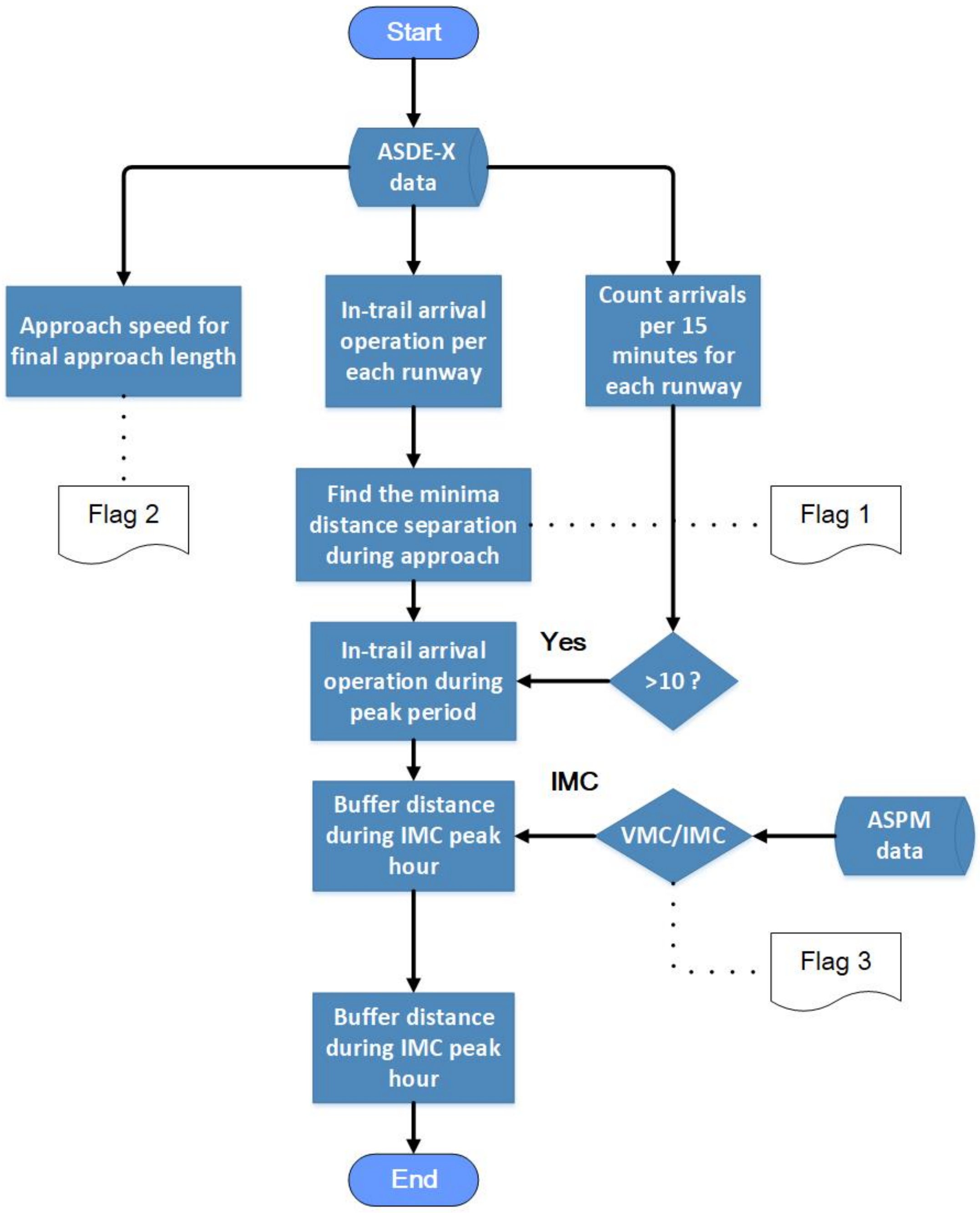
Currently, at many airports, the runway is the limiting factor for the overall capacity. Among the most critical parameters are the wake turbulence separation minimum expressed in the distance and the differences in speed profile between the leader and follower aircraft for arrivals on final approach that limit the arrival flow at many airports (Gerz, Holzäpfel et al. 2002).

For dealing with the existing system error caused by facilities and the human behavior error caused by pilots, the Air Traffic Control (ATC) consider a technical buffer based on experience for ensuring minimum separation. However, there is no criteria or detailed research on buffer time between in-trail arrivals as lack of data. For most of the airport capacity simulation models, buffer time is defined as a fixed value to cover 95% of the normal distribution based on the inputs given by the user.

This study shows that the technical buffers differs for different in-trail groups and does not follow a normal distribution. The main causes are the minimum separation and approach speed varies for different in-trail groups and aircraft types. The study finally creates a kernel distribution for the buffer times per each in-trail group to improve runway throughput estimation.

3.2 Methodology to obtain technical buffers

The study focuses on identifying the distribution of technical buffers during peak period to better estimate the maximum throughput capacity. The study identifies a threshold of peak period based on observation and analysis of ASDE-X data. As each airport is unique (i.e., configuration, elevation, weather, fleet matrix, etc.), the threshold for peak period may be different from airport to airport. For example, airports with high demand like ORD, CLT, DEN, and ATL have a higher threshold for peak periods. On the other hand, some more conservative airport with lower demand such as MCO and IAD have a lower threshold. The technical error also different from airports to airports because of the airport configuration and air traffic. In this research, the ADSE-X data for each airport is used to identify these criteria. Figure 8 shows the flowchart about how to use ADSE-X data to obtain the technical error.


Start
ASDE-X data
Approach speed for final approach length
In-trail arrival operation per each runway
Count arrivals per 15 minutes for each runway
Flag 2
Find the minima distance separation during approach
Flag 1
Yes
In-trail arrival operation during peak period
>10 ?
IMC
Buffer distance during IMC peak hour
VMC/IMC
ASPM data
Flag 3
Buffer distance during IMC peak hour
End

Flag 1: Location of minimum distance separation indicator (0–Threshold,1-Near threshold,3-Far from threshold)
Flag 2: “Opening” or “Closing” case indicator (1-Opening case,2-Closing case,3-Marginal case (Delta speed < 5))
Flag 3: IMC/VMC indicator (1-VMC,2-IMC)

Figure 8 Flowchart to Obtain Buffer Time-based on ASDE-X Data.

This study uses a dynamic way to find the minimum distance separation during the final approach phase. Notice it is hard to recognize the value and location of the minimum distance separation between in-trail groups. For example, for a "closing case" (when the leading aircraft approach speed is higher than the following aircraft), the minimum separation will locate at the runway threshold theoretically. For an "opening case" (when the leading aircraft approach speed is lower than the following aircraft), the minimum separation will locate at the Final Approach Fixed Point (FAF). This research checks the distance between in-trail arrivals for every second and identifies the value and location of the minimum distance separation as shown in Figure 9. For future studies, a flag indicator is introduced to identify the location for the minimum distance separation (0 means at the threshold, 1 means not at threshold but near, 2 means not at threshold).

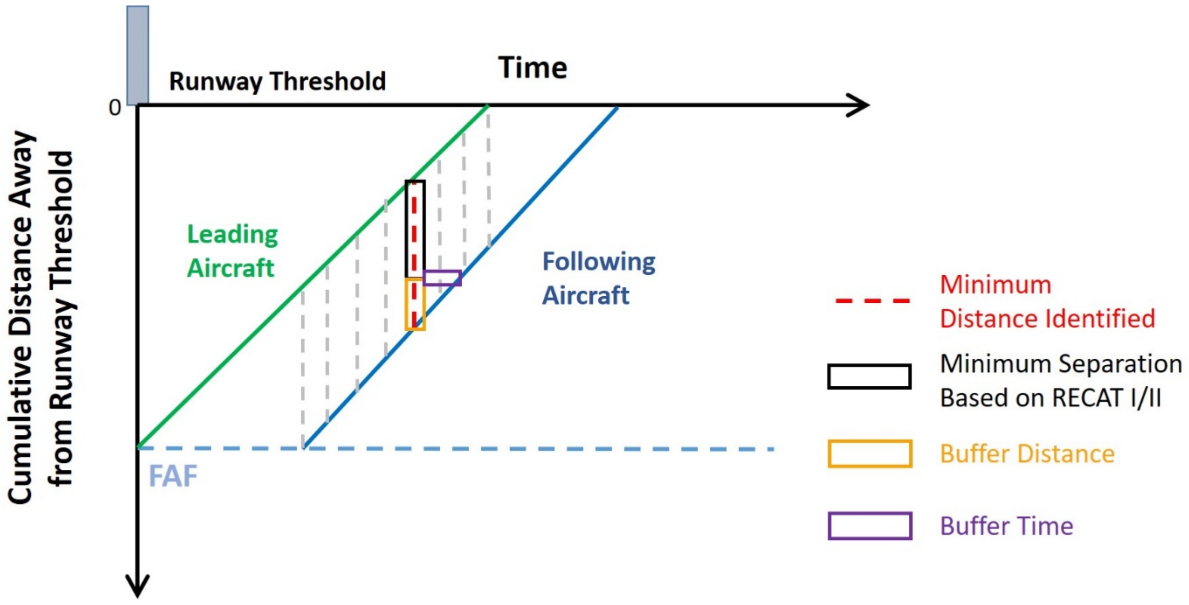

Runway Threshold
Time
0
Cumulative Distance Away from Runway Threshold
Leading Aircraft
Following Aircraft
FAF
Minimum Distance Identified
Minimum Separation Based on RECAT I/II
Buffer Distance
Buffer Time

Figure 9 Methodology to Identify Minimum Distance between In-trail Arrivals, Buffer Distance, and Buffer Time.

This study adds three flag indicators for future analysis of in-trail arrival behaviors: Location of minimum distance separation indicator (Flag 1), "Opening" and "Closing" case indicator (Flag 2), and IMC/VMC indicator (Flag 3). For flag 1, if the minimum separation between in-trail arrival appears at the threshold, flag 1 equals to zero; if the minimum separation appears near threshold (within 10 distant points, near 0.5 nautical miles), flag 1 equals one; else the minimum separation is identified far away from threshold, and flag 1 equals two. For flag 2, if the average final approach speed for leading aircraft is 5 knots larger than the leading aircraft, the distance between the in-trail arrivals is increasing and thus identified as an "opening" case (Flag 2 equals 1). If the average final approach speed for leading aircraft is 5 knots less than the leading aircraft, it will be identified as a “closing” case (Flag 2 equals 2). If the speed difference between leading and the following aircraft is within 5 knots, it will be identified as a “marginal” case (Flag 2 equals 3). For flag 3, one standard for the in-trail arrival operates under VMC condition and two standards for IMC condition.

3.2.1 Visibility Condition Factor

This study incorporates the visibility condition from Aviation System Performance Metrics (ASPM) data with the ASDE-X data. Visibility condition of the airport is also important for identifying the buffers. The Air Traffic Control (ATC) can operate Visual Flight Rules (VFR) only under Visual Meteorological Conditions (VMC), while IFR permits an aircraft to operate in IMC, which is essentially any weather condition less than VMC but in which aircraft can still operate safely. The main difference between IFR and VFR for this research lies in the separation between in-trail arrivals. Pilots are allowed to maintain the separation by themselves to some degree under VFR conditions. This study only focuses on the buffer times under IMC to be credible.

3.2.2 Identifying the Peak Period

This study defines the peak period threshold as ten arrivals per 15 minutes for each runway end based on ASDE-X data. The peak period buffer times will be used to estimate the runway throughput in RUNSIM. Currently, there is no definition or criteria about the peak period for each airport. For each runway end, the histogram of daily arrivals per 15 minutes is generated to identify the peak period threshold. The threshold not only makes sure that the arrivals are operating almost at capacity, but also should have a certain frequency so that enough data could be obtained for buffer time. For example, based on the arrival frequencies at Runway 10C at ORD airport as shown in Figure 10, a threshold of 10 arrivals landing on the runway per 15 minutes is set as a standard of the peak period. The airport capacities published by FAA in the year 2014 also indicates ORD could reach up to 40 arrivals per hour for each arrival runway (Gentry, Duffy et al. 2014).

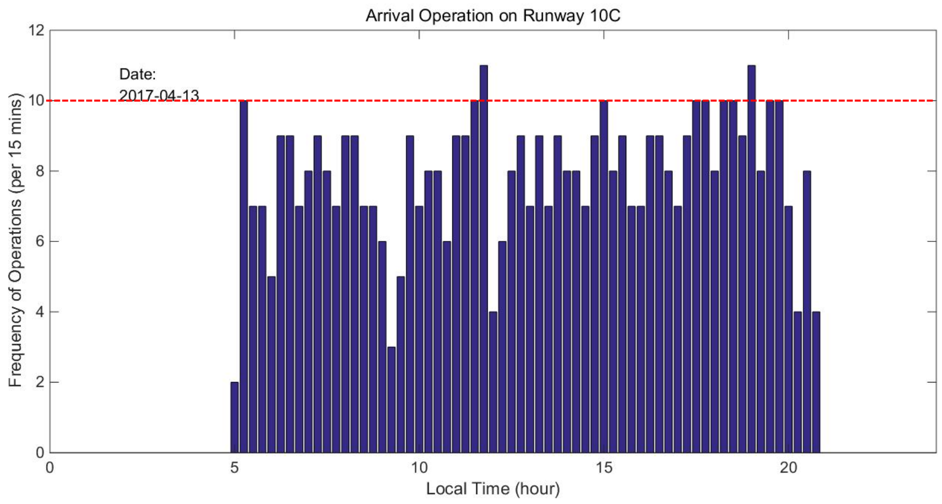


Figure 10 Sample Histogram Plot of Arrivals per 15 Minutes on Runway 10C in ORD.

Notice that since the minimum separations are different for different in-trail groups, this method may exclude some of the in-trail groups with larger separations. For example, when the leading aircraft belongs to wake category B and the following aircraft belongs to wake category E (in-trail group B-E), the minimum separation is five nautical miles according to Wake Turbulence Re-categorization Phase I (RECAT I). Since the increase of separation, it needs more time to clear the wake vortex, which leads the decreasing of frequency for arrivals during peak periods. The threshold may change to obtain more data to capture the buffer times between these in-trail groups with larger separation in the future study.

3.2.3 Identifying the Final Approach Length

This study identifies approach length for each airport. The approach length is also critical to identify the buffer for "opening cases" and approach speed. Since the Final Approach Fix (FAF) locates differently from the airport to airport, the approach lengths are different for different airports. The ADSE-X data contains the longitude and latitude point for each operation per every second. Based on the consecutive positions of the aircraft, the cumulative distance and the heading per each second can be obtained by using the "legs" function in Matlab (Matlab R2016b – academic use), which is used to calculate the courses and distance between navigational waypoints. However, since there exists facility error in the ADSE-X data, the course degree may wag from side to side. To deal with the "noise" of data, we group each sequential five-course data and use the median value to standard for the heading as shown in Figure 11. The final approach length is identified when the heading deviation is more than 10 degrees from the runway orientation. Figure 12 shows that the approach length for ORD is 9 nautical miles from runway threshold for Runway 10C. The 8.5 nautical miles is selected to obtain more data from the overall ORD airport, and 6 nautical miles for CLT, DEN, and ATL airport, which accommodates to the FAF position from FAA report (Airnav.com).

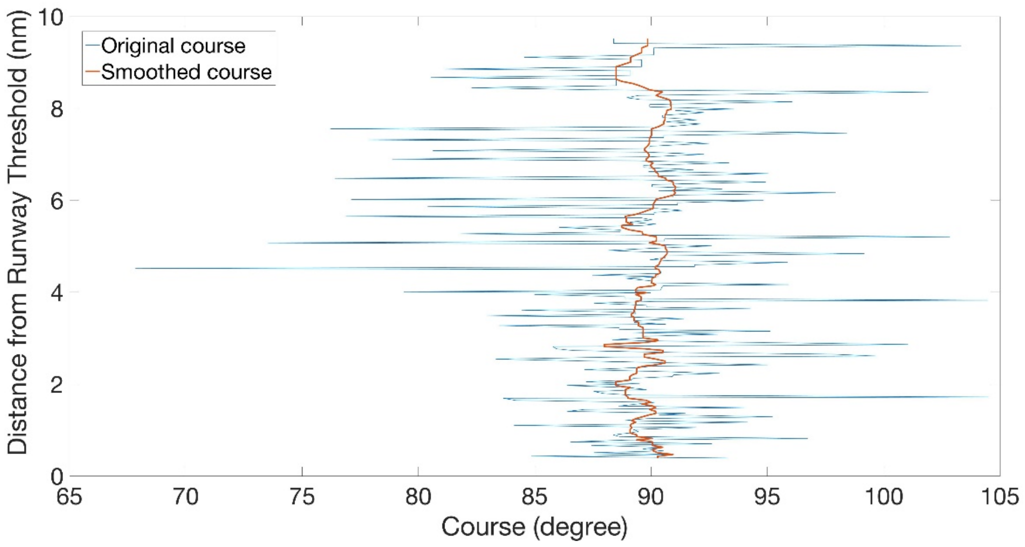


Figure 11 Sample Cumulative Distance – Course Plot for Sample Arrival Operation at ORD.

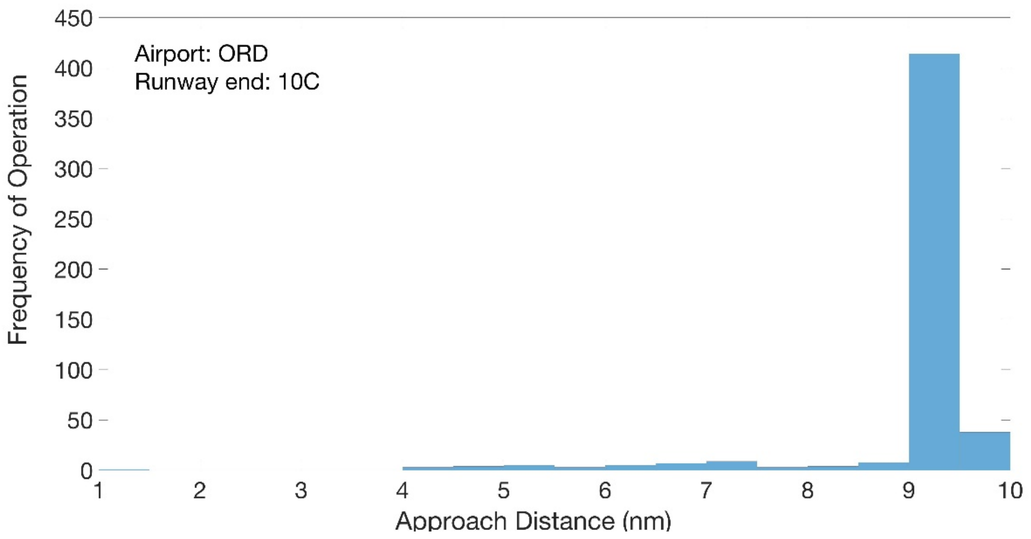


Figure 12 Sample Approach Length Distribution for Runway 10C at ORD.

3.3 Buffer time Result for Several Main Airports

This study analyzed buffer times for several main airports: Chicago O'Hare (ORD), Douglas International Airport (CLT), Hartsfield - Jackson Atlanta International Airpot (ATL), Denver International Airport (DEN), Ronald Reagan Washington National Airport (DCA), LaGuardia Airport (LGA), San Diego International Airport (SAN), Washington Dulles International Airport (IAD), and Orlando International Airport (MCO).ORD, DEN, LGA, ATL and CLT airports are applying RECAT 1 Wake Separation rules while DCA, IAD and MCO airports are still applying Legacy Wake Separation rules. SAN airport is using RECAT Phase 2. However, SAN has only one single runway and is under mixed operation, the separation between in-trail arrivals are larger because of the departures between in-trail arrivals. For Legacy Separation Airports, reduced separation (2.5 nm) between arrivals is authorized for instrument approaches to Runway 1 at DCA, Runway 17L, Runway 17R, and 18R at MCO, Runway 19C and 19L at IAD. For DCA airport, since the configuration of runways, the arrivals on Runway 1 could be influenced by the departures on other runways. For MCO airport, because of the configuration of the airport, there may exit a gate problem, which may influence the capacity of the airport and increase the gaps between arrivals. For DEN airport, the approach speeds are influenced by airport elevation and thus influence the buffer time between in-trail arrivals. For these reasons, airports under RECAT 1 are critical to this research, which are ORD, LGA, DEN, ATL, and CLT. Based on the results, the airports can be categorized into two groups: Airports below 2,500 feet and Airports above 2,500 feet.

3.3.1 Buffer Time Comparison between Different Airports

The histogram of buffer times in Figure 13 shows that there exists a significant difference between airports below 2,500 feet and airports above 2,500 feet. For example, the mean buffer time is 19.21 seconds for ORD, 18.69 seconds for CLT, and 18.44 seconds for ATL. All of these airports' elevations are below 2.500 feet and has similar buffer times. On the other hand, DEN has a mean buffer time of 27.48 seconds with an airfield elevation of 5,437 feet. The buffer time cumulative density plot for ORD, CLT, ATL, and DEN in Figure 14 also indicates that buffer times distribution should be categorized as airports below 2,500 feet and airports above 2500 feet.

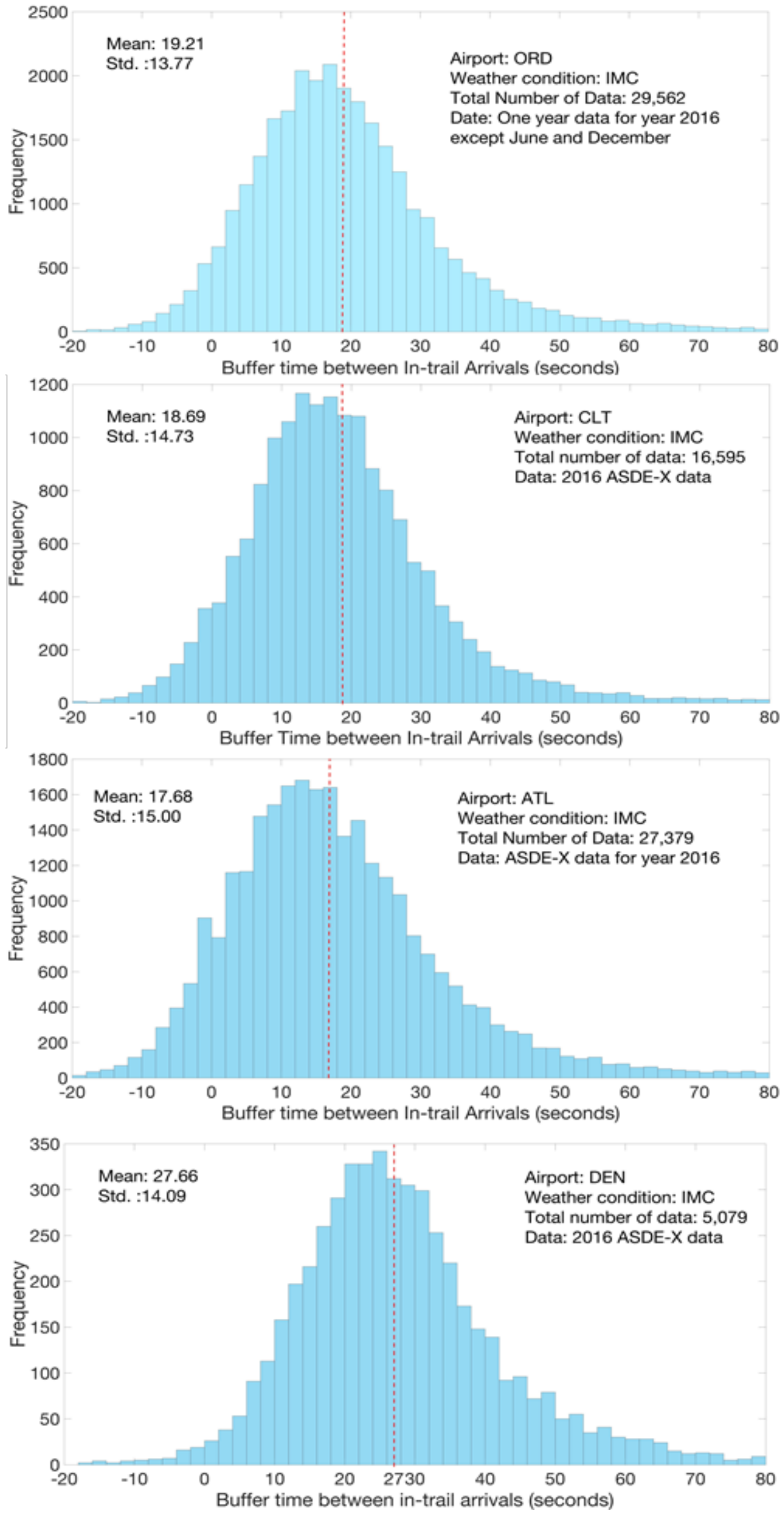


Figure 13 Buffer Time Distribution during IMC at ORD, CLT, ATL, and DEN.

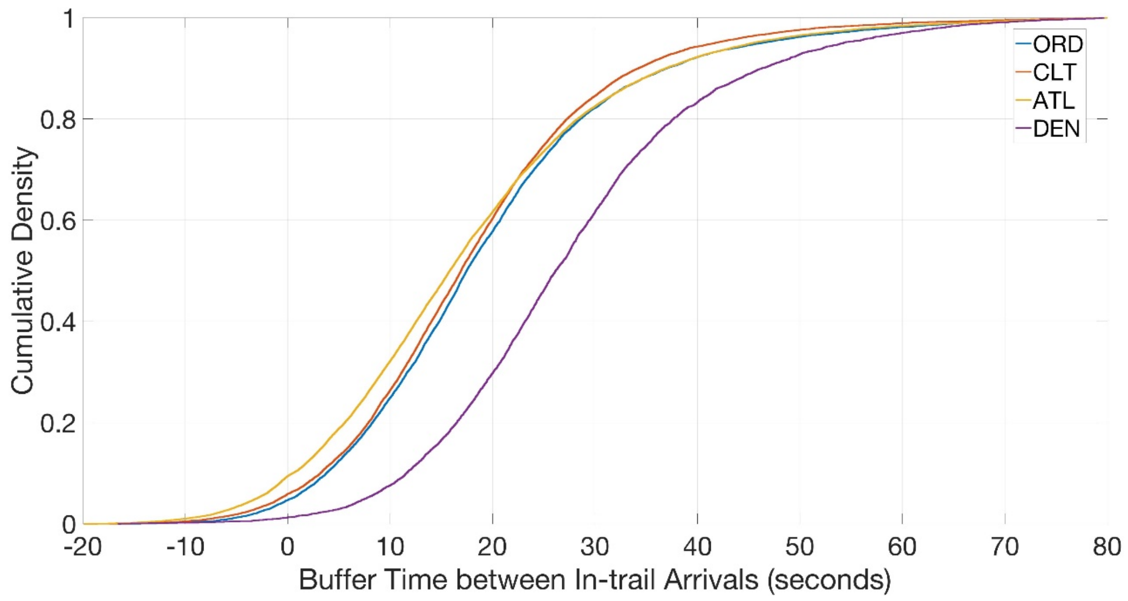


Figure 14 Cumulative Density Plot of All Buffer Times at ORD, CLT, ATL, and DEN.

From the comparison, the buffer times at DEN trends to be 8 seconds larger than buffer times at airports below 2,500 feet. One of the main cause could be the difference of approach speed between arrivals at these airports. As DEN has an elevation of 5,434 feet, the air density is lower, and the approach speed is higher with higher elevation according to the Mach speed equation 1. Table 7 shows that from the analysis of ADSE-X data, the airport has an elevation above 2,500 feet trends to have a higher approach speed comparing to airports below 2,500 feet.

$$M_{true} = \sqrt{5\left[\left\{\frac{\rho_0}{\rho}\left(\left[1+0.2\left(\frac{V_{IAS}}{661.5}\right)^2\right]^{3.5}-1\right)+1\right\}^{0.286}-1\right]} \tag{1}$$

Table 7 Approach Speed for Different Wake Group Aircrafts at ORD, CLT, ATL, and DEN.

Airport	Elevation (feet)	Average approach speed for last 2.5 nm (knots)				
		B763	A320	B737	CRJ 9	LJ75
		C	D	D	E	F
ORD	680	138.6	135.6	133.6	134.2	129.4
CLT	748	137.0	138.7	135.9	135.5	127.4
ATL	1026	142.2	138.6	136.6	139.3	128.2
DEN	5434	152.5	153.2	150.7	150.4	144.9

3.3.2 Buffer Time Comparison between Different In-trail Groups

The study also shows a significant difference in buffer times between different in-trail groups. Since for most of the airports, the majority of the operated aircraft belongs to wake group D and group E, the buffer time distribution between the critical in-trail groups (i.e., D-D, D-E, E-D, and E-E) is detail analyzed. The cumulative density distribution plots (CDF-plots) for these four groups at both ORD and CLT shows that there exits significant difference in buffer times between different in-trail groups in Figure 15 and Figure 16. The box plot and the statistic test result table of the four critical in-trail arrival groups at ORD shown in Figure 17 and Table 8 indicate additional 6 seconds buffer time is applied to in-trail D-E (leading aircraft belongs to wake group D with the following aircraft belongs to wake group E) groups compared with in-trail E-D groups. The Bartlett's test has a *p*-value of 0.01, which indicates there exists strong relativity between buffer times and in-trail groups. (Null hypothesis: there is no significant relativity between buffer times and in-trail groups)

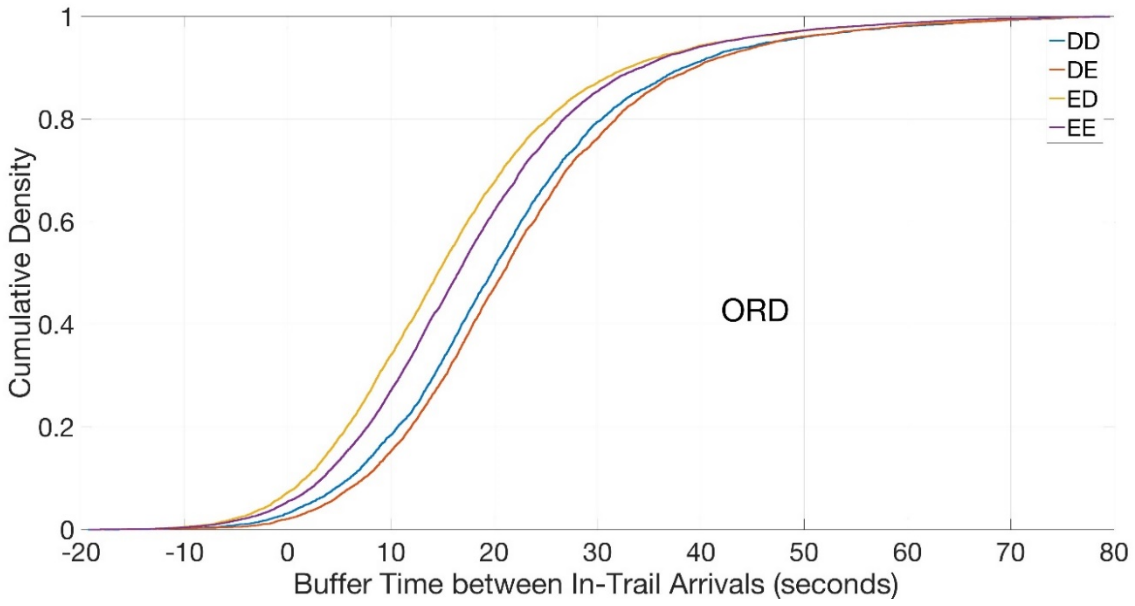


Figure 15 Cumulative Density Plot of Buffer Time for DD, DE, ED, EE groups during IMC at ORD.

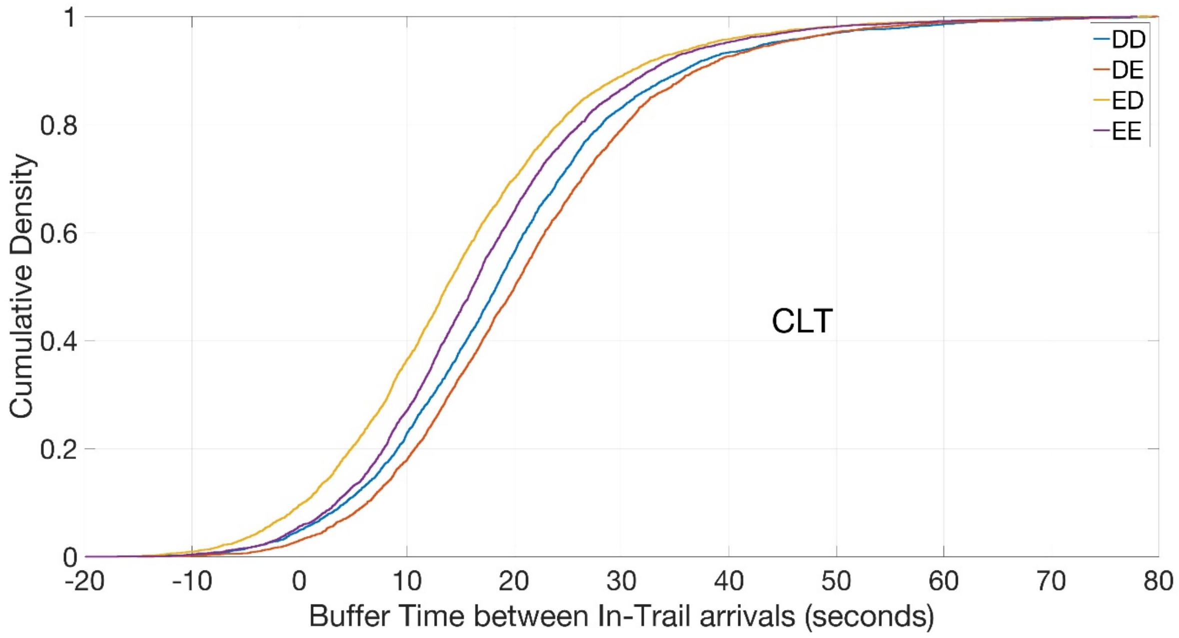


Figure 16 Cumulative Density Plot of Buffer Times for DD, DE, ED, EE groups during IMC at CLT.

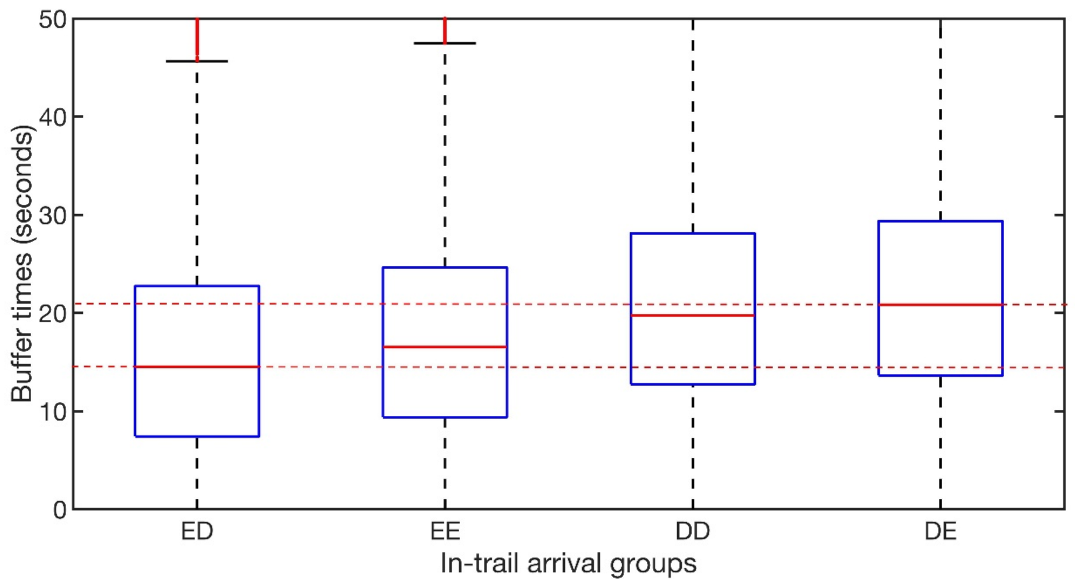
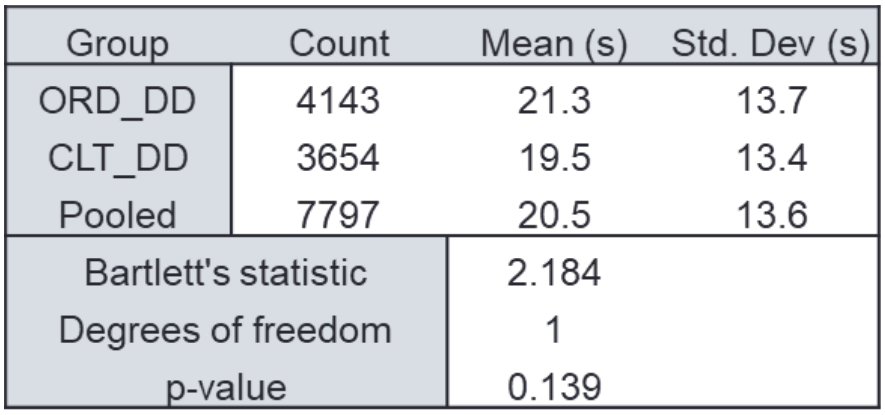


Figure 17 Box-plot of Buffer Time Distribution for ED, EE, DD, and DE In-trail Groups at ORD.

Table 8 Bartlett's Statistic Result Table between Buffer Time and Critical In-trail Groups at ORD.



Group	Count	Mean (s)	Std. Dev (s)
ORD_DD	4143	21.3	13.7
CLT_DD	3654	19.5	13.4
Pooled	7797	20.5	13.6

Bartlett's statistic	2.184
Degrees of freedom	1
p-value	0.139

From the cumulative density plots for the in-trail groups with more than 100 data points during IMC conditions at ORD in Figure 18, four phenomena for buffer times are found:

- Airports with higher elevations have higher buffer times comparing airports with lower elevations
- There exists a significant difference in buffer times between different in-trail groups; the buffer times decreases when the leading aircraft belongs to D comparing with when the leading aircraft belongs to F

- For relative large aircraft followed by relatively small aircraft with separation larger than minimum radar separation, the buffer times decrease with the increase of minimum separation
- Buffer time increase a little bit when the leading aircraft belongs to wake group E

The distribution of buffer time between different in-trail arrival groups are very similar between airports below 2,500 feet. As for ORD, CLT and ATL are airports applying RECAT I with large amounts of operations; we compare the buffer times between these airports to identify the similarities in buffer time. Table 9 shows the mean value and standard deviation of the four critical in-trail groups (i.e., D-D, D-E, E-D, E-E) at ORD, CLT, and ATL. For each in-trail group, the difference in buffer time is less than 2.5 seconds among these airports. The standard deviation for each in-trail group between ORD and CLT are less than one while ATL trends to have three to four more in standard deviation. This may lie in that for ORD and CLT, the arrivals are independent to each other, while for ATL, arrival Runway 10/28 is dependent with arrival Runway 9R/27L. In conclusion, as the buffer time is very similar for all three airports. For this reason, we adopt the buffer times in our simulation models for airports less than 2,500 feet.

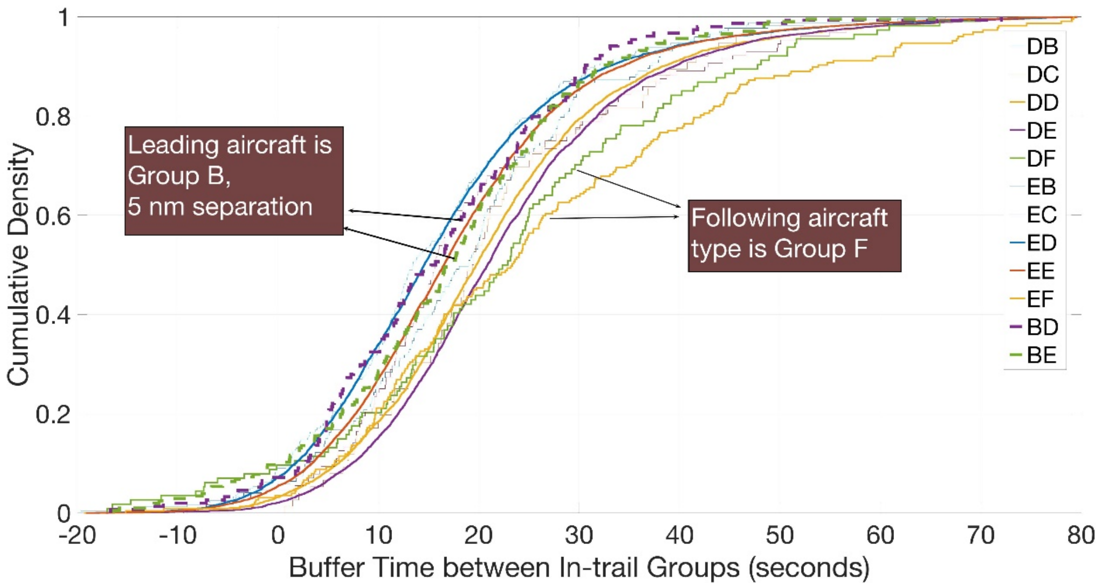


Figure 18 Cumulative Density Distribution of Buffer Time for All In-trail Groups with More Than 100 Data Sets during IMC at ORD.

Table 9 Similar Mean and Standard Deviation for Buffer Times between DD, ED, EE, DE in-trail groups at ORD, CLT and ATL (airports below 2.500 feet).

Airport_In-trail Group	Count	Mean (s)	Std. Dev(s)
ORD_DD	4,143	21.3	13.7
CLT_DD	3,654	19.5	13.4
ATL_DD	17,356	18.8	17.1
ORD_ED	6,365	16.4	13.6
CLT_ED	3,984	15.1	12.9
ATL_ED	3,887	16.3	17.3
ORD_EE	10,293	18.0	13.2
CLT_EE	4,474	17.4	12.4
ATL_EE	918	17.4	15.4
ORD_DE	6,326	22.4	13.4
CLT_DE	3,972	21.1	13.0
ATL_DE	3,872	21.1	16.5

3.4 Re-generation of Buffer Time Distribution in RUNSIM

This analysis shows that a kernel distribution can approximate the observed buffer time distributions and for different in-trail groups. A kernel distribution is adopted in RUNSIM. Currently, the buffer times are defined as a normal distribution in simulation software such as Quick Runway Simulator and SIMMOD, etc. A chi-square test shows that buffer times do not follow a normal distribution. Some other parameter based distributions such as Weibull, Lognormal and Beta distributions are also tested, however, none of these distributions can fit all of the distributions of buffer times for different airports. A data-based distribution called kernel distribution is recommended for this research to describe buffer times. A kernel distribution is a nonparametric representation of the probability density function (PDF) of a random variable and is defined by a smoothing function and a bandwidth value, which controls the smoothness of the density curve. Figure 18 shows a comparison of the kernel distribution and normal distribution to fit buffer times between D-D in-trail groups at ORD. The Chi-square test between simulated buffer times and real buffer times for normal and kernel distribution shows that kernel distribution is a better fit of buffer time distribution better. Figure 19 shows that the kernel distribution can represent the "tail" of the distributions (small and large buffer times). This is important to calculate the percentage of violations of the minimum wake separation time versus runway occupancy time.

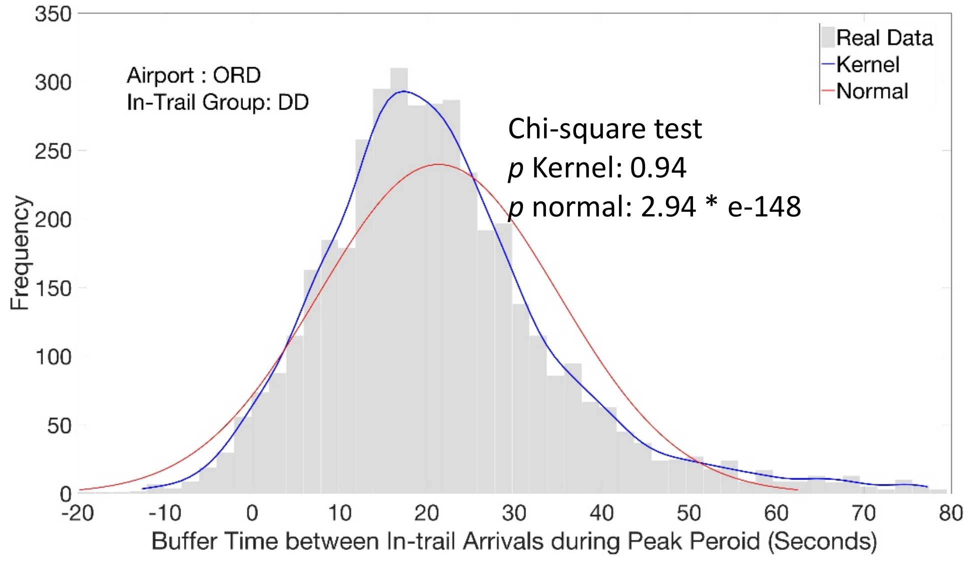


Figure 19 Comparison of Kernel distribution and normal distribution with real data for in-trail arrivals D-D group at ORD.

To validate the result, we generate 10,000 random number with buffer times for each in-trail arrival group at ORD. Figure 20 shows the cumulative distribution function of generated data goes well compared with observed data in Figure18.

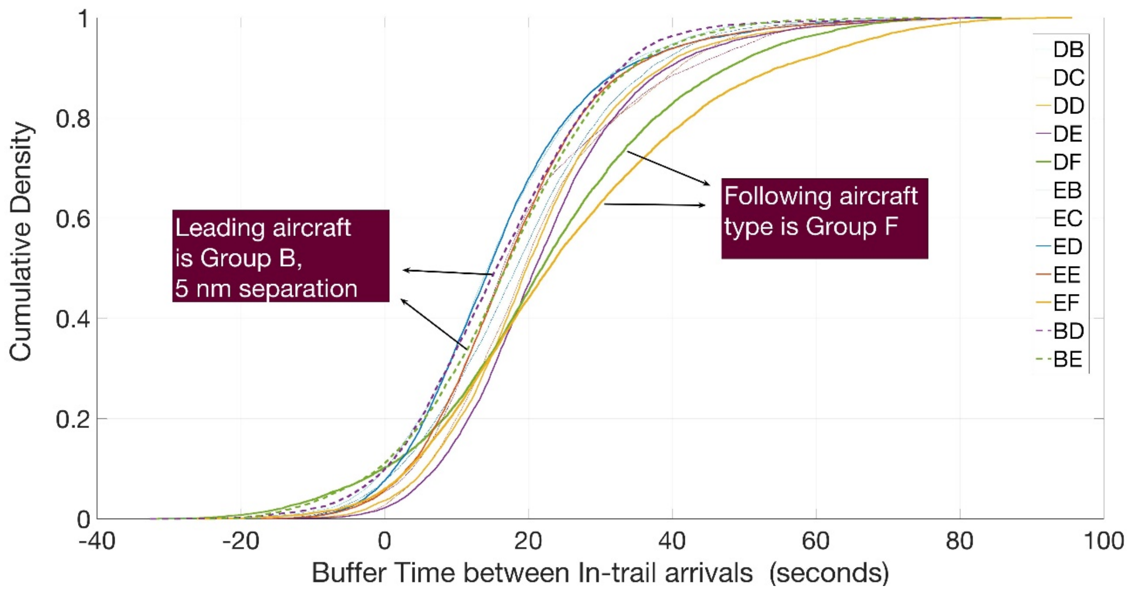


Figure 20 Cumulative distribution function plot for 10,000 random number from the simulation for each in-trail arrival group at ORD.

3.4.1 Statistic Tests

Kruskal-Wallis one-way analysis of variance is used to validate the correlation between simulation results from kernel distribution and the real data. Kruskal-Wallis test is a non-parametric method for testing whether samples originate from the same distribution, which is normally used for comparing two or more independent samples of equal or different sample sizes. The null hypothesis is the two data sets comes from the same data set. The methodology is shown below:

Rank all data from all groups together:

The test statistic is given by:

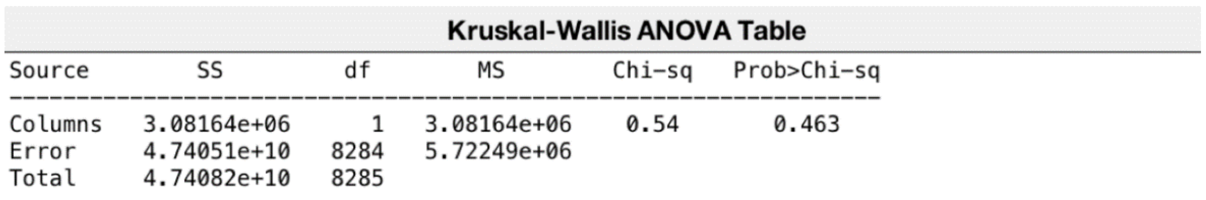
$$H = (N-1)\frac{\sum_{i=1}^{g} n_i(\overline{r_i}-\bar{r})^2}{\sum_{i=1}^{g}\sum_{j=1}^{g}\left(\overline{r_{ij}}-\bar{r}\right)^2} \qquad (2)$$

- n_i is the number of observations in group i
- r_{ij} is the rank (among all observations) of observation j from group i
- N is the total number of observations across all groups
- $\overline{r_i} = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$ is the average rank of all observations in group i
- $\bar{r} = \frac{1}{2}\ (N+1)$ is the average of all the r_{ij}

The p-value is approximated by $\Pr\left(\chi_{g-1}^2 > \mathrm{H}\right)$

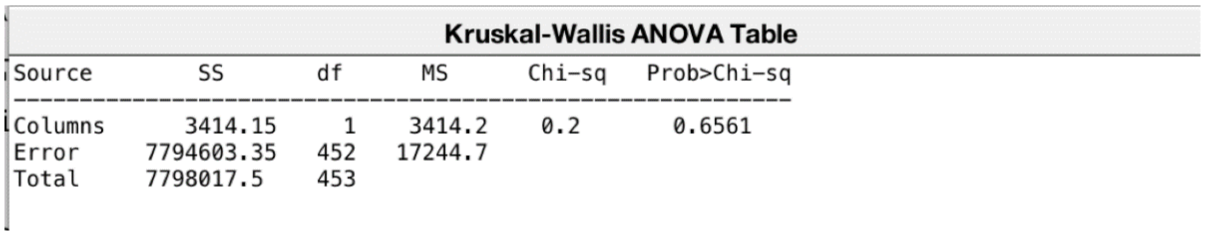
The Kruskal-Wallis test indicates that the kernel distribution is a good approximation of buffer times for both large (more than 1,000) and small (within 100 to 1,000) datasets. For example, Table 10 shows based on large amounts of data (4,142 data points in total) points for D-D pairs in-trail group buffer times; the Kruskal-Wallis test obtains a *p*-value of 0.46 between the kernel distributed buffer time and the real buffer. If a smaller sample is used (266 data points in total) for E-F in-trail groups, the Kruskal-Wallis test obtains a *p*-value of 0.66 as shown in Table 11. Since the *p* values are greater than 0.05, the statistic test shows the kernel distributed buffer time and real buffer time comes from the same distribution at 5% significance level. Figure 21 and 22 shows the box-plot of buffer times for D-D in-trail group and E-F in-trail groups. These figures indicates the similarities between kernel distributed buffer times and real buffer times.

Table 10 Kruskal-Wallis ANOVA Table for real data and simulation data for D-D in-trail arrival group at ORD



```
                         Kruskal-Wallis ANOVA Table
Source          SS          df        MS         Chi-sq    Prob>Chi-sq
-----------------------------------------------------------------------
Columns    3.08164e+06       1    3.08164e+06     0.54        0.463
Error      4.74051e+10    8284    5.72249e+06
Total      4.74082e+10    8285
```

Table 11 Kruskal-Wallis ANOVA Table for real data and simulation data for E-F in-trail arrival group at ORD



```
                       Kruskal-Wallis ANOVA Table
Source          SS         df        MS        Chi-sq    Prob>Chi-sq
--------------------------------------------------------------------
Columns      3414.15        1      3414.2      0.2         0.6561
Error      7794603.35     452     17244.7
Total      7798017.5      453
```

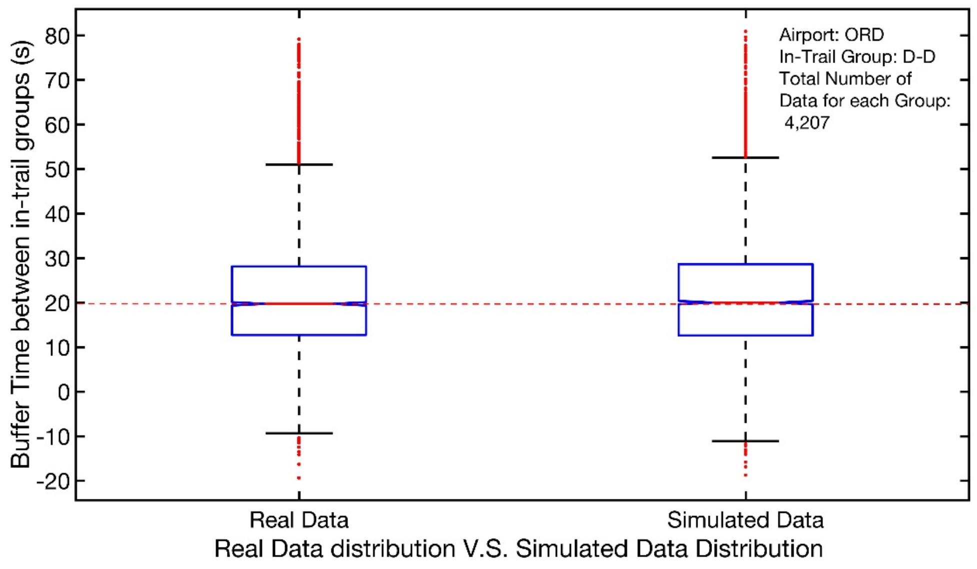


Figure 21 Comparison of box plot between real data and simulated data for D-D group at ORD

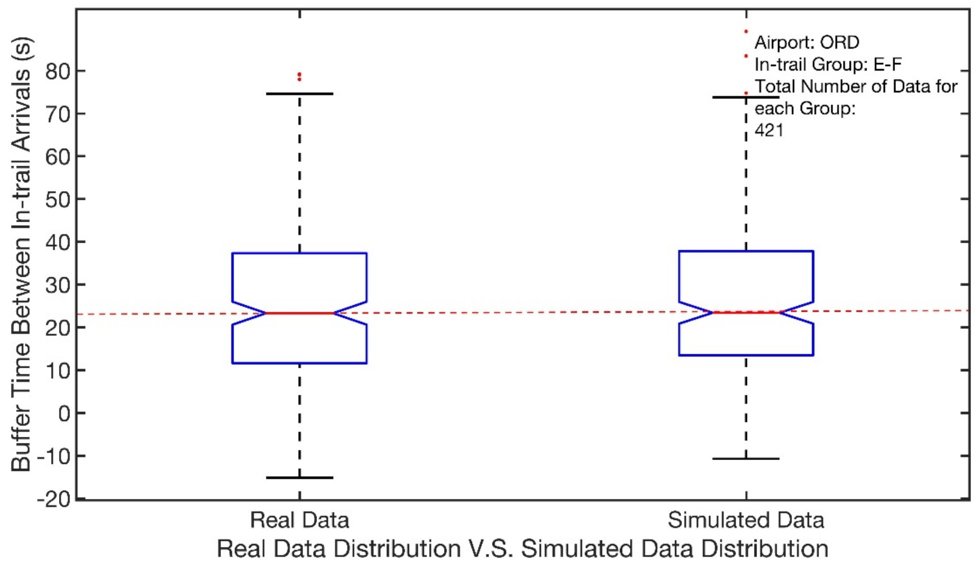


Figure 22 Comparison of box plot between real data and simulated data for E-F group at ORD

3.5 Converting factor between IMC and VMC

Pilots flying under VFR assume responsibility for their own separation from all other aircraft and are generally not assigned routes or altitudes by air traffic control (ATC). The minimum separation under VFR operations trends decreases comparing with IFR operations. A converting factor between IMC and VMC operations is introduced based on ASDE-X and ASPM data to estimate the separations and airport capacities under VFR operations.

The converting factor is estimated by the ratio between the typical time intervals between in-trail arrivals for IMC and VMC operations. Approach speeds and the buffer times are assumed to be independent of visibility conditions, the ratio between time intervals is equivalent to the ratio of distance separations for IMC and VMC operations. The minimum acceptable time intervals are defined as the peak time intervals from the kernel distribution plots based on ASDE-X data. Since the elevation and the airport configuration are unique for different airports, the conversion factor may change from airport to airport. Based on the ASDE-X data, the converting factor for five hub airports (ORD, CLT, DEN, ATL,) are analyzed in Figures 23 to 30.

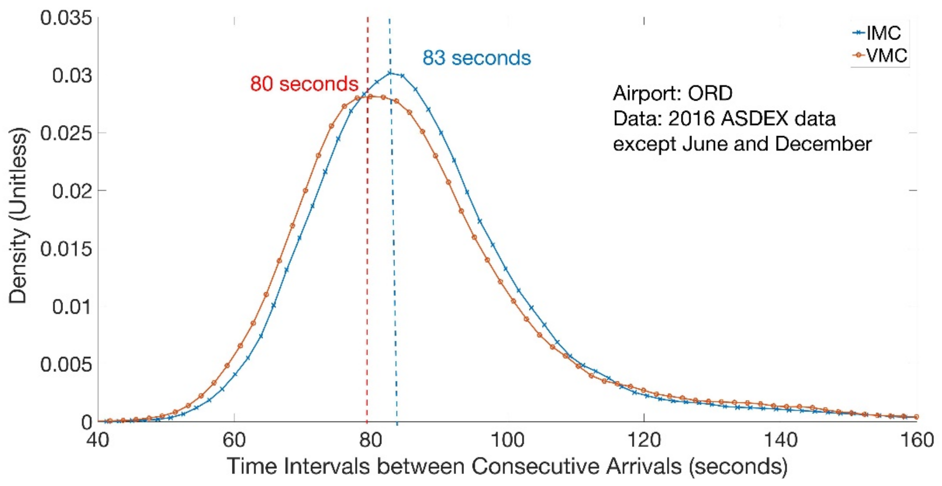


Figure 23 In-trail Time Interval Kernel Distribution during VMC and IMC at ORD

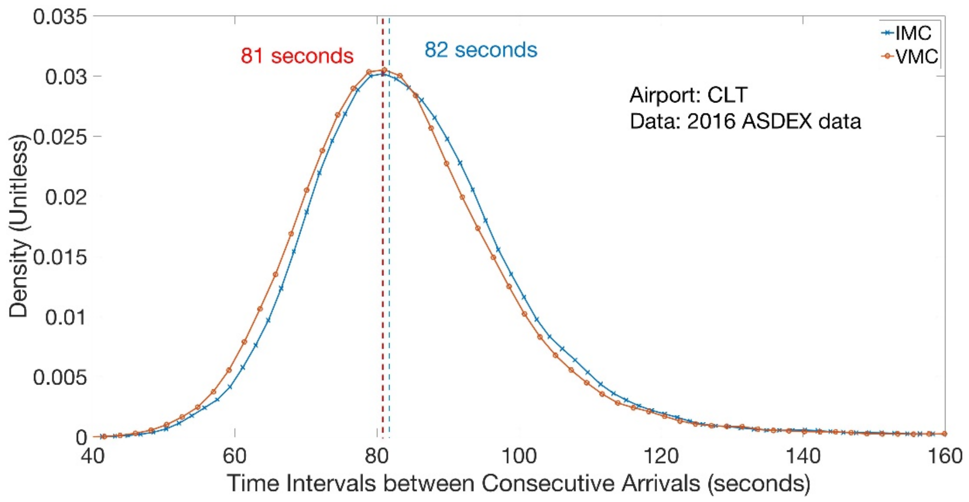


Figure 24 In-trail Time Interval Kernel Distribution during VMC and IMC at CLT

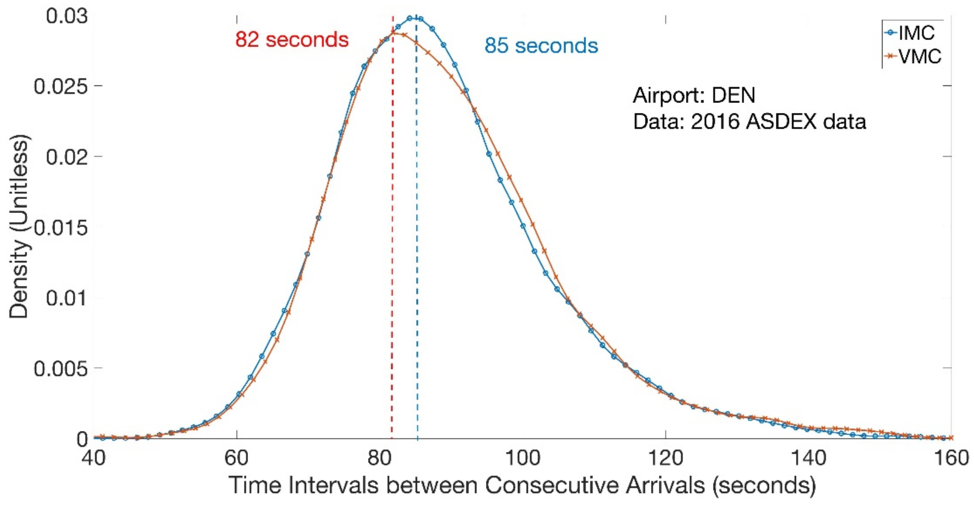


Figure 25 In-trail Time Interval Kernel Distribution during VMC and IMC at DEN

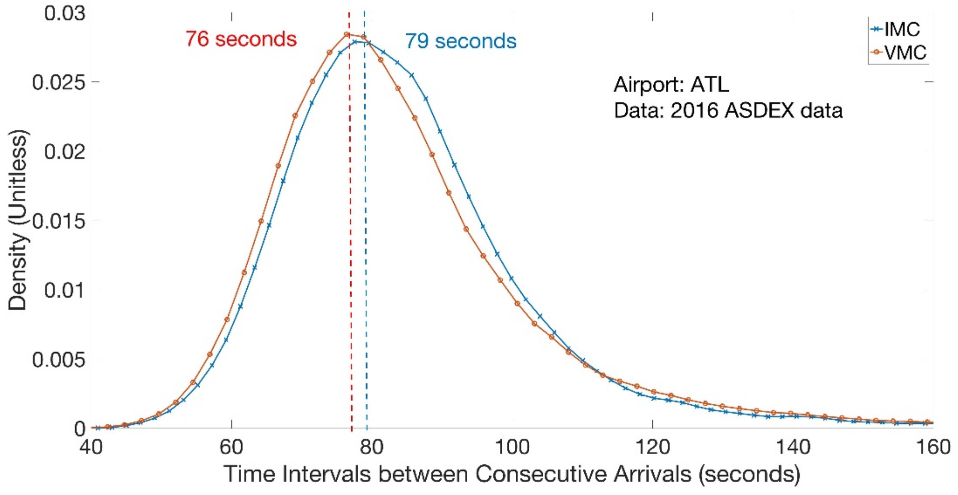


Figure 26 In-trail Time Interval Kernel Distribution during VMC and IMC at ATL

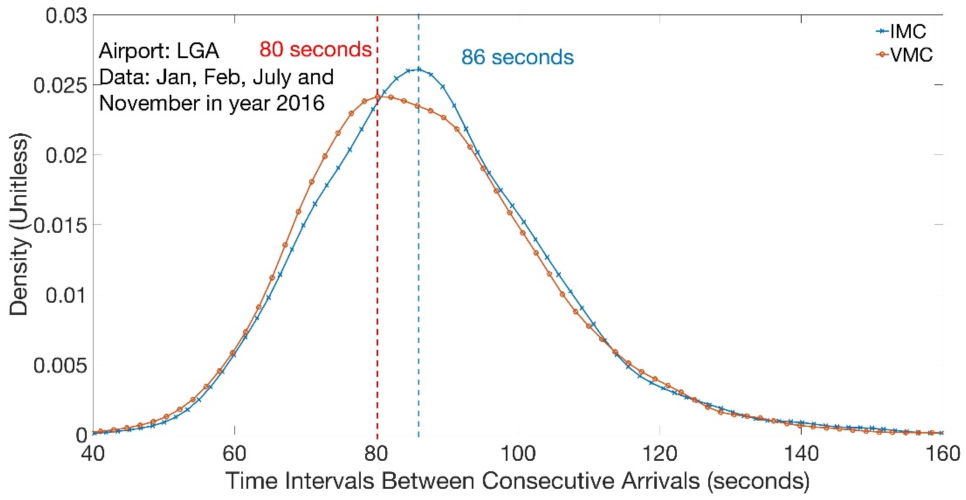


Figure 27 In-trail Time Interval Kernel Distribution during VMC and IMC at LGA

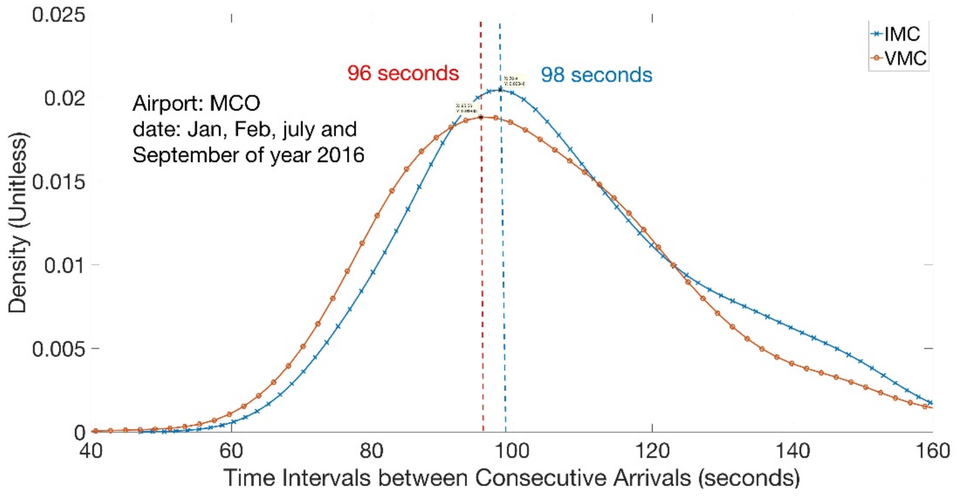


Figure 28 In-trail Time Interval Kernel Distribution during VMC and IMC at MCO

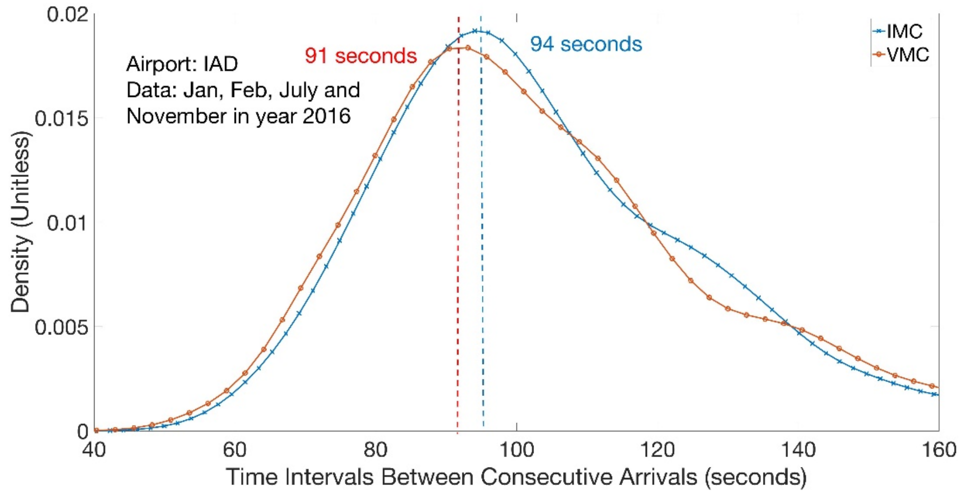


Figure 29 In-trail Time Interval Kernel Distribution during VMC and IMC at IAD

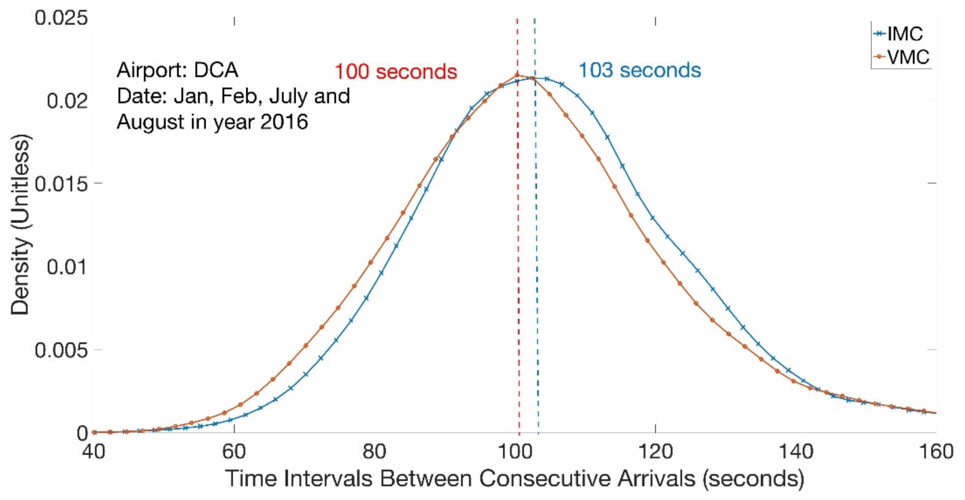


Figure 30 In-trail Time Interval Kernel Distribution during VMC and IMC at DCA

The kernel distribution of time intervals between successive arrivals show a slight difference between VMC and IMC operations during the peak period. This is as expected because the pilots still need to follow the minimum separation during VMC operations. The converting factor between IMC and VMC varies from airport to airport, and range from 0.93 to 0.99. The converting factor at ORD, DEN, ATL, and MCO are 0.96 while is 0.93 for LGA and 0.97 for IAD and DCA. The converting factor at CLT is 0.99 and 0.98 for MCO. The converting factor is very important to estimate the operations during VMC and will be used for estimating the airport capacity in the Quick Response Runway Capacity Model (RUNSIM).

4. Runway Occupancy Time Constraint and Runway Throughput Estimation under RECAT II

4.1 Improvements to Quick Response Runway Capacity Model

The Quick Response Runway Capacity Model (RUNSIM) is modified to simulate the runway throughputs and the percent of times ROT is the limiting factor in runway throughput under IMC and VMC conditions. The analysis is done for different separation rules and weather conditions. RUNSIM model improvements include: 1) revising the methodology to estimate the runway throughput; 2) updating the basic inputs as mentioned in Chapter 2; 3) adding an indicator to record when the ROTs constrain the separation between in-trail arrivals; and 4) incorporating an additional ROT separation rule to arrival operations.

4.1.1 Methodology to Estimate Single Runway Capacity in RUNSIM

RUNSIM estimates the runway throughput based on the simulations of a large number of ordered sequence of flights simulating arrivals that start at the final approach segment. For each in-trail arrival group, based on the aircraft types of leading and the following aircraft, the minimum separation is applied based on the separation rules described in Chapter 1. The approach speeds, ROTs are also generated for each aircraft based on the aircraft type and normalized distributions. For those parameters, RUNSIM has been modified to incorporate a kernel distribution of buffer times from real data based on different in-trail groups as explained in Chapter 3.

With these inputs, the error-free headway between in-trail arrivals during peak period can be simulated based on “opening” and “closing” cases shown in Figure 31. According to the FAA regulations, aircraft must maintain a minimum separation during the final approach segment. In RUNSIM, when the leading aircraft has a higher approach speed than the following aircraft, the distance between the in-trail arrivals increases when they approach the runway end. This instance is called as “opening” case. For the “opening” case, the minimum separation occurs at the entry gate of the final approach segment as shown in Figure 28. When the leading aircraft has a lower speed than the following aircraft, the minimum separation occurs at the runway threshold as the following aircraft is catching up with the leading aircraft. Theoretically, the time gap for "opening" case should be larger than the "closing" case because of the speed difference. Equations 3 and 4 are applied to capture the difference between “opening” and “closing” case in RUNSIM to estimate the error-free headway caused by wake vortex separation.

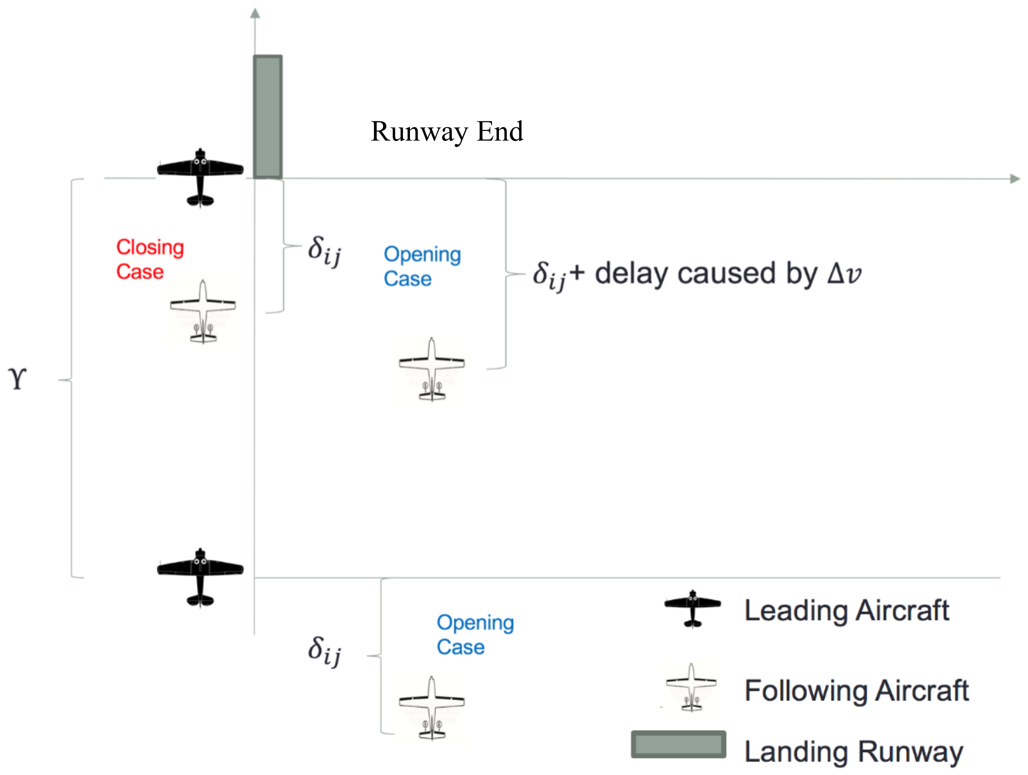


Figure 31 “Opening” and “Closing” Cases during the Final Approach

For “Closing” case:

$$T_{ij} = \frac{\delta_{ij}}{V_{j\_last2.5nm}} \quad (3)$$

For “Opening” case:

$$T_{ij} = \frac{\delta_{ij}}{V_{j\_last2.5nm}} + \gamma\left(\frac{1}{V_{j_{final}}} - \frac{1}{V_{i_{final}}}\right) \quad (4)$$

- T_{ij} is the error free headway based on wake vortex separation between leading aircraft i and following aircraft j (seconds)
- δ_{ij} is the minimum wake vortex separation between leading aircraft i and following aircraft j (seconds)

- γ is the approach length for final approach (knots)
- $V_{j\_last2.5nm}$ is the average approach speed for the last 2.5 nm for the following aircraft j (knots)
- $V_{j_{final}}$ is the average approach speed for final approach for the following aircraft j (knots)
- $V_{i_{final}}$ is the average approach speed for final approach for the leading aircraft i (knots)

In the real world, ATC controllers apply technical buffers to accommodate real operations. In RUNSIM, the actual headway (time between successive arrivals) is compared to the ROT and error-free headway as shown in equation 5. In reality, as the Air Traffic Tower Controller is adding a technical buffer as mentioned in Chapter 2, the actual headway should compare the ROT with the error-free headway plus the technical buffer as shown in Equation 6. In today's operations, the wake vortex separation is still the domination factor to runway throughput; the error-free headway is larger than the ROT for the leading aircraft. The default equation could still be used to estimate the runway capacity under RECAT I. However since RECAT II has decreased the minimum separation to 2 nautical miles, ROT becomes an important constraint to runway throughput. Then buffer time will play an important role in not only runway throughput estimation, but also at estimating the percent of the time that ROT limits runway throughput. In conclusion, Equation 6 is adapted instead of the default Equation 5.

Default equation in RUNSIM:

$$Actual\ Headway(s) = \max\{ROT\ (s), Errorfree\ Headway\ (s)\} + Buffer(s) \tag{5}$$

The updated equation in RUNSIM:

$$Actual\ Headway\ (s) = \max\{ROT\ (s), Errorfree\ Headway(s) + Buffer(s)\} \tag{6}$$

The headways between all in-trail aircraft created in the simulation is calculated, and the single runway throughput is therefore estimated analytically by using equation 7.

$$Capacity = \frac{1}{E(headway)} \tag{7}$$

4.2 Basic Inputs from ASDE-X Data for RUNSIM Simulation

The external inputs (ROT and approach speed) are also updated from the ASDE-X data at the airport to estimate better the throughput and the ROT limiting percentage for each runway end. Since the main target for RUNSIM is to estimate the overall airport capacity, the same default aircraft characteristic inputs are applied to all of the runway ends. As the configurations of the runways are unique, each runway end may have different types of exits, fleet mix and operation behaviors. To capture the difference between runways and to be more realistic, the inputs such as approach speed and ROTs are updated from the ASDE-X data as shown in Table 12.

Table 12 Sample Inputs for Runway 10C at ORD

Aircraft ID	A319	A320	B737	B738	B744	B763	B772	CRJ2	CRJ7	E135	E145	E170	MD82	MD83
Fleet Mix(%)	4	13	2	21	2	3	3	5	14	2	18	8	2	3
Mean ROT(s)	58	58	56	57	74	61	63	54	53	51	52	54	57	57
ROT std.	7.2	7.5	7.4	7.4	10.9	8.5	9.4	6.6	6.9	6.6	6.0	6.9	7.0	6.6
90 Percentile ROT (s)	67	67	65	67	88	72	75	62	62	59	60	62	66	66
95 Percentile ROT (s)	70	71	69	71	92	76	81	64	65	62	63	65	68	69
Approach Ground Speed Final Approach (knots)	150	154	153	158	162	154	155	152	152	153	155	151	153	153
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	153	138	138	134	135	132	134	131	134	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	10.2	8.9	9.4	8.4	8.4	8.6	8.8	8.4	8.6	8.6

4.2.1 Average Approach Speed per Aircraft Type

Two approach speeds and their standard deviation are incorporated into RUNSIM for each aircraft instead of the deterministic value of approach speed in RUNSIM. The default values were created by combining the Basic of Aircraft Data (BADA) Reporting System (PDARS) data. ASDE-X data provides second by second for aircraft position information. According to the Equations 3 and 4, there are two approach speed: average approach speed for minimum wake vortex separation (δ_{ij}) and average approach speed for final approach segment (γ). As the minimum wake vortex separation is reduced to 2.5 nautical miles under RECAT I and 2 nautical miles for RECAT II, which applies for most of the cases studied, the average approach speed for last 2.5 nautical miles is selected with a normalized standard deviation from ASDE-X data. Since the average speed during final approach segment is correlated with the average speed for last 2.5 nautical miles, a linear regression is used to estimate the approach speed for the final approach segment.

The linear regression for the 33 most frequently operated (more than 1,000 operations) aircraft at ORD and CLT are shown in Figure 32 and Figure 33.

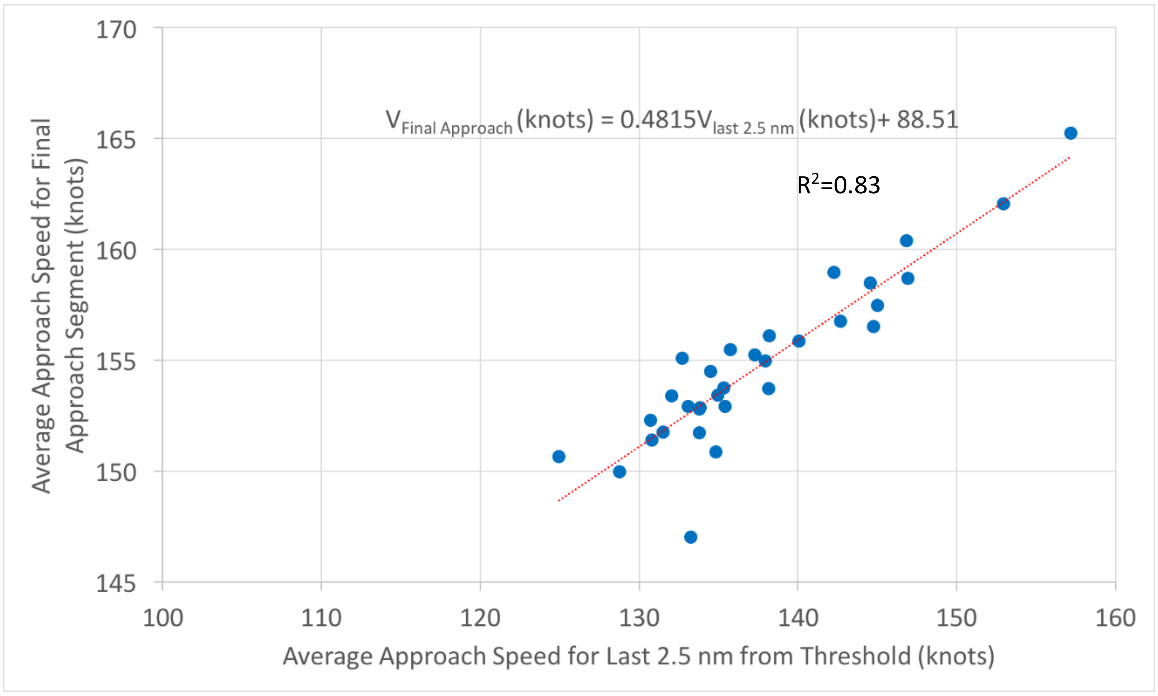


Figure 32 Linear Regression between Average Approach Speed for Final Approach and Last 2.5 nm for 33 Aircraft Types at ORD

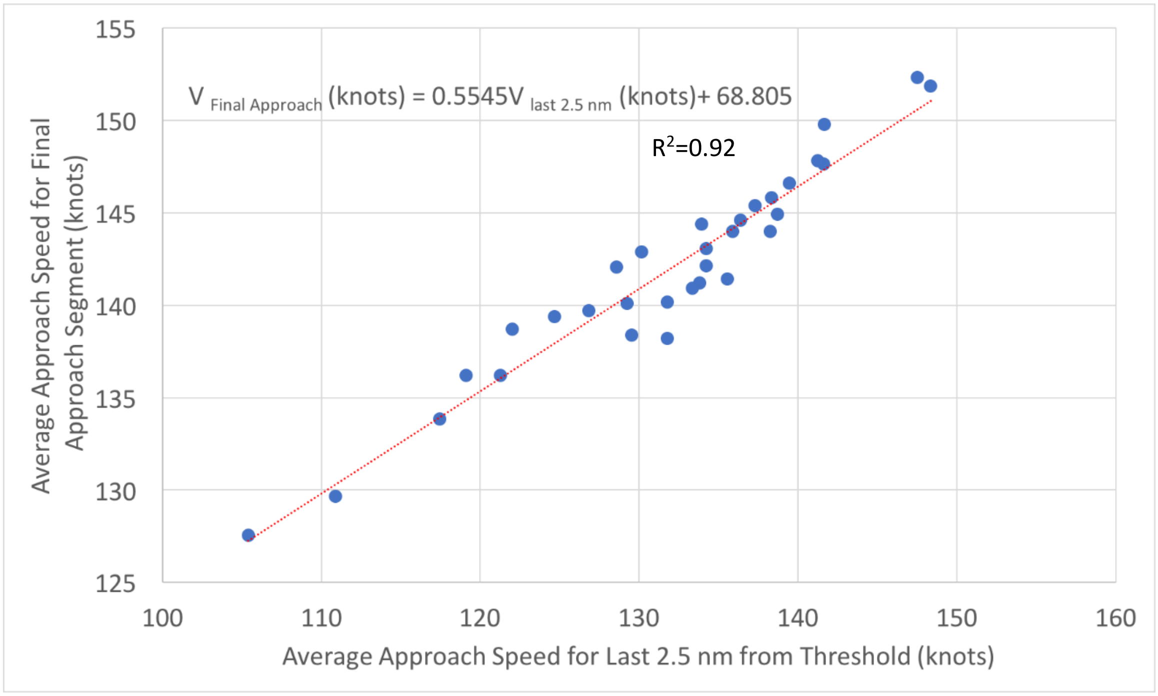


Figure 33 Linear Regression between Average Approach Speed for Final Approach and Last 2.5 nm for 33 Aircraft Types at CLT

4.2.2 Runway Occupancy Time by Each Aircraft Type per Each Runway

The study shows the ROTs should be aircraft-type and runway-end specified. Since each runway is unique in the runway exits location and design, the length of the runway, and the fleet mixes; the same aircraft could have different ROTs for different runway ends. Figure 34 shows the cumulative ROT density plot for Airbus 320 (A320) at ORD 9L, 10C, 10R and 10L. Figure 34 indicates different ROT distributions for different runway ends. Based on the runway configurations shown in Figure 35, Runway 10R/28L has multiple high-speed exits which explains the low values of ROT. The runway length also plays a key factor to ROTs. Runway 10C/28C and Runway10L/28R have relatively large runway lengths which may increase the ROTs and the standard deviations. Notice Runway 10L/28R at ORD is mainly used for departures, and the large ROTs will not influence the arrival operations. The split of ROTs based on runway ends will not only benefits the runway capacity estimation but also vital to estimating the number of times ROT will limit runway throughput capacities.

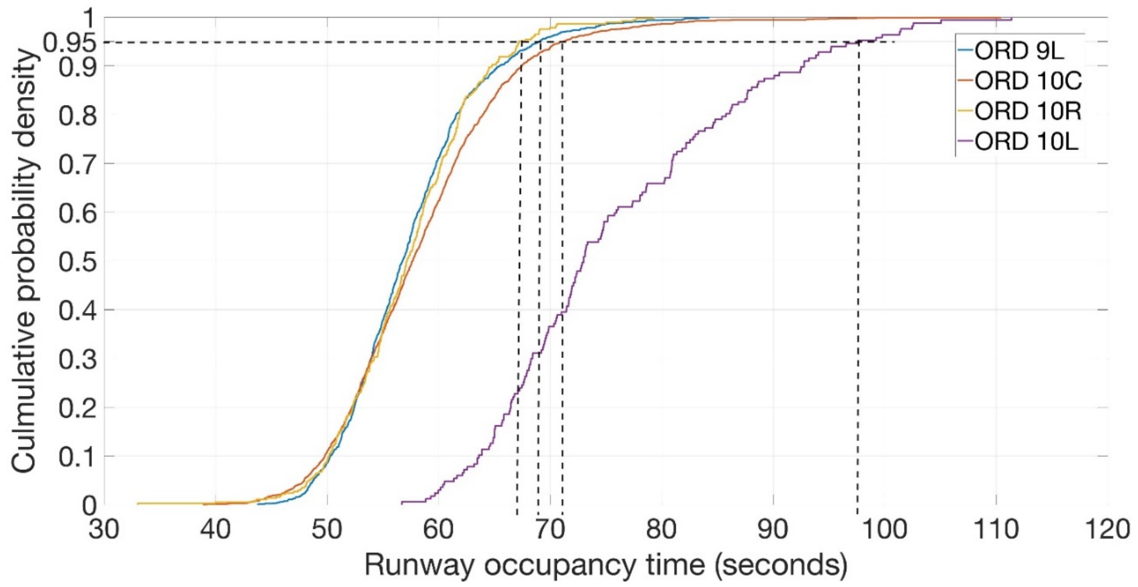


Figure 34 Normalized Runway Occupancy Time Histogram plot for Aircraft A320 at Runway 9L, 10C, 10R, and 10L

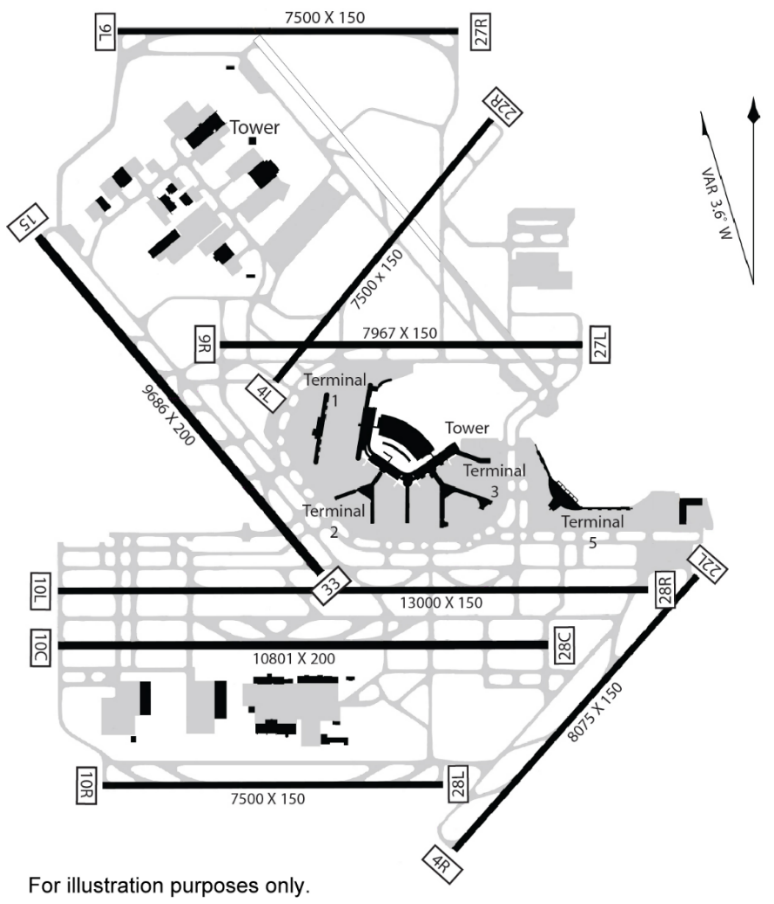


Figure 35 Runway Configurations at ORD (Source: Airnav.com).

4.3 ROT Operations Limited by ROT under RECAT I at ORD for IMC and VMC

The runway throughput is influenced by two main factors: minimum wake separation and runway occupancy time (ROT). However, as the current wake vortex separation still dominates the separations between in-trail arrivals, the ATC does not apply a ROT constraint to separate arrivals. With the development of reduced wake vortex separation rules, ROT could become a vital factor to runway capacity in the near future.

The study simulates the number of times ROT limits runway throughput under RECAT I during peak period under both IMC and VMC conditions by RUNSIM. For each in-trail group, the ROT could limit runway throughput when the leading aircraft ROT exceeds wake vortex separation (the equivalent minimum separation time plus buffer). In RUNSIM, based on the distribution of ROTs, approach speeds for last 2.5 nautical miles and buffer times, we simulates 4,000 in-trail groups for each runway end.

A flag indicator is incorporated in RUNSIM to record the cases when ROT limits the runway throughput as shown in Figure 36. The RUNSIM simulation result for the major used runway ends at ORD is shown in Table 13.

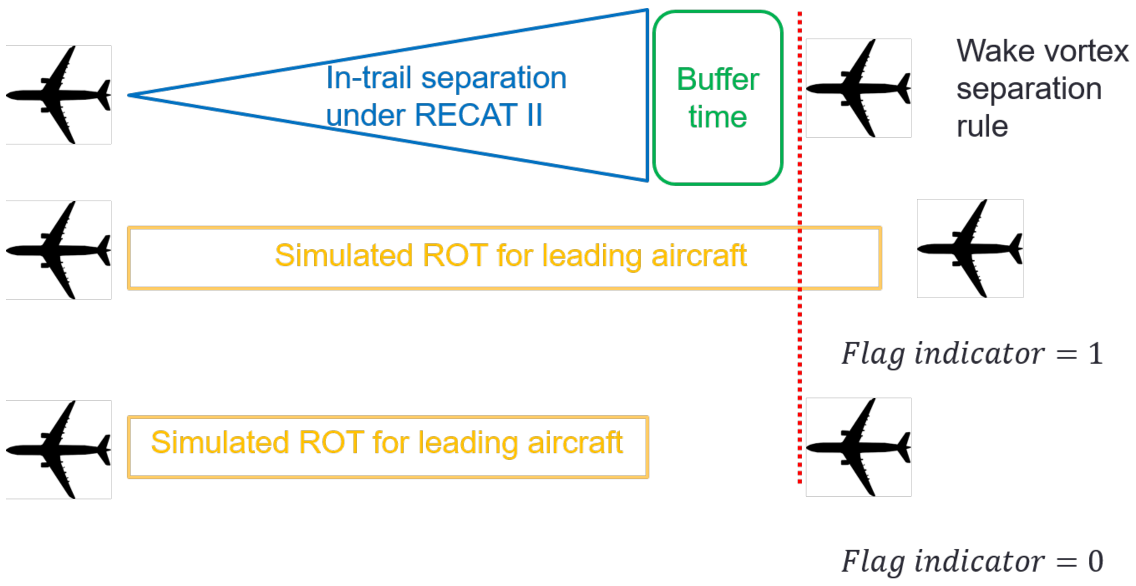

In-trail separation under RECAT II
Buffer time
Wake vortex separation rule
Simulated ROT for leading aircraft
$Flag\ indicator = 1$
Simulated ROT for leading aircraft
$Flag\ indicator = 0$

Figure 36 Flag Indicator is introduced to Record the Cases when ROT Limits the Runway Throughput in RUNSIM.

Table 13 RUNSIM Simulation Results with RECAT I Separation for the Main Runway Ends at ORD.

Runway end		Capacity (arr/hr)		ROT Limiting Throughput Percentage (%)	
		IMC	VMC	IMC	VMC
West flow	28C	37.4	39.3	1.4	1.8
	27R	40.0	42.0	1.4	2.2
	27L	39.9	42.0	0.9	1.2
East flow	09L	40.0	42.1	1.8	2.4
	10C	38.9	40.5	1.4	2.0
	10R	40.0	41.9	0.9	1.3
Not frequently used	09R	40.3	41.2	1.5	2.1
	※22R	39.3	40.1	7.1	8.5
	※28R	37.8	38.5	11.4	13.8
	※10L	36.4	38.3	19.8	26.5

※ Means these runways are not operated at capacity

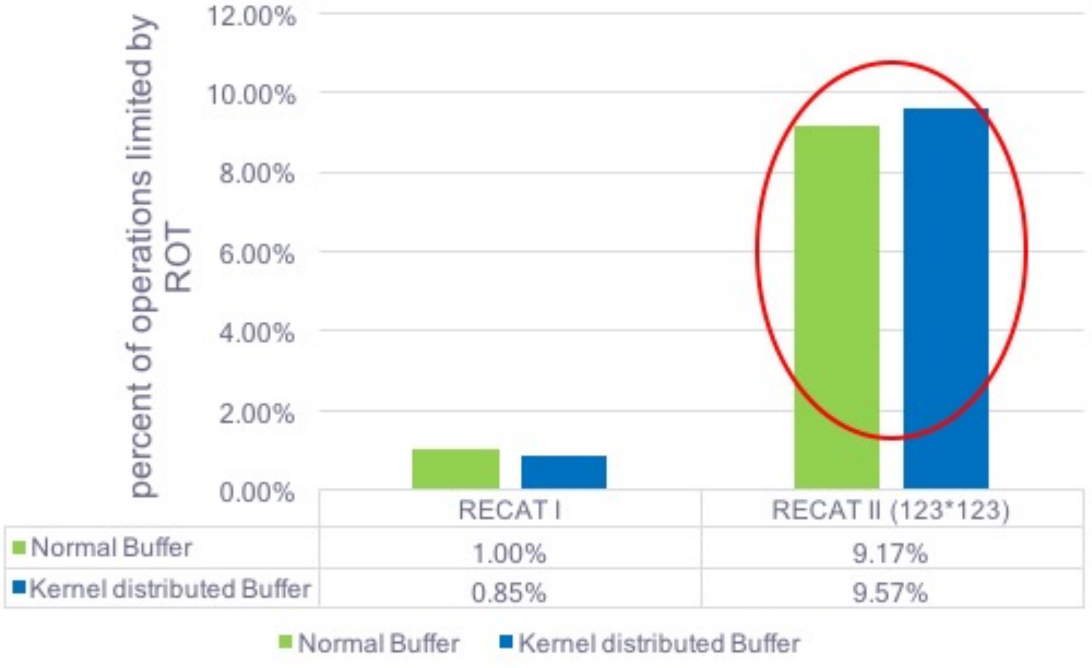
The simulation results show that the throughput for a single runway can reach up to 40 arrivals per hour during IMC at ORD, which matches the observations from ASDE-X data as shown in Figure 10; each runway end can gain one or two arrivals per hour during VFR operations comparing to IFR operations. The ROT limiting percentage shows that during peak period, for west flow and east flow runway ends at ORD, more than 98% of the times, the wake vortex separation determines the minimum separation, which matches current operations. For those not frequently used the runway, however, the ROTs plays an important role in throughput even under RECAT I. This is caused by runway geometry. For example, runway 28R/10L is the longest runway at ORD as shown in Figure 31, which leads to the larger randomness of ROTs. In reality, most of the time runway 28R/10L is used for departures and used for arrivals only during off-peak periods. For Runway 22R, which is only used under strong wind conditions, only has one high-speed exit locates far away from threshold as shown in Figure 30, which leads to the large ROTs. This study shows that the ROTs will limit the throughput if Runway ends 22R, 28R, and 10L are used during the peak period.

4.4 RECAT II Runway Throughput With ROT Constraints at ORD and CLT

This section analyzes the runway capacities and the percent of operations limited by ROT under RECAT II for each runway end at ORD and CLT. With the development of reduced wake vortex separation, an additional separation rule is suggested to put RECAT II in operation. The target for this analysis is to estimate the potential gains from the RECAT II with insignificant changes in ROT limiting percentage.

4.4.1 ROT limitations under RECAT II at ORD under IMC

The RUNSIM simulation shows near 10% of the cases ROT will limit runway throughput under RECAT II during peak period under IMC condition at ORD as shown in Figure 33. The simulation result also indicates the distribution of buffer time plays an important role in estimating ROT limiting percentage. The simulation shows 0.4% difference in ROT limiting estimation between buffer time with kernel distribution and normal distribution. Since the near 10% of ROT limiting percentage will not be acceptable operationally, a new ROT-based separation rule is suggested to the system to inform ATC controllers.



	RECAT I	RECAT II (123*123)
🟩 Normal Buffer	1.00%	9.17%
🟦 Kernel distributed Buffer	0.85%	9.57%

Figure 37 The ROT Limiting Percentage under RECAT I and RECAT II with the Kernel and Normal Distributed Buffer Time for ORD.

4.4.2 The Suggestion of an Additional ROT-based Separation Rule

The study suggests an additional ROT-based separation rule besides the wake vortex separation. The additional ROT-based separation rule aims to deal with the potential conflicts between reduced wake vortex separation and leading aircraft runway clearance and should be able to cover the majority of the possible ROTs. The “new” in-trail separation will use the maximum of wake vortex separation rule and additional ROT-based separation rule as shown in Figure 38.

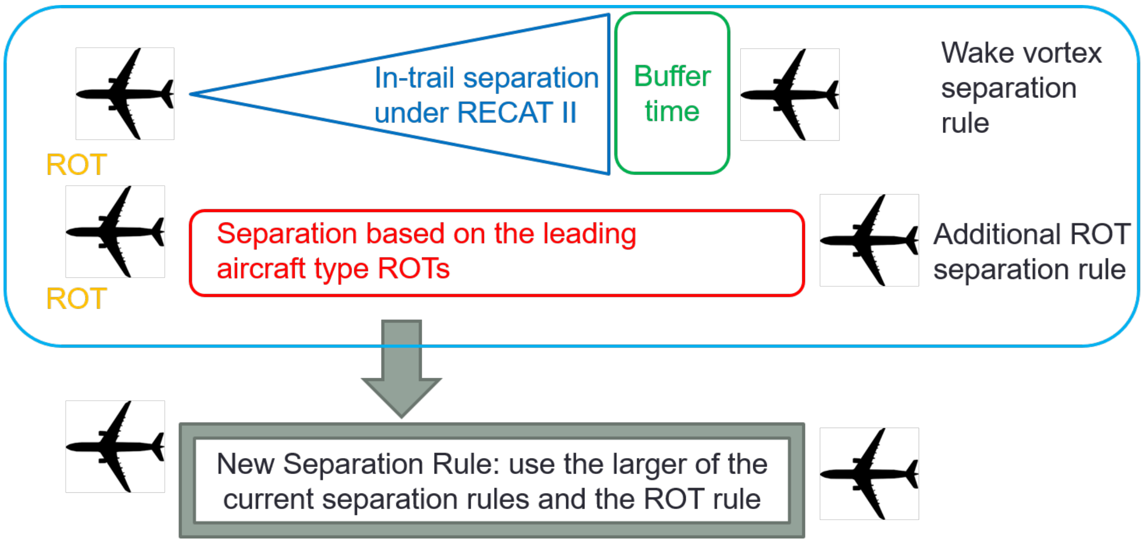

In-trail separation under RECAT II
Buffer time
Wake vortex separation rule
ROT
ROT
Separation based on the leading aircraft type ROTs
Additional ROT separation rule
New Separation Rule: use the larger of the current separation rules and the ROT rule

Figure 38 The Methodology of the Suggested Separation Rule with the Additional ROT Separation Rule Consideration.

Two standards of the additional separation rule are tested for this research: 90 percentile and 95 percentile ROT separation rule. For each runway end, the 90 percentile and 95 percentile value of ROTs for each aircraft type are used as the additional separation rule based on the year 2016 ASDE-X data. Figure 35 shows a sample of 90 and 95 percentile ROT value based on the histogram of ROTs for Airbus 320 (A320) at ORD Runway 10C. The modified flag indicator is used to detect the cases when the ROT limits the runway throughput under the "new" separation rule as shown in Figure 39.

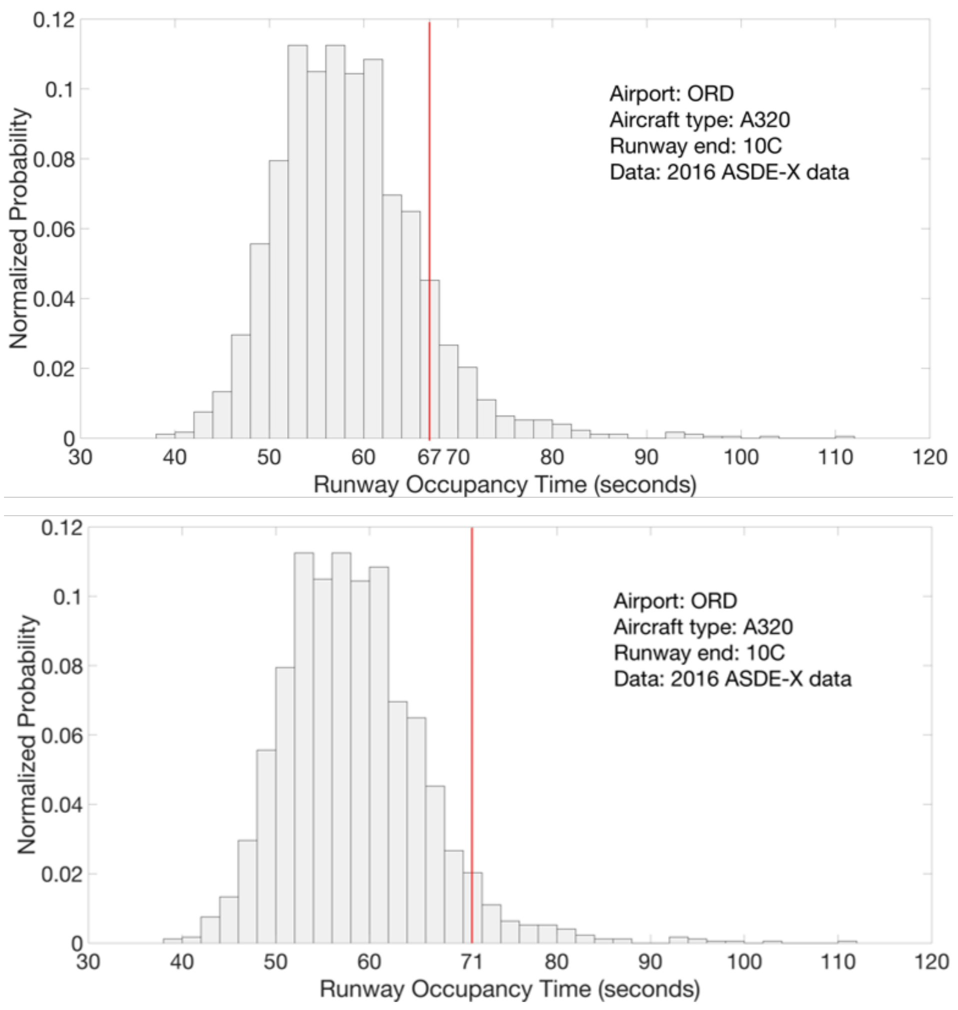


Figure 39 Sample 90 and 95 Percentile ROT value for A320 at ORD Runway 10C Based on the Year 2016 ASDE-X Data.

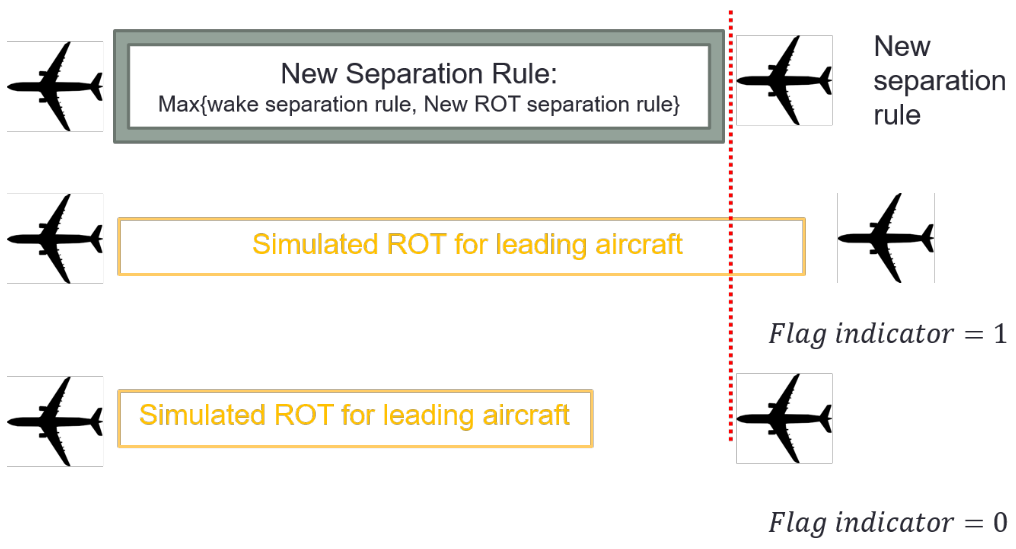

New Separation Rule:
Max{wake separation rule, New ROT separation rule}
New separation rule
Simulated ROT for leading aircraft
Flag indicator = 1
Simulated ROT for leading aircraft
Flag indicator = 0

Figure 40 Modified Flag Indicator under the Suggested Separation Rule.

4.4.3 ORD 28C, 27R, 27L

For ORD west flow, the runway capacities and the probability of operations limited by ROT are simulated for mainly used runway ends (28C, 27R, and 27L) based on several inputs: 1) approach speed for last 2.5 nautical miles per each aircraft type; 2) ROT per each aircraft type per each runway end; 3) fleet mix per each runway end; 4) minimum separation matrix. The inputs for each runway end and the simulation results are shown from Table 14 to Table 16 and Figure 41 to Figure 43.

Table 14 Inputs from ASDE-X data for ORD 28C.

Aircraft ID	A319	A320	B737	B738	B744	B763	B772	CRJ2	CRJ7	E135	E145	E170	MD83
Fleet Mix(%)	4	10	1	19	6	4	4	5	14	2	17	8	6
Mean ROT(s)	57	57	56	57	71	58	60	54	54	54	53	54	56
ROT std.	6.36	6.60	7.10	6.78	11.77	7.63	8.34	5.96	6.37	6.43	6.40	6.05	6.01
90 Percentile ROT (s)	65	66	65	65	87	68	71	62	62	62	61	62	63
95 Percentile ROT (s)	68	69	68	68	91	72	76	65	65	65	64	65	66
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	153	138	138	134	135	132	134	131	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	10.2	8.9	9.4	8.4	8.4	8.6	8.8	8.4	8.6

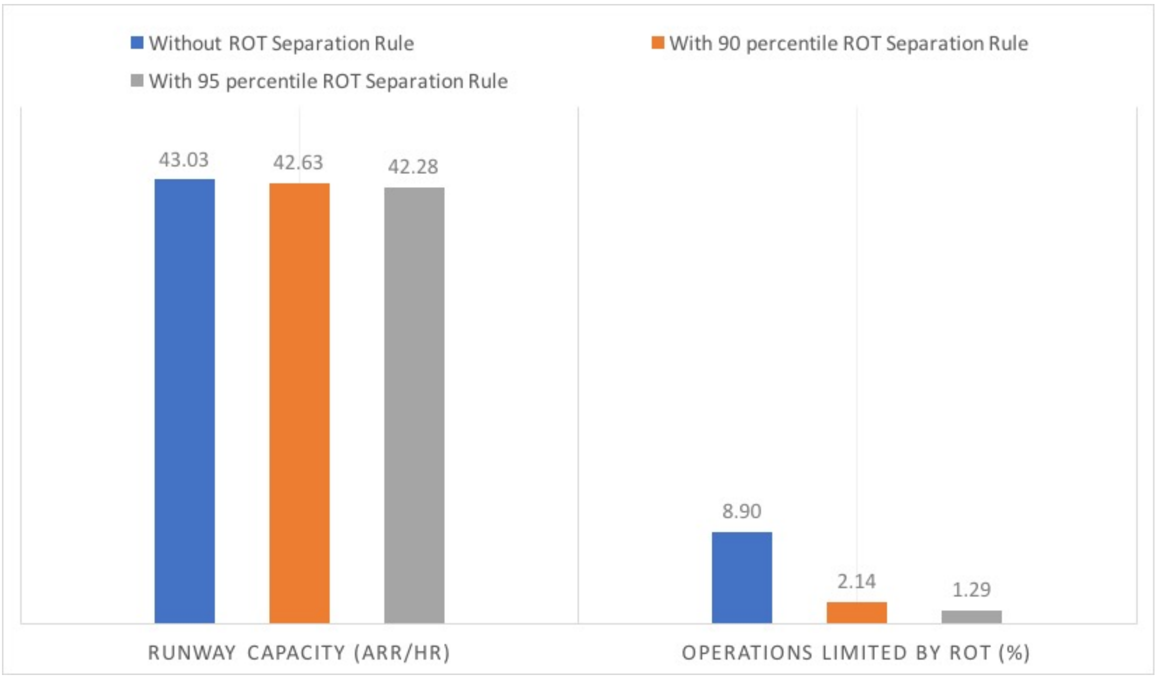


Figure 41 Runway Capacity & ROT Limiting Percent under RECAT II for ORD 28C.

Table 15 Inputs from ASDE-X data for ORD 27R.

Aircraft ID	A319	A320	B738	CRJ2	CRJ7	E135	E145	E170
Fleet Mix(%)	5	8	14	11	22	4	26	10
Mean ROT(s)	59	57	57	58	58	56	56	58
ROT std.	6.28	6.22	6.78	5.89	6.54	5.77	6.09	6.27
90 Percentile ROT (s)	68	65	66	65	67	64	64	66
95 Percentile ROT (s)	71	68	69	68	71	67	67	69
Approach Ground Speed Last 2.5 nm (knots)	129	135	145	134	135	132	134	131
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	9	8.4	8.4	8.6	8.8	8.4

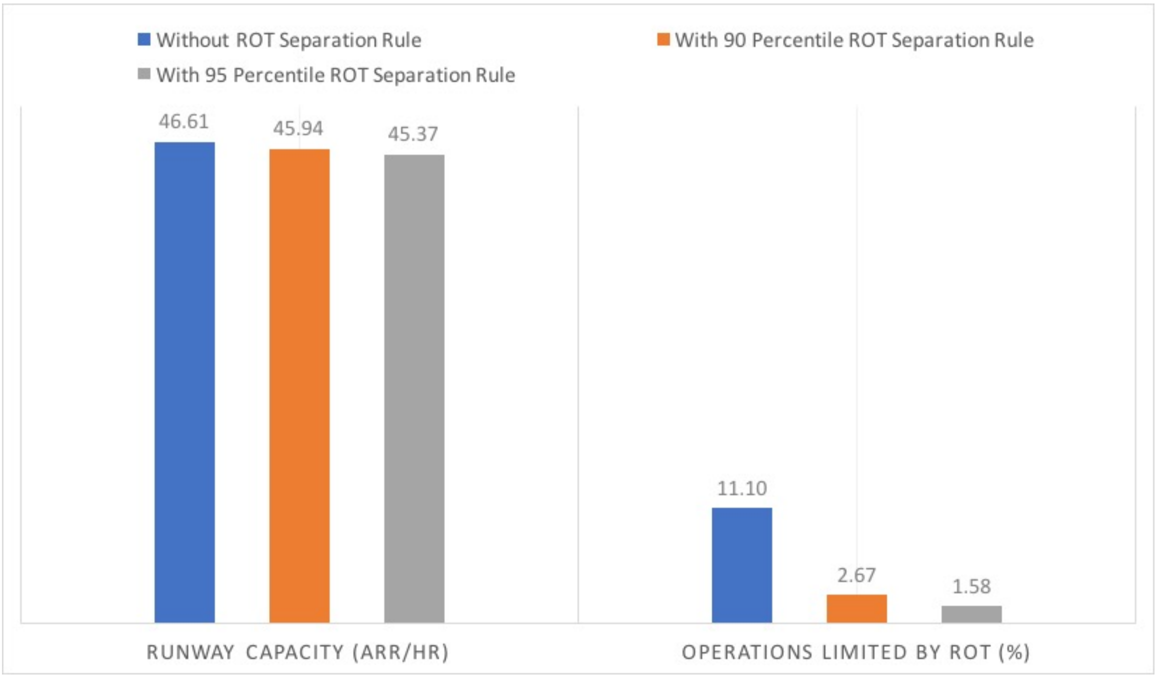


Figure 42 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 27R.

Table 16 Inputs from ASDE-X Data for ORD 27L.

Aircraft ID	A319	A320	B738	CRJ2	CRJ7	E135	E145	E170	MD83
Fleet Mix(%)	5	14	23	6	15	2	22	11	2
Mean ROT(s)	54	55	54	49	49	48	48	51	54
ROT std.	6.9	7.9	8.17	6.33	6.30	5.96	6.06	6.19	7.59
90 Percentile ROT (s)	63	65	65	57	58	56	56	59	63
95 Percentile ROT (s)	67	70	69	60	61	59	59	62	68
Approach Ground Speed Last 2.5 nm (knots)	129	135	145	134	135	132	134	131	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	9	8.4	8.4	8.6	8.8	8.4	8.6

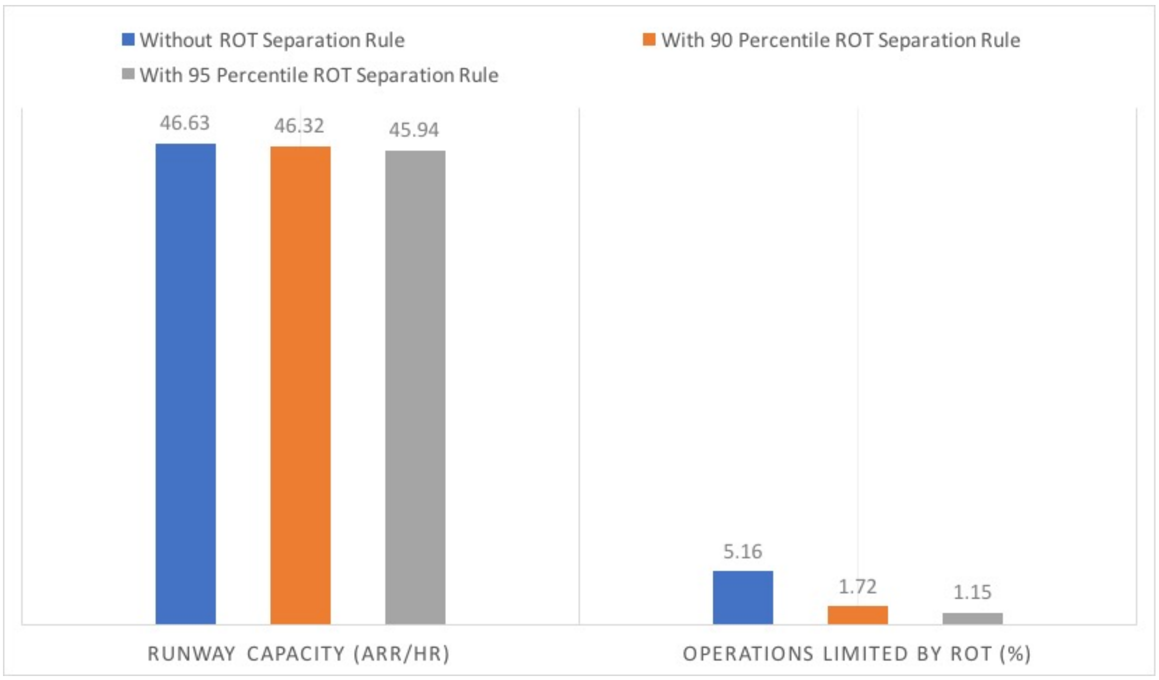


Figure 43 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 27L.

For the west flow at ORD, the ROT constraint has slightly influence to runway throughput. This lies in the good runway exits design and runway length at runway ends 28C, 27R, and 27L. Most of the aircraft can clear the runway within 60 seconds for these runway ends, and almost 95% of them can clear the runway within 70 seconds, which is still less than the error-free headway under RECAT II operations. Runway end 28C has less throughput comparing to Runway end 27R and 27L because of the fleet mix. Some of the heavy aircrafts such as B744 and B772 lands on Runway 28C as it has a relatively longer length. As shown in Table 18, these heavy aircraft need more time to clear the runway and more separation distance because of the wake vortex. Regarding operations limited by ROT, almost or more than 90% percent of the time, the ROTs still not affect the throughput. Runway end 27R has a relative higher ROT limiting percent because it has only one high-speed exit near runway end, which increase the ROTs. With a constraint of 95 percentile ROT constraint, this number decreases to less than 2%, which is close to the limiting percentage under current operations.

4.4.4 ORD 09L, 09R, 10C, 10R

For ORD east flow, the runway capacities and the probability of ROT constraints are simulated for mainly used runway end (09L, 09R, 10C and 10R) based on the inputs from ASDE-X data. The inputs for each runway end and the simulation results are shown from Table 17 to Table 20 and Figure 37 to Figure 40.

Table 17 Inputs from ASDE-X Data for ORD 09L.

Aircraft ID	A319	A320	B737	B738	CRJ2	CRJ7	E135	E145	E170	MD83
Fleet Mix(%)	5	8	2	20	10	15	3	24	11	2
Mean ROT(s)	60	58	62	58	58	59	57	57	58	58
ROT std.	5.62	5.82	6.33	6.53	5.57	5.98	5.58	5.75	5.63	6.82
90 Percentile ROT (s)	67	65	70	67	65	66	64	64	65	68
95 Percentile ROT (s)	70	69	73	70	68	69	67	67	68	71
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	134	135	132	134	131	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	8.4	8.4	8.6	8.8	8.4	8.6

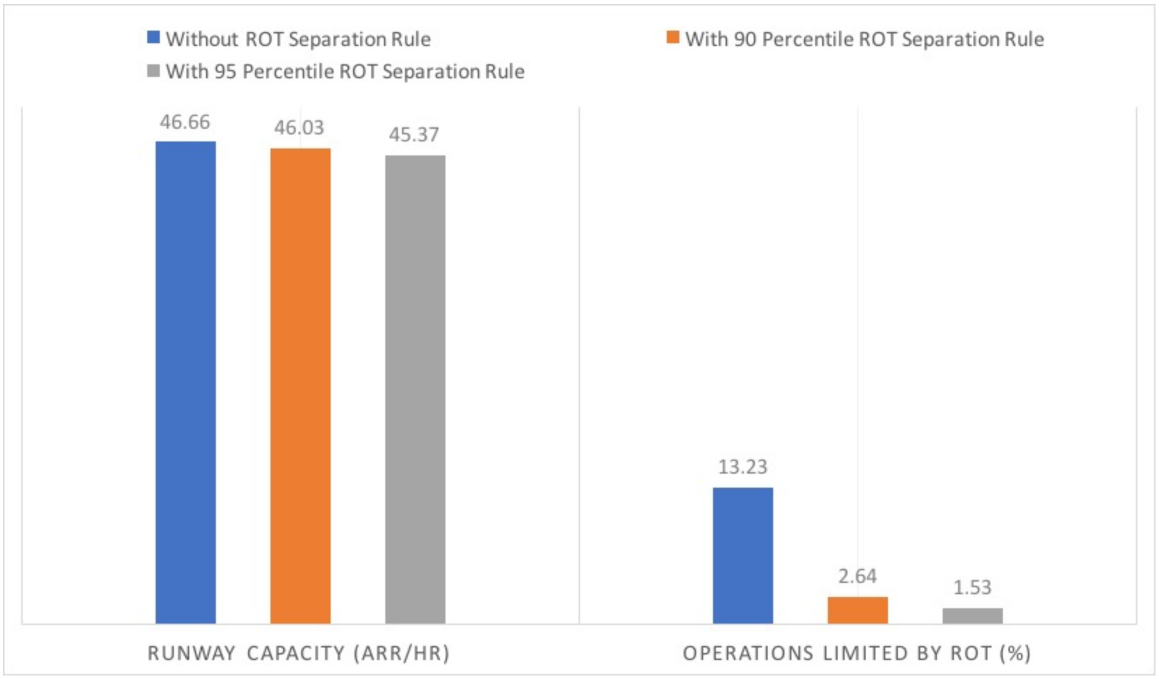


Figure 44 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 09L.

Table 18 Inputs from ASDE-X Data for ORD 10C.

Aircraft ID	A319	A320	B737	B738	B744	B763	B772	CRJ2	CRJ7	E135	E145	E170	MD82	MD83
Fleet Mix(%)	4	13	2	21	2	3	3	5	14	2	18	8	2	3
Mean ROT(s)	58	58	56	57	74	61	63	54	53	51	52	54	57	57
ROT std.	7.18	7.45	7.38	7.44	10.93	8.48	9.41	6.64	6.90	6.59	6.02	6.94	6.96	6.55
90 Percentile ROT (s)	67	67	65	67	88	72	75	62	62	59	60	62	66	66
95 Percentile ROT (s)	70	71	69	71	92	76	81	64	65	62	63	65	68	69
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	153	138	138	134	135	132	134	131	134	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	10.2	8.9	9.4	8.4	8.4	8.6	8.8	8.4	8.6	8.6

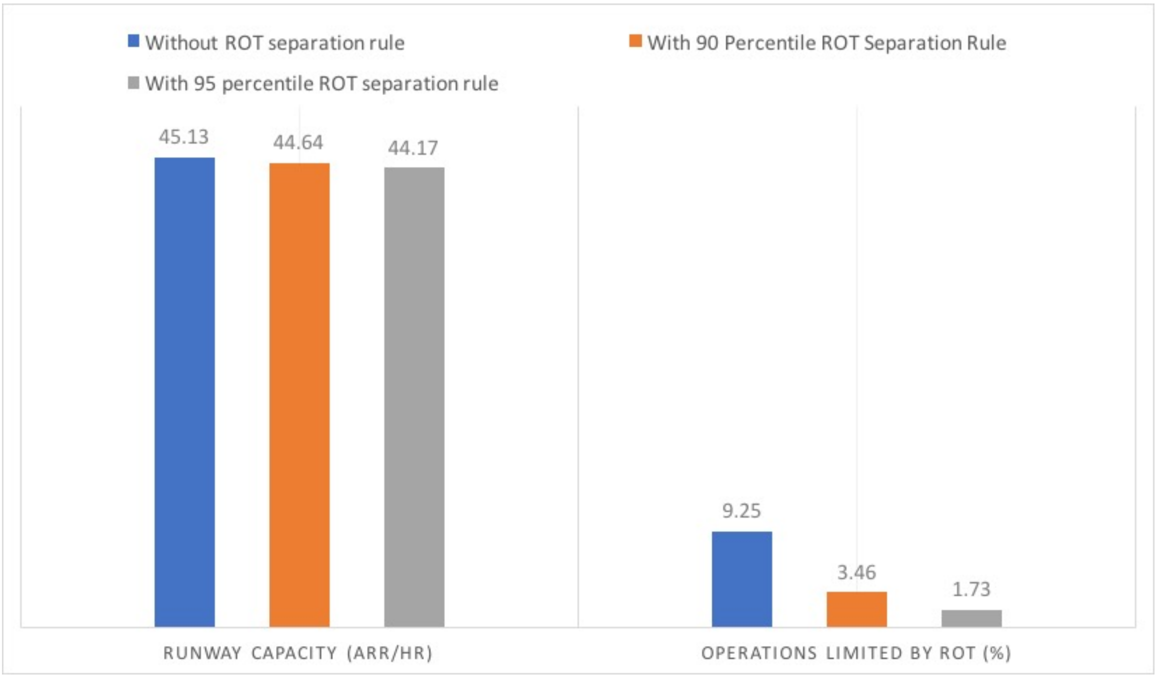


Figure 45 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10C.

Table 19 Inputs from ASDE-X Data for ORD 10R.

Aircraft ID	A319	A320	B738	CRJ2	CRJ7	E135	E145	E170	MD82
Fleet Mix(%)	5	12	14	5	16	3	25	18	2
Mean ROT(s)	58	58	56	54	54	52	53	54	57
ROT std.	6.14	5.71	5.92	5.23	5.90	5.97	5.97	5.64	5.79
90 Percentile ROT (s)	66	65	64	61	61	60	60	61	64
95 Percentile ROT (s)	68	67	66	63	64	63	63	64	67
Approach Ground Speed Last 2.5 nm (knots)	129	135	145	134	135	132	134	131	134
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	9	8.4	8.4	8.6	8.8	8.4	8.6

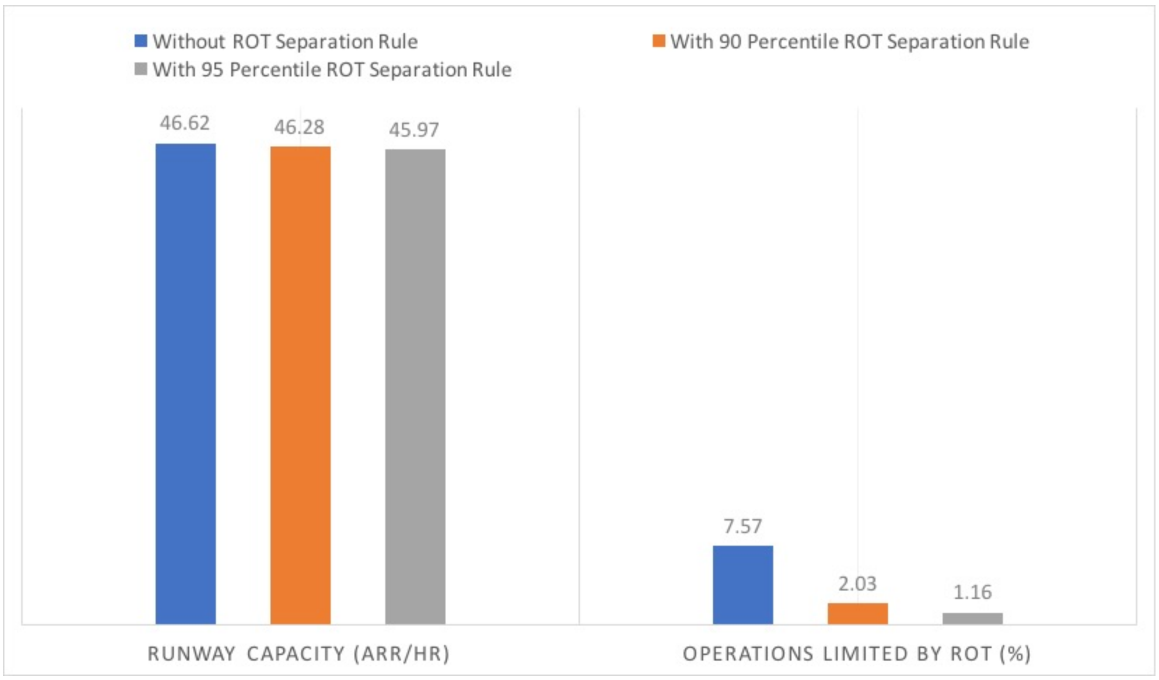


Figure 46 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10R.

Table 20 Inputs from ASDE-X Data for ORD 09R.

Aircraft ID	A319	A320	B738	CRJ2	CRJ7	E145	E170	MD82	MD83
Fleet Mix(%)	4	9	26	8	16	23	8	2	4
Mean ROT(s)	58	57	57	55	55	53	54	56	57
ROT std.	6.93	8.47	9.51	5.48	6.80	7.16	6.74	8.05	7.43
90 Percentile ROT (s)	67	67	67	62	63	62	63	67	65
95 Percentile ROT (s)	72	75	72	64	67	66	68	74	69
Approach Ground Speed Last 2.5 nm (knots)	129	135	145	134	135	134	131	134	135
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	9	8.4	8.4	8.8	8.4	8.6	8.6

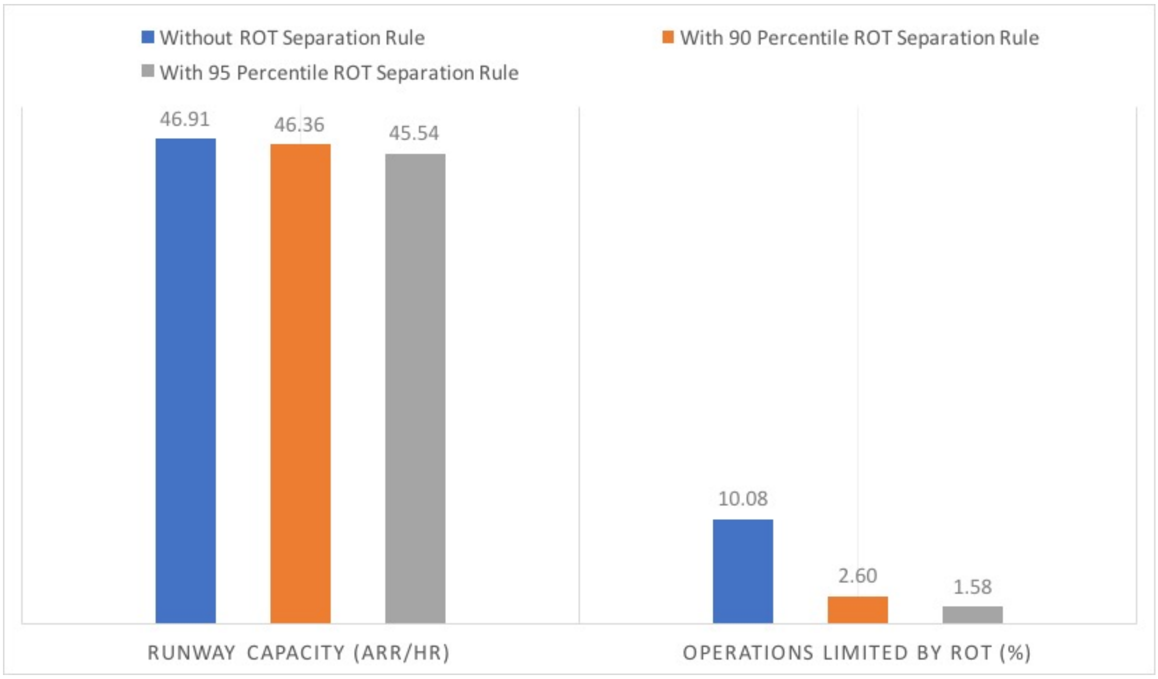


Figure 47 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 09R.

For the east flow at ORD, runway throughput lose nearly one arrival per hour with a 95 percentile ROT separation rule. For these mainly used runway ends (09L, 10C, 10R and 09R) for east flow at ORD, the runway configurations are well designed so that the ROTs will not limit the throughput even under RECAT II operations. Runway 10C has one arrival less per hour for capacity because of heavy aircraft operations. Runway end 09L has a relative higher ROT limiting percentage because it has only one high-speed exit near runway end, which increase the ROTs. With a constraint of 95 percentile ROT constraint, the ROT limiting percentage decrease to less than 2%, which is close to the limiting percentage under current operations.

4.4.5 ORD 22R

Runway 22R is used only during certain wind conditions. It is important to simulate the runway throughput under RECAT II because the arrival capacity mainly rely on Runway 22R and Runway 22L (Runway 22L is not analyzed for the lack of recorded operations). The inputs and the simulation results are shown in Table 21 and Figure 48.

Table 21 Inputs from ASDE-X Data for Runway 22R.

Aircraft ID	A319	A320	B738	B753	B772	CRJ2	CRJ7	DC10	E145	E170
Fleet Mix(%)	3	23	35	6	3	4	8	2	11	4
Mean ROT(s)	56	61	63	64	71	53	54	81	51	54
ROT std.	10.78	13.98	14.96	16.06	14.52	9.53	9.83	9.41	8.60	9.06
90 Percentile ROT (s)	67	81	84	88	89	64	65	95	62	68
95 Percentile ROT (s)	75	85	89	94	95	75	74	97	68	75
Approach Ground Speed Last 2.5 nm (knots)	129	135	145	142	138	134	135	135	134	131
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	9	7.8	9.4	8.4	8.4	9.9	8.8	8.4

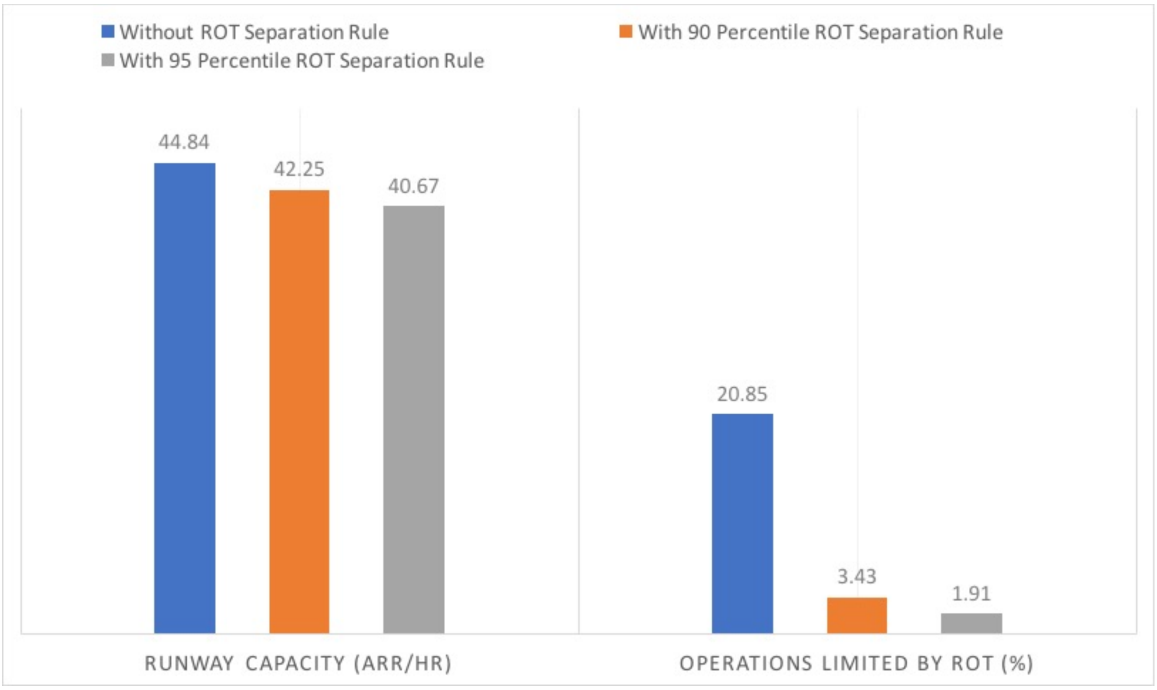


Figure 48 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 22R.

The runway throughput at Runway 22R will reduce up to 4 arrivals per hour with a 95 percentile ROT separation rule. The low capacity is caused by two main factors: fleet mix and poor runway configuration. According to Table 25, some heavy aircraft as B772 is operated on Runway 22R, which leads to the large wake vortex separation between in-trail arrivals. The poor runway configuration leads to the high ROTs as shown in Table 25. According to Figure 30, there is only one high-speed exit locate one-third from runway end, which leads to the large difference between the average ROT value and 95 percentile ROT value. In other words, if the landing aircraft miss the high-speed exit, it needs to taxi all the way to the runway end to exit. A 95 percentile ROT separation rule is very necessary to decrease ROT limiting percentage to less than 2% during peak periods under RECAT II at Runway 22L.

4.4.6 ORD 28R, 10L

Runway 28R/10L at ORD are mainly used for departures. Since it is the longest runway (13,000 feet) at ORD, during night times or some off-peak periods, some heavy cargo aircrafts such as B744 and B772 lands on this runway. During peak periods, Runway 29R/10L is mainly used for departures. However, the throughput and ROT limiting percentage are also assessed for applying RECAT II on this runway. The inputs and simulation results are shown from Table 22 to Table 23, and Figure 49 to Figure 50.

Table 22 Inputs from ASDE-X Data for ORD 28R.

Aircraft ID	A319	A320	B737	B738	B744	B753	B772	CRJ2	CRJ7	DC10	E145	E170
Fleet Mix(%)	2	18	30	18	5	5	2	4	7	6	3	2
Mean ROT(s)	67	67	72	67	86	72	71	61	63	88	60	62
ROT std.	10.69	9.80	7.10	10.35	9.61	11.65	11.15	7.34	9.26	9.27	8.71	8.87
90 Percentile ROT (s)	86	79	85	80	98	91	71	72	75	98	71	75
95 Percentile ROT (s)	89	84	90	86	99	94	87	74	80	99	75	80
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	153	142	138	134	135	135	134	131
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	10.2	7.8	9.4	8.4	8.4	9.9	8.8	8.4

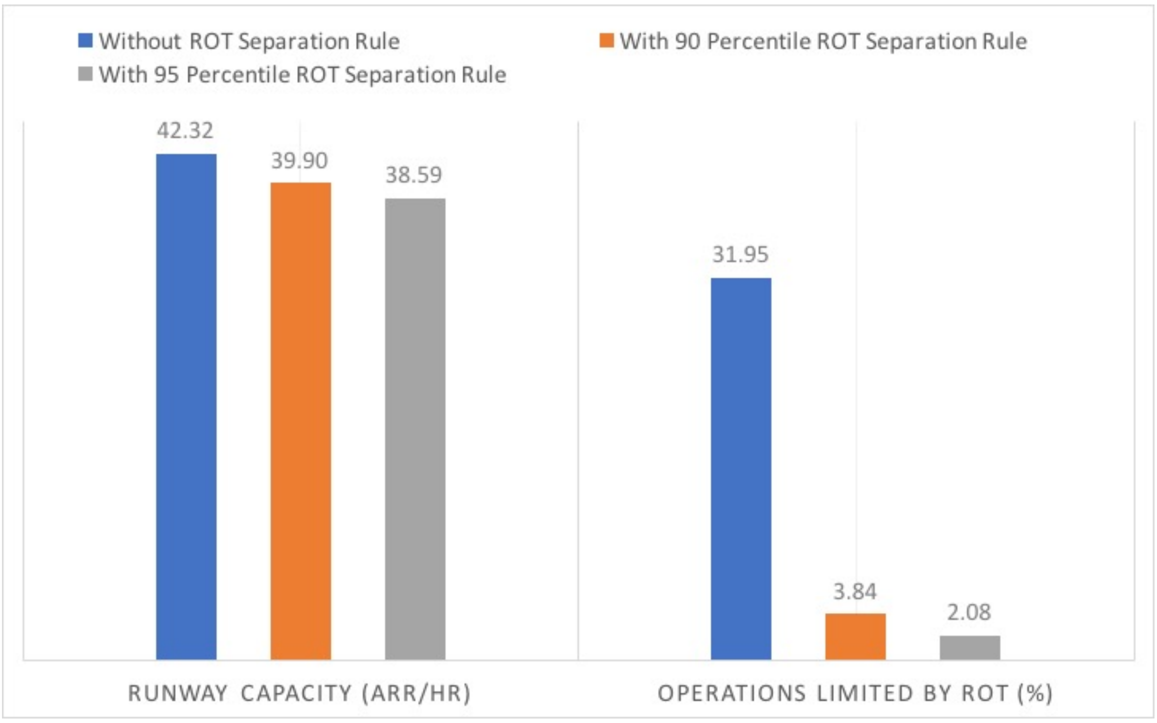


Figure 49 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 28R.

Table 23 Inputs from ASDE-X Data for ORD 10L.

Aircraft ID	A319	A320	B737	B738	B744	B753	B772	CRJ7	DC10	E145	E170
Fleet Mix(%)	3	26	3	34	8	6	3	4	2	5	6
Mean ROT(s)	78	75	81	76	82	76	76	75	66	72	72
ROT std.	8.7	9.5	9.5	9.3	9.0	10.2	10.0	5.7	8.3	10.4	9.9
90 Percentile ROT (s)	91	89	93	89	95	88	91	84	79	88	87
95 Percentile ROT (s)	93	94	94	93	97	93	92	89	81	90	92
Approach Ground Speed Last 2.5 nm (knots)	129	135	134	145	153	142	138	135	135	134	131
Approach Ground Speed Last 2.5 nm std.	7.5	7.2	8	9	10.2	7.8	9.4	8.4	9.9	8.8	8.4

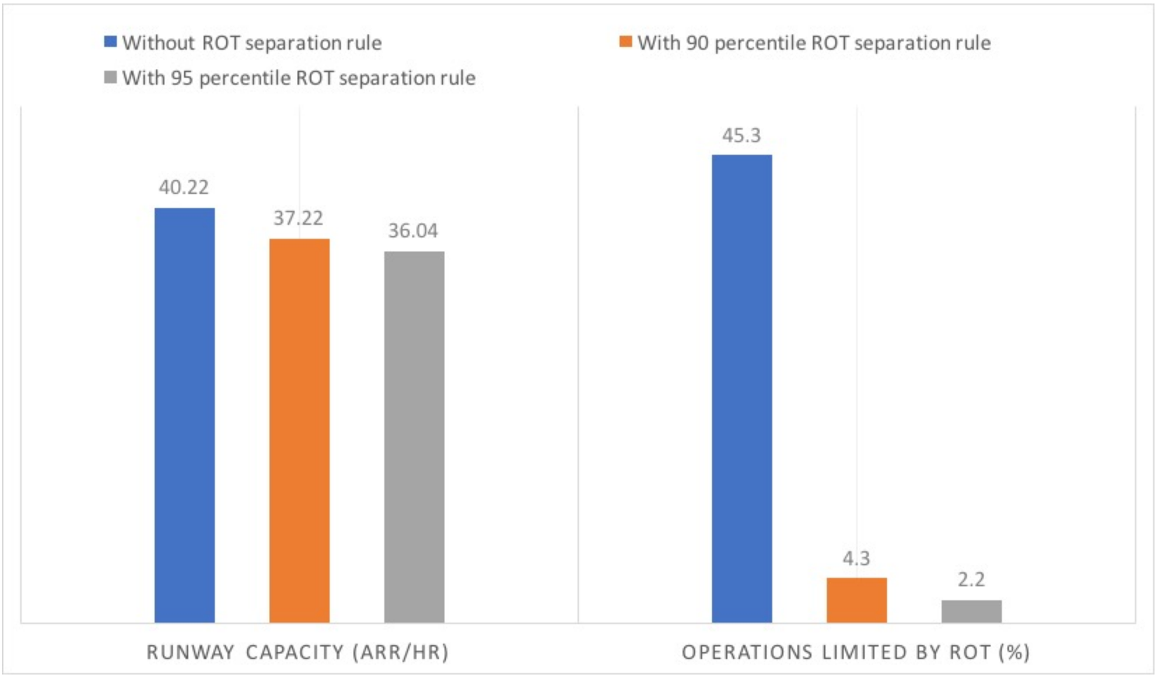


Figure 50 Runway Capacity & ROT Limiting Percentage under RECAT II for ORD 10L.

From the simulation result, the ROT will play an important role in throughput if Runway 28R/10L is operated at capacity for arrivals. From Table 26 and Table 27, the heavy aircrafts such as B772 and B744 take up to 10% of the fleet mix, which leads to the decrease of throughput; the long runway length and the multiple highway exits shown in Figure 30 leads to a relatively high average ROT values and a large standard deviation of ROTs. With a 95 percentile ROT constraint, through the ROT limiting percentage goes down to 2.2%, the arrival throughput decreases from 40.22 to 36.08 arrivals per hour.

This analysis indicates the importance of adding an ROT separation rule to each runway end to operate RECAT II successfully. The research shows that with a 95 percentile ROT separation rule, there will be near or less than 2% of the operations limited by ROT under RECAT II during peak periods, which is close to the current operations.

4.4.7 CLT 18R, 18L, 18C

For CLT south flow, the runway capacities and the operations limited by ROT are simulated for mainly used runway end (18R, 18L, and 18C) based on the inputs from ASDE-X data. The inputs for each runway end and the simulation results are shown from Table 24 to Table 26 and Figure 51 to Figure 53.

Table 24 Inputs from ASDE-X Data for CLT 18R.

Aircraft ID	A319	A320	A321	B712	B738	CRJ2	CRJ7	CRJ9	DH8C	E170	MD88
Fleet Mix(%)	17	6	19	1	2	17	7	22	3	5	1
Mean ROT(s)	55	55	55	54	55	51	52	50	53	53	53
ROT std.	5.7	6.4	6.3	6.67	6.8	4.9	5.0	4.7	5.2	5.4	7.1
90 Percentile ROT (s)	62	63	63	62	64	57	58	57	60	60	61
95 Percentile ROT (s)	65	66	66	65	67	60	61	59	63	62.8	66
Approach Ground Speed Last 2.5 nm (knots)	134	139	148	138	148	138	132	136	121	134	142
Approach Ground Speed Last 2.5 nm std.	7.9	7.2	7.8	7.6	8	8	7.7	7.6	8.8	8.3	7.9

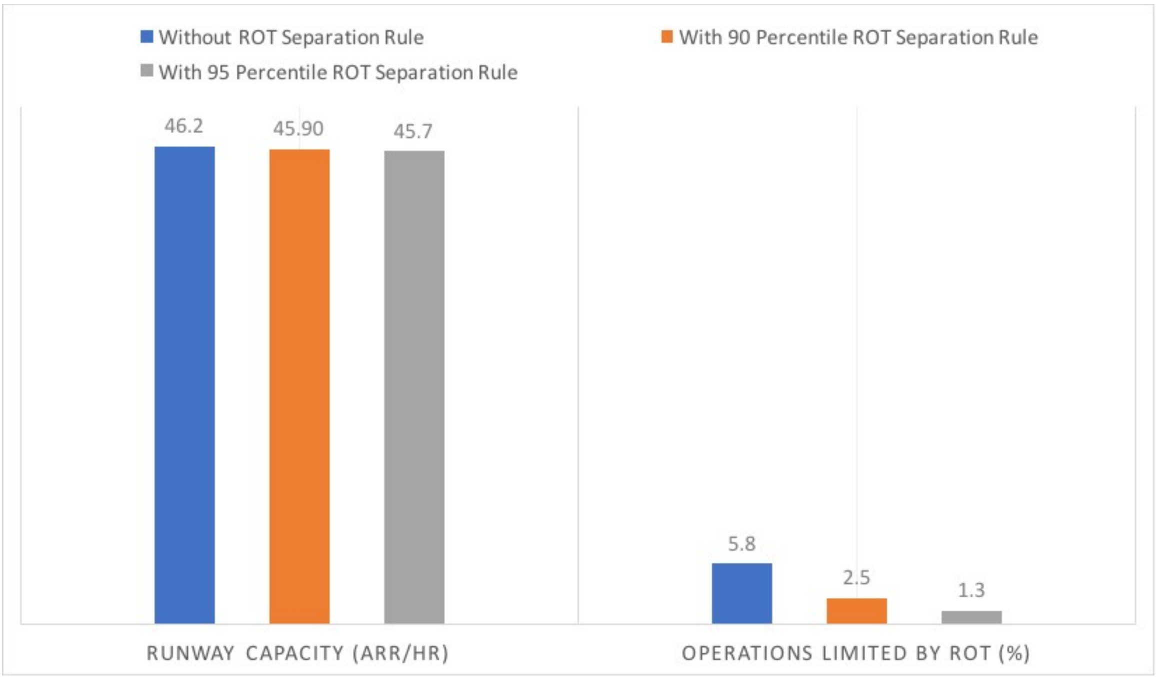


Figure 51 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18R.

Table 25 Inputs from ASDE-X Data for CLT 18L.

Aircraft ID	A319	A320	A321	B738	CRJ2	CRJ7	CRJ9	DH8C	E170
Fleet Mix(%)	16	9	14	4	11	7	24	11	4
Mean ROT(s)	55	55	57	56	50	52	51	49	51
ROT std.	6.8	6.6	6.9	7.3	6.7	6.9	6.7	6.5	6.1
90 Percentile ROT (s)	64	64	65	64	59	61	61	57	60
95 Percentile ROT (s)	66	66	68	68	62	64	64	61	62
Approach Ground Speed Last 2.5 nm (knots)	134	139	148	138	132	132	136	121	134
Approach Ground Speed Last 2.5 nm std.	7.9	7.2	7.8	8	8	7.7	7.6	8.8	8.3

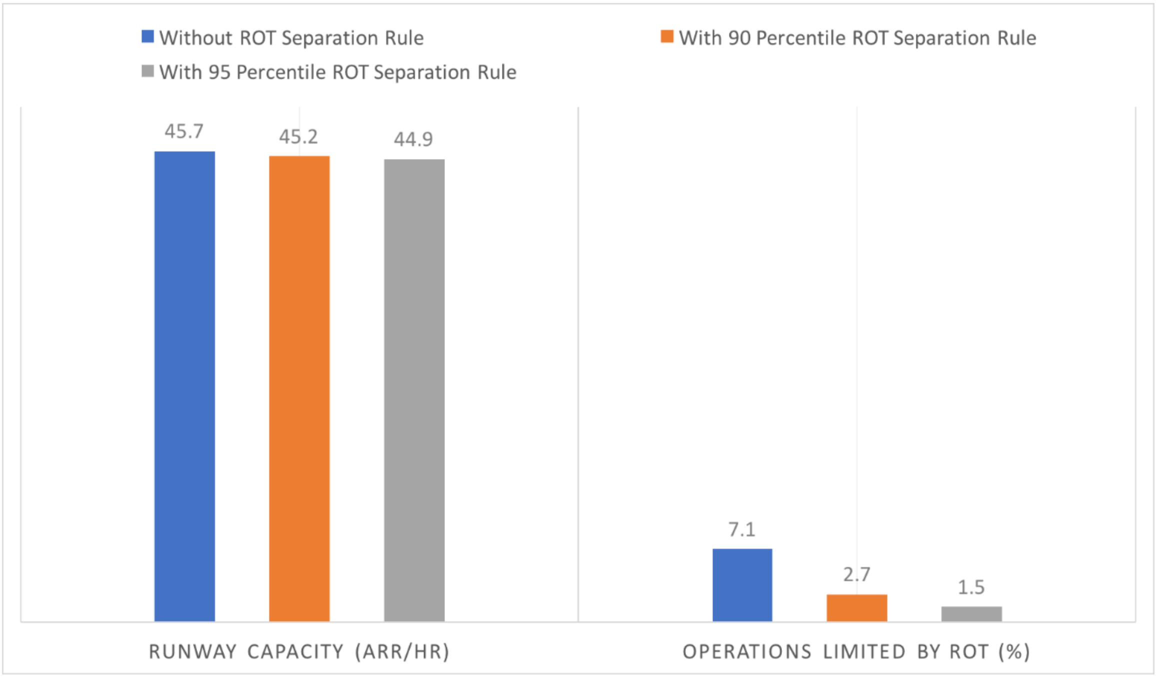


Figure 52 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18L.

Table 26 Inputs from ASDE-X Data for CLT 18C.

Aircraft ID	A306	A319	A320	A321	A332	A333	B737	B738	CRJ2	CRJ7	CRJ9	DH8C	E170
Fleet Mix(%)	2	15	5	21	4	5	2	2	14	6	16	2	7
Mean ROT(s)	72	58	60	62	68	67	57	60	52	54	54	52	53
ROT std.	10.6	9.3	9.7	7.5	7.7	7.6	9.4	9.8	8.4	9.0	8.9	8.4	8.5
90 Percentile ROT (s)	85	71	72	71	78	76	70	72	64	67	66	61	66
95 Percentile ROT (s)	90	75	76	74	82	80	73	76	68	71	70	64	70
Approach Ground Speed Last 2.5 nm (knots)	130	134	139	148	134	139	136	148	138	132	136	121	134
Approach Ground Speed Last 2.5 nm std.	8.0	7.9	7.2	7.8	7.4	7.8	8.6	8.0	8.0	7.7	7.6	8.8	8.3

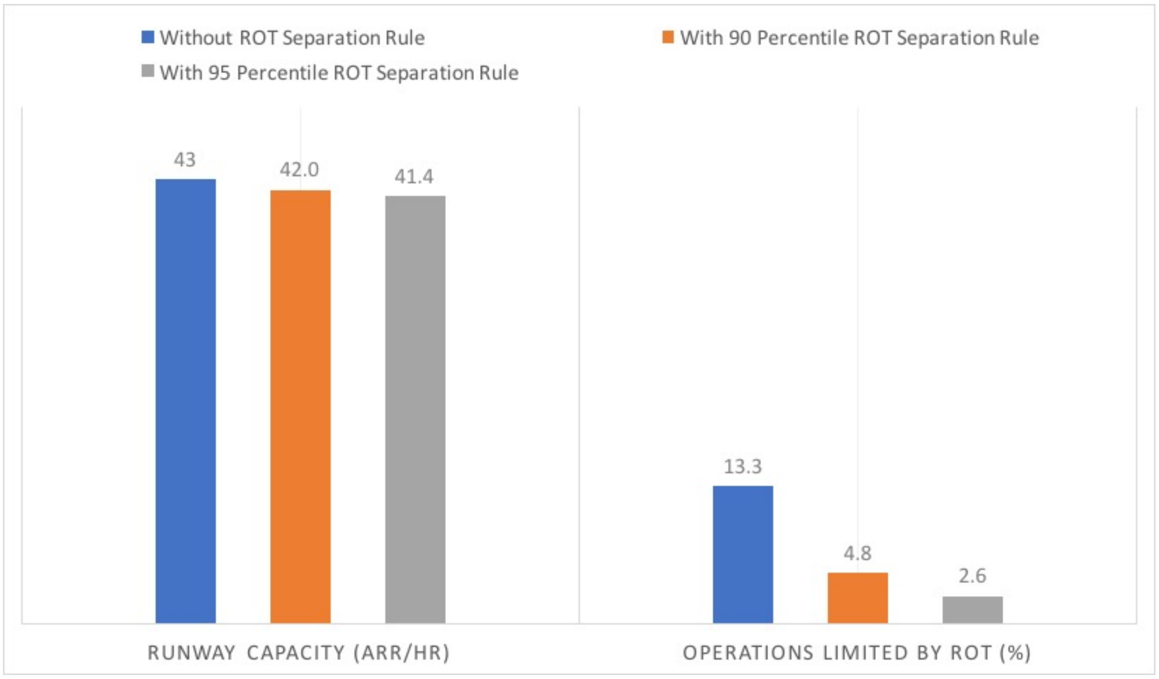


Figure 53 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 18C.

For CLT south flow, the runway throughput will reduce less than 2 arrivals per hour while can reduce the ROT limiting percent to near or less than 2.5% during peak period per each runway end. For CLT 18R and 18L, the throughput can raise up to 45 arrivals per hour with a 95 percentile ROT separation rule during RECAT II. The reason lies in that these two runways has relatively shorter runway length and multiple high-speed runway exits as shown in Figure 54. While for runway end 18C, fewer throughput could be obtained as the larger runway length and more relatively large aircraft operations.

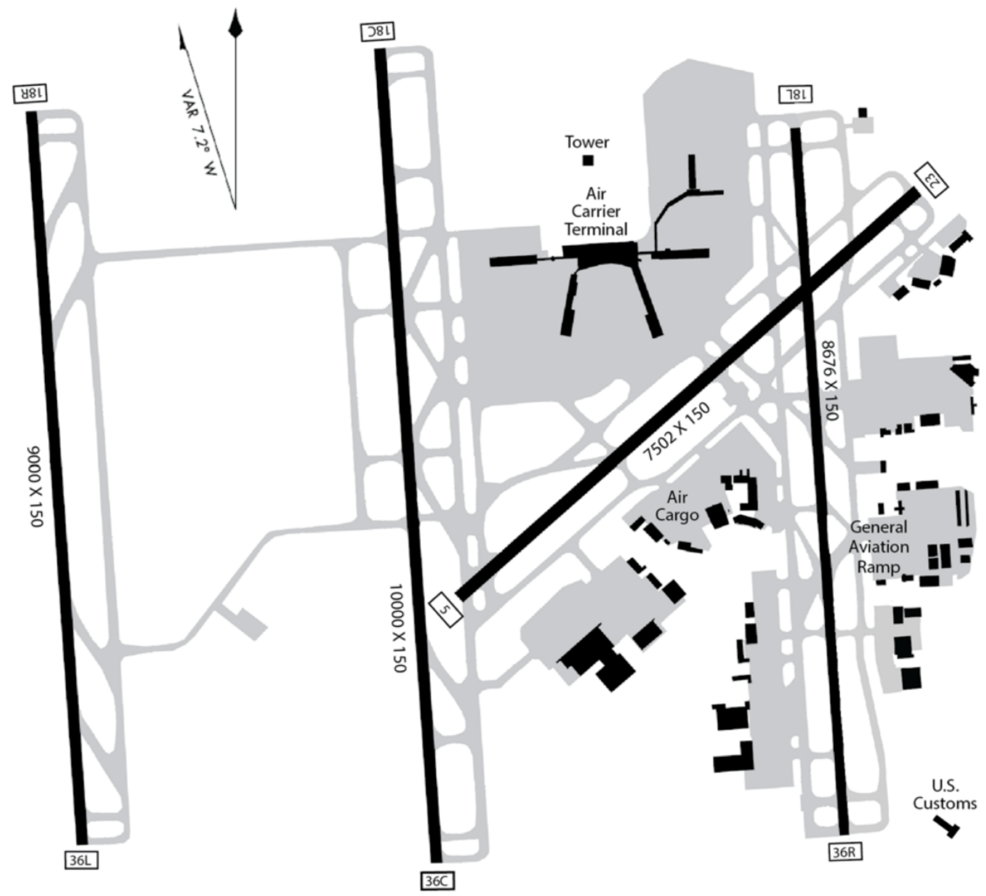


Figure 54 Runway Configurations at CLT (Source: Airnav.com.).

4.4.8 CLT 36L, 36R, 36C

For CLT north flow, the runway capacities and the operations limited by ROT are simulated for mainly used runway end (36L, 36R, and 36C) based on the inputs from ASDE-X data. The inputs for each runway end and the simulation results are shown from Table 27 to Table 29 and Figure 55 to Figure 57.

Table 27 Inputs from ASDE-X Data for CLT 36L.

Aircraft ID	A319	A320	A321	B712	B737	B738	CRJ2	CRJ7	CRJ9	DH8A	DH8C	E145	E170	MD88
Fleet Mix(%)	17	7	18	1	1	2	17	7	22	1	3	1	5	1
Mean ROT(s)	54	53	54	53	54	54	50	51	49	57	53	51	52	51
ROT std.	5.5	5.9	6.3	6.4	6.0	6.6	4.8	4.9	4.7	5.1	5.0	5.7	5.0	6.4
90 Percentile ROT (s)	61	61	62	62	62	62	56	57	56	63	59	58	58	59
95 Percentile ROT (s)	64	64	64	64	64	66	59	59	58	66	61	61	61	62
Approach Ground Speed Last 2.5 nm (knots)	134	139	148	138	136	148	138	132	136	117	121	137	134	142
Approach Ground Speed Last 2.5 nm std.	7.9	7.2	7.8	7.6	8.6	8	8	7.7	7.6	9.4	8.8	9	8.3	7.9

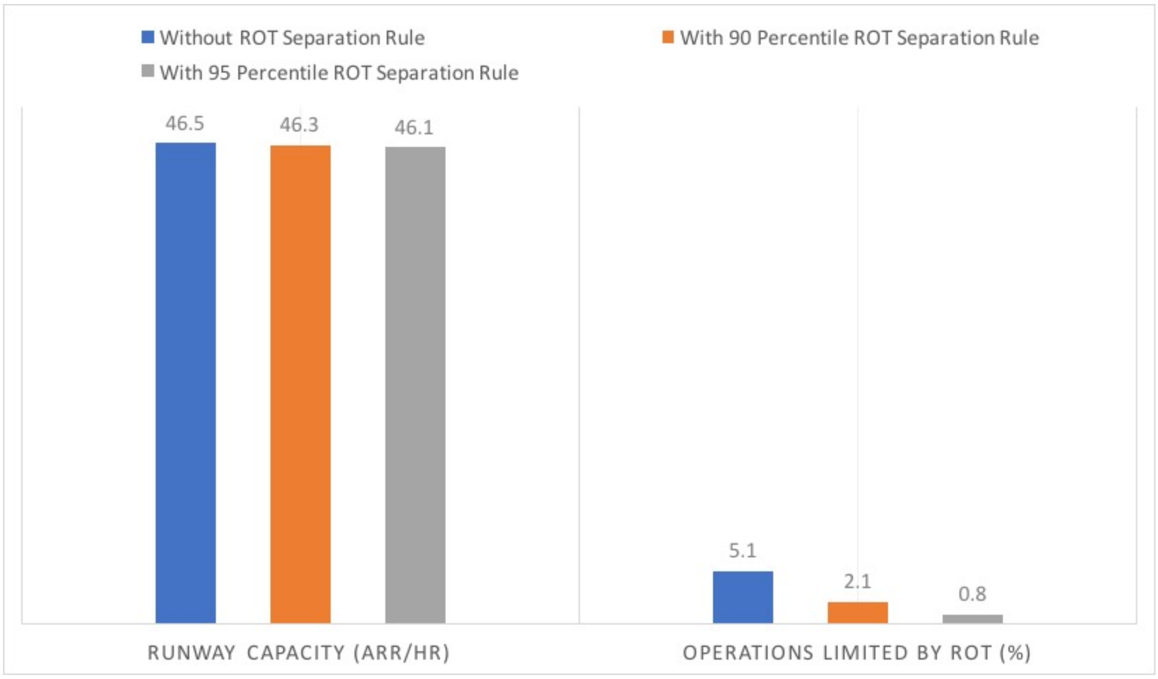


Figure 55 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36L.

Table 28 Inputs from ASDE-X Data for CLT 36R.

Aircraft ID	A319	A320	A321	A333	B738	B752	C56X	CRJ2	CRJ7	CRJ9	DH8A	DH8C	E170	E190
Fleet Mix(%)	16	8	12	1	3	1	2	11	7	21	3	9	4	1
Mean ROT(s)	61	62	63	76	60	66	56	64	62	63	59	57	51	55
ROT std.	10.3	9.7	7.8	10.1	9.9	10.8	8.5	11.8	12.3	11.3	14.4	13.2	8.8	10.4
90 Percentile ROT (s)	73	74	73	88	72	79	66	78	77	76	82	77	64	70
95 Percentile ROT (s)	77	77	76	93	75	84	69	84	82	82	87	83	68	73
Approach Ground Speed Last 2.5 nm (knots)	134	139	148	139	148	132	125	138	132	136	117	121	134	136
Approach Ground Speed Last 2.5 nm std.	7.9	7.2	7.8	7.8	8	9.9	9.7	8.0	7.7	7.6	9.4	8.8	8.3	7.5

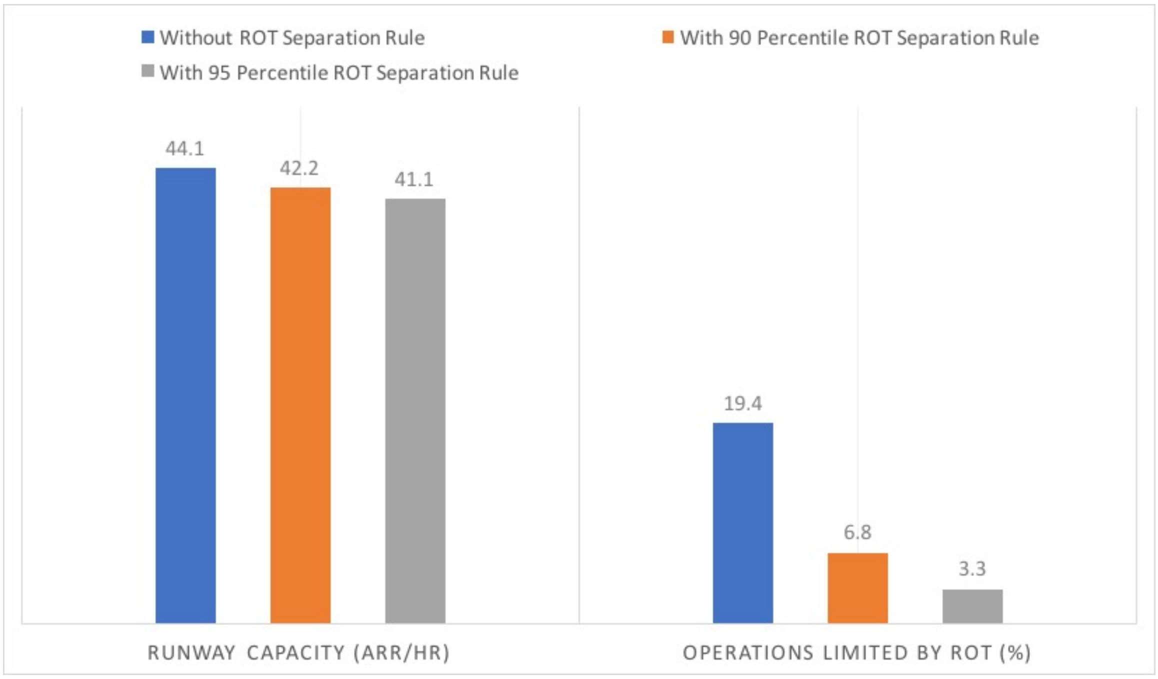


Figure 56 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36R.

Table 29 Inputs from ASDE-X Data for CLT 36C.

Aircraft ID	A306	A319	A320	A321	A332	A333	B737	B738	CRJ2	CRJ7	CRJ9	DH8C	E170	MD88
Fleet Mix(%)	2	14	6	25	2	2	2	2	13	6	15	2	7	2
Mean ROT(s)	67	61	60	62	69	66	64	63	54	56	55	52	56	60
ROT std.	13	10.5	9.6	7.9	9.9	7.9	9.7	9.5	9.1	10.2	9.5	6.7	9.5	7.9
90 Percentile ROT (s)	84	75	73	72	80	76	76	75	66	70	67	61	69	70
95 Percentile ROT (s)	89	79	77	75	86	80	81	80	71	75	72	65	73	73
Approach Ground Speed Last 2.5 nm (knots)	130	134	139	148	134	139	136	148	138	132	136	121	134	142
Approach Ground Speed Last 2.5 nm std.	8.0	7.9	7.2	7.8	7.4	7.8	8.6	8.0	8.0	7.7	7.6	8.8	8.3	7.9

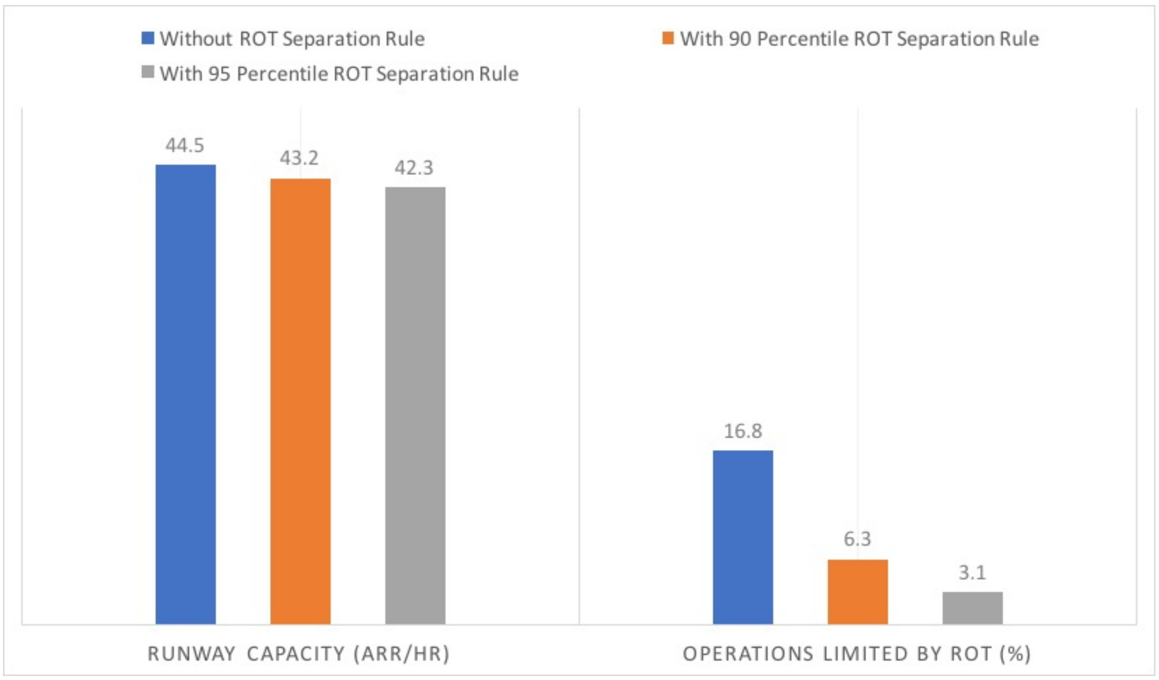


Figure 57 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 36C.

For CLT north flow, the ROT will have more influence to the runway throughput (up to reducing 3 arrivals per hour) per each runway during peak period comparing to south flow. For CLT 36R, 19.4% of the operations will be limited by ROT if no additional ROT separation rule is applied, which shows a big difference with its opposite operations (7.1% for CLT 18L). The reason lies in the terminal area locates near the end Runway 36R so that the pilots trend to taxi all the way to the runway end, which could be explained by the large ROTs in Table 28. For runway 36C, less throughput could be obtained because the long runway length and the large aircraft operations. Notice the Runway 18C/36C applies a mixed operation and is used mainly for departures.

4.4.9 CLT 23

Runway 23 at CLT is used only during certain wind conditions. The runway capacities and the operations limited by ROT are simulated based on the inputs from ASDE-X data. The inputs for each runway end and the simulation results are shown in Table 30 and Figure 58.

Table 30 Inputs from ASDE-X Data for CLT 23.

Aircraft ID	A319	A320	A321	B738	B752	C56X	CRJ2	CRJ7	CRJ9	DH8A	DH8C	E170	E190
Fleet Mix(%)	16	9	12	3	1	1	11	7	22	3	9	4	1
Mean ROT(s)	57	58	60	59	63	56	51	53	53	50	48	51	52
ROT std.	10.7	14.1	8.8	9.9	11.5	9.9	9.9	11	9.8	4.9	4.7	11.5	11.3
90 Percentile ROT (s)	70	70	69	69	75	67	64	67	65	56	54	67	66
95 Percentile ROT (s)	73	72	71	72	79	77	67	70	67	58	56	71	69
Approach Ground Speed Last 2.5 nm (knots)	134	139	148	148	132	125	138	132	136	117	121	134	136
Approach Ground Speed Last 2.5 nm std.	7.9	7.2	7.8	8.0	9.9	9.7	8.0	7.7	7.6	9.4	8.8	8.3	7.5

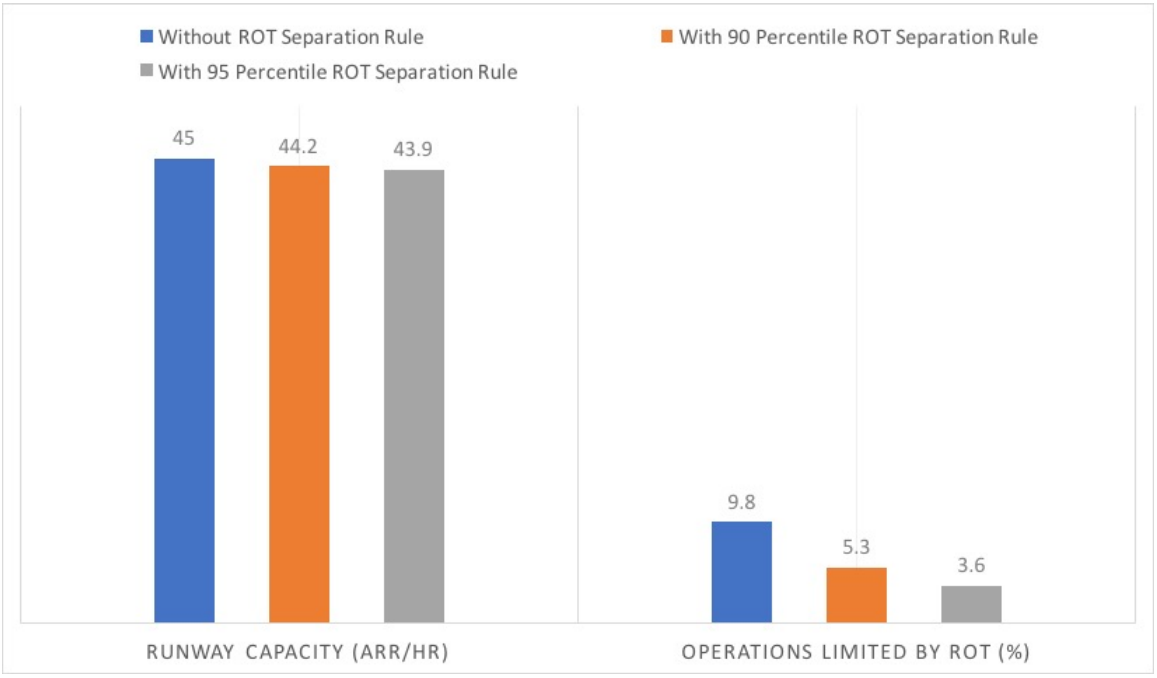


Figure 58 Runway Capacity & ROT Limiting Percentage under RECAT II for CLT 23.

For Runway 23 at CLT, the additional 95 percentile ROT separation rule can reduce the percent of operations limited by ROT to 3.6% from 9.8%. The relatively high percent is caused by the lack of high-speed exit as shown in Figure 54. This could also be explained by comparing ROT between Runway 23 and Runway 18R/36L. Though Runway 23 has relatively shorter runway length (7,500 feet comparing with 9,000 feet for 18R/36L), the mean and 95 percentile ROT value trends to be higher at Runway 23 because only the 90 degree exits is applied at Runway 23.

4.5 Runway Throughput with Additional 95 Percentile ROT Separation Rule under RECAT I and RECAT II

The study shows the arrival runway throughput under RECAT II could reduce less than 7% under the additional 95 percentile ROT separation rule. For example, Figure 59 shows for the main arrival runways at ORD (i.e. Runway 10C/28C, Runway 09L/10R, and Runway 09R/10L at ORD), the reduction of runway throughput with additional 95 percentile ROT separation rule is less than 3% comparing the runway throughput with wake vortex separation rule only. For CLT, Figure 60 shows the runway throughput for CLT36R will reduce near 7% under the additional 95 percentile ROT separation rule because of the terminal location effects. For other runway ends, there will be less than 5% of capacity loss with the additional 95 percentile ROT separation rule.

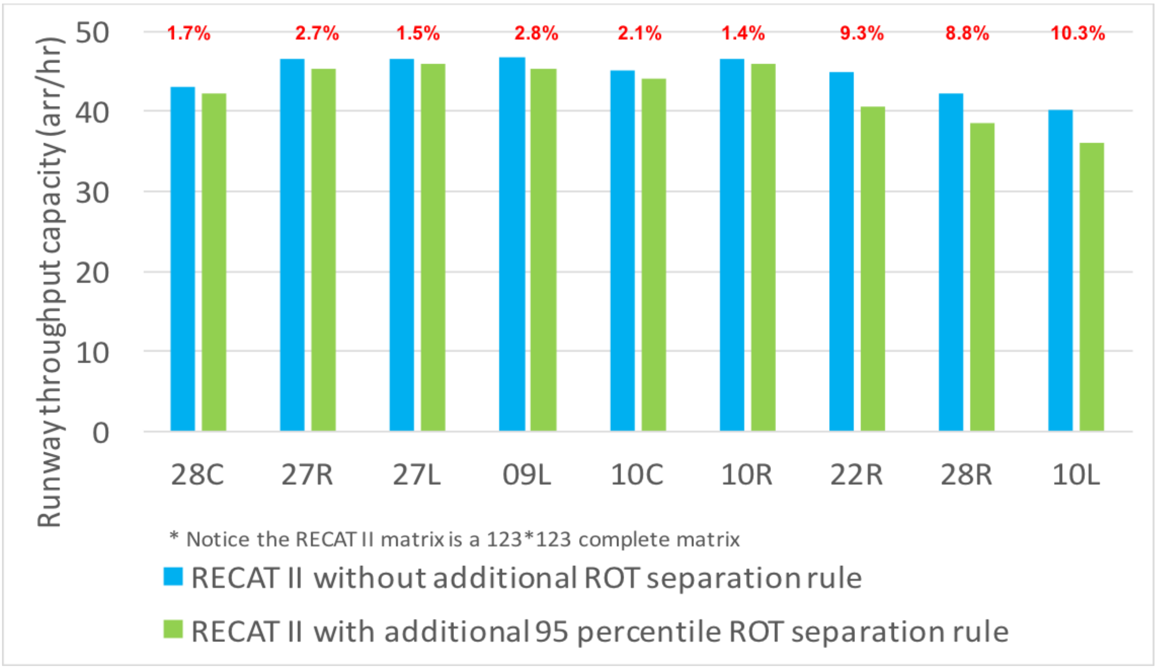


Figure 59 Runway Throughputs With and Without 95 Percentile ROT Separation Rule at ORD.

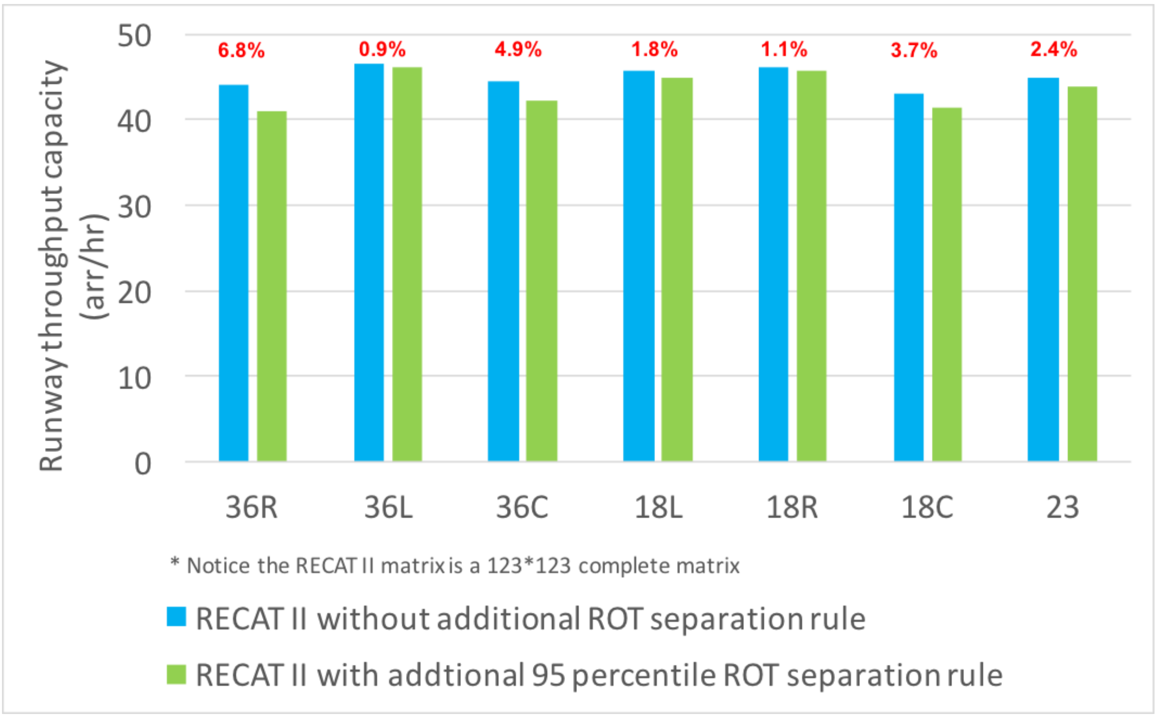


Figure 60 Runway Throughputs With and Without 95 Percentile ROT Separation Rule at CLT.

The study shows the runway throughputs could obtain from 10% to 16% more by applying RECAT II under the additional 95 percentile ROT separation rule comparing with RECAT I. For example, Figure 61 shows all of the main arrival runways at ORD could obtain 13% to 15% more throughputs under RECAT II with the additional 95 percentile ROT separation rule comparing to RECAT I. While Figure 62 shows all of the runways at CLT can obtain more than 10% throughput benefits at CLT.

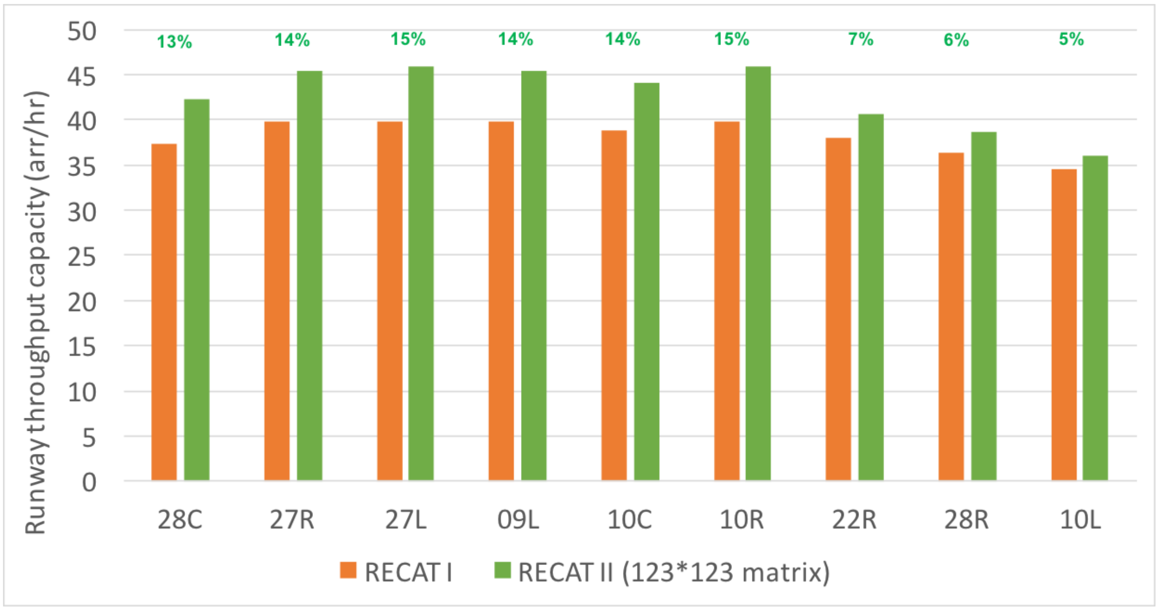


Figure 61 Runway Throughput Comparison between RECAT II and RECAT I with Additional 95 Percentile ROT Separation Rule at ORD.

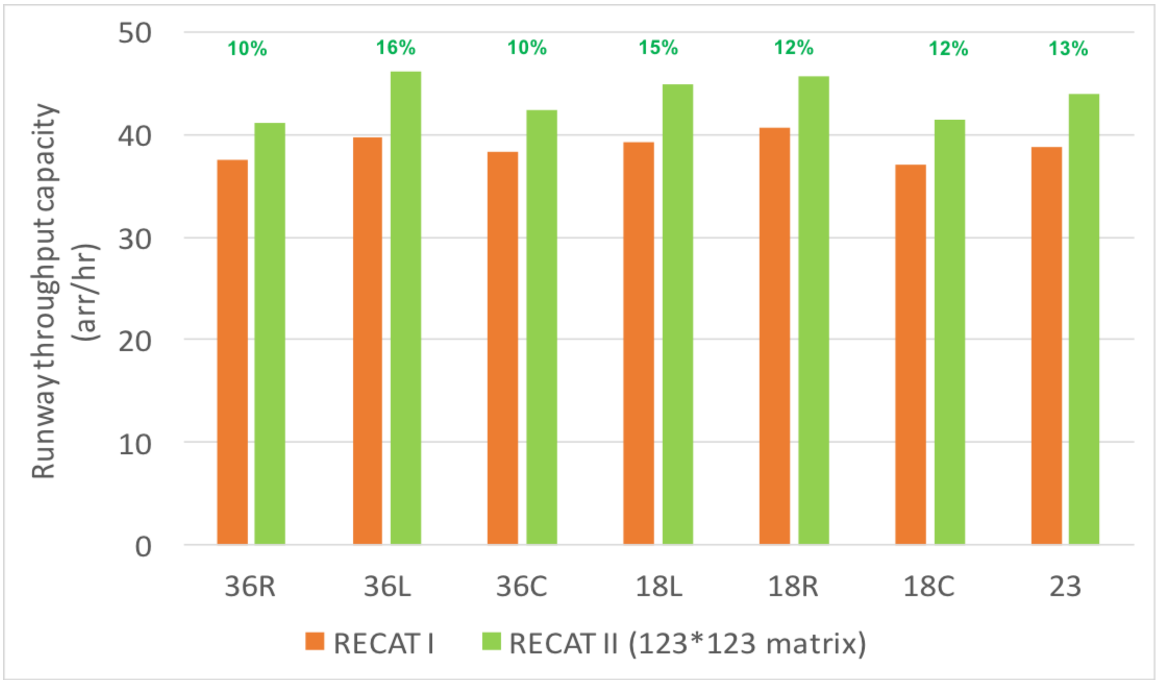


Figure 62 Runway Throughput Comparison between RECAT II and RECAT I with Additional 95 Percentile ROT Separation Rule at CLT.

5. Potential Go-around Estimation under RECAT II

5.1 Methodology to Estimate Potential Go-Around under RECAT II

This section focus on measuring the risk of potential go-arounds if lower separation rules (RECAT II) are applied. The potential go around is estimated by comparing the ROTs of the leading aircraft and the estimated in-trail time intervals under RECAT II. The hypothesis lies the in-trail time separation will reduce equivalent time for the reduced distance separation under RECAT II as shown in Figure 63. Equation 8 and 9 shows the calculation procedure of the estimated in-trail separation under RECAT II.

$$\Delta\, wake\ separation = RECAT\ II\ Separation - RECAT\ I\ Separation \qquad (8)$$

$$Time\ interval_{RECAT\ II} = Time\ interval_{current} - \frac{\Delta\, wake\ separation}{Approach\ speed\ for\ following} \qquad (9)$$

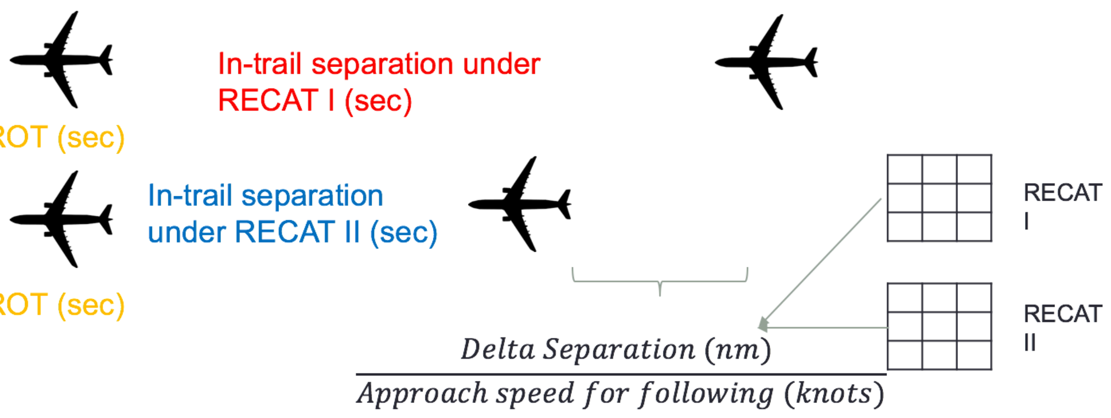

In-trail separation under RECAT I (sec)
ROT (sec)
In-trail separation under RECAT II (sec)
ROT (sec)
RECAT I
RECAT II
$\frac{Delta\ Separation\ (nm)}{Approach\ speed\ for\ following\ (knots)}$

Figure 63 Methodology for Estimating the Time Interval between In-trail Arrivals under RECAT II.

To illustrate the detailed procedure of in-trail time separation estimation under RECAT II, Table 31 shows two samples of in-trail group analysis. For the first in-trail group, the leading is a CRJ7 with a ROT of 75.7 seconds and the following aircraft is an E145 with an approach speed of 146 knots. The minimum separation for this in-trail group under RECAT I is 2.5 nautical miles and 2 nautical miles for RECAT II. Based on equation 9, the in-trail time under RECAT II is estimated to be 63.68 seconds, which is less than the ROT for the leading aircraft. For this case, a potential go around could happen because the leading aircraft cannot clear the runway when the following aircraft cross the threshold under RECAT II. For the second in-trail group, the leading aircraft is an E145 with an ROT of 56.77 seconds and the following aircraft is a CRJ 2 with an approach speed of 128 knots. The in-trail time under RECAT II is estimated to be 103.9 seconds, which is still larger than the ROT of leading aircraft. For this case, the following aircraft can lands successfully under RECAT II without any ROT constraint.

Table 31 Samples of In-trail Arrival Information and Estimated in-trail time under RECAT II.

Leading	Following	Leading aircraft ROT (s)	Following approach speed (knots)	Time interval under RECAT I (s)	Separation under RECAT I (nm)	Separation under RECAT II (nm)	Delta Separation (nm)	In-trail Time under RECAT II (s)
'CRJ7'	'E145'	75.70	146.09	76	2.5	2	0.5	63.68
'E145'	'CRJ2'	56.77	127.64	118	2.5	2	0.5	103.90

The study also estimates the potential go-arounds with an additional 95 percentile ROT separation rule. With the additional ROT separation rule, the in-trail separation under RECAT II will be calculated by Equation 10.

$$Time\ interval_{RECAT\ II} = MAX\left\{Time\ interval_{current} - \frac{\Delta\, wake\ separation}{Approach\ speed\ for\ following},\ ROT\ Constraint_{leading}\right\} \quad (10)$$

5.2 Potential Go-arounds Estimation with and without Additional ROT Separation Rule under RECAT II for Several Hub Airports

This study shows that with the additional ROT separation rule, the potential go-arounds could be reduced to less than 1% for analyzed hub airports (i.e. ORD, CLT, DEN, and ATL). To better illustrate the effects of ROT to potential go-arounds, Figure 64 and Figure 65 shows the distribution of time intervals between in-trail arrivals under RECAT I and RECAT II with wake vortex separation consideration only, and the ROTs of leading aircraft at ORD. Figure 65 shows it is obvious that there will be more potential go-arounds caused by ROT under RECAT II with wake vortex separation only because the estimated delta time between in-trail time intervals and ROTs of leading aircraft is negative for some cases. Figure 66 and Figure 67 shows the time gaps between in-trail arrivals and leading aircraft ROT under RECAT II with and without an additional 95 percentile ROT separation rule at ORD 27L, which indicates the additional 95 percentile ROT separation rule could highly reduce the number of times of potential go-arounds under RECAT II. Table 32 to Table 35 shows the estimated potential go-arounds with and without the additional ROT separation rule for ORD, CLT, DEN, and ATL. From the comparison, CLT and DEN trends to have lower potential go-arounds than ORD and ATL even without the additional ROT separation rule. This lies in the appropriate design of the runway length and high-speed exit locations at CLT and DEN. The potential go-arounds at ORD and ATL could raise up to 7% and 6% respectively without the additional ROT separation rule, which is not acceptable operationally.

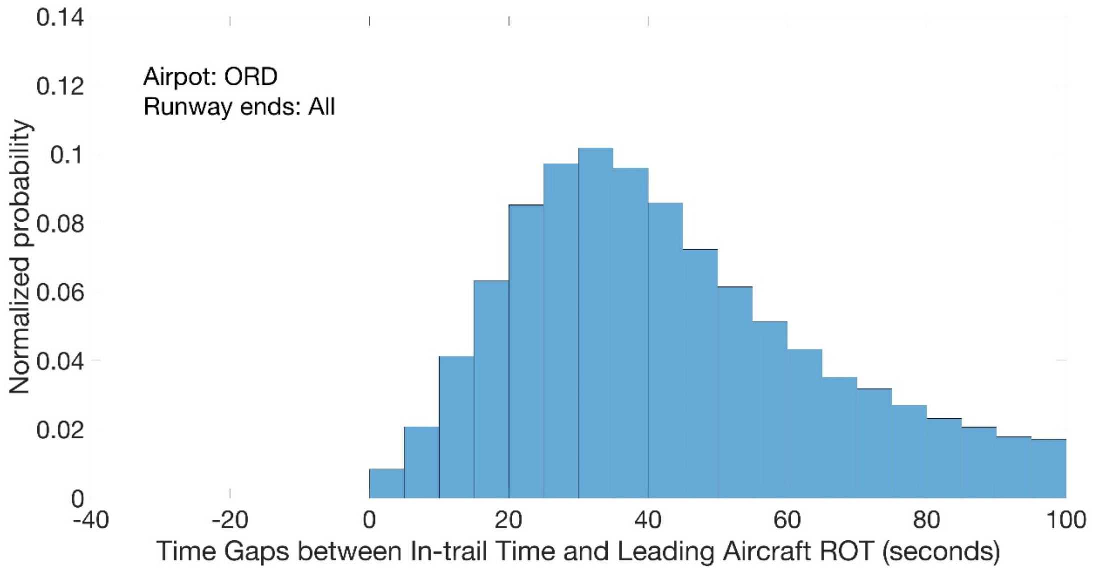


Figure 64 Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT I with only Wake Vortex Separation Rule at ORD.

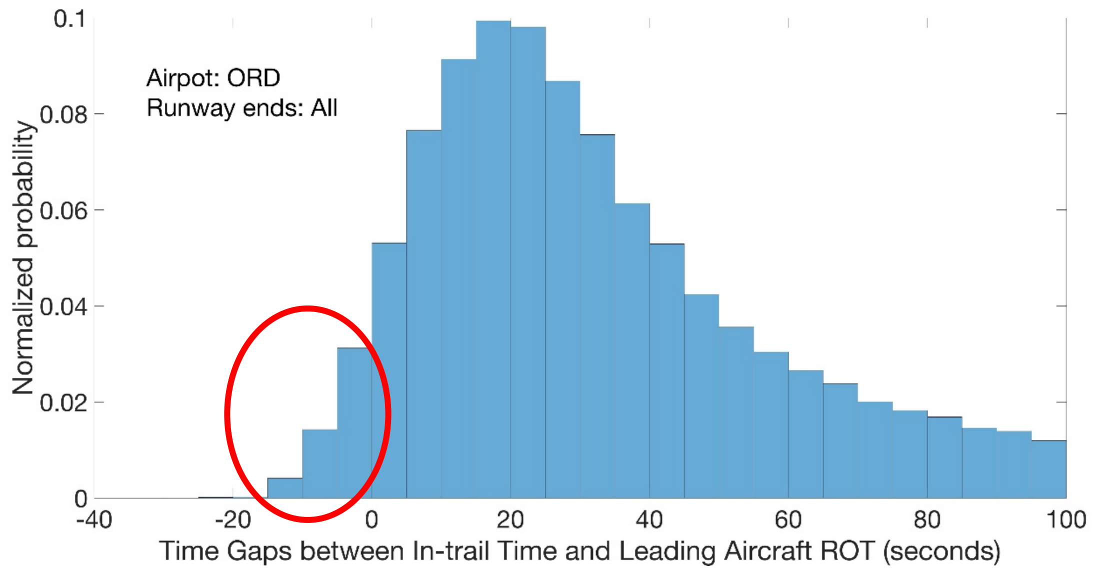


Figure 65 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with Only Wake Vortex Separation Rule at ORD.

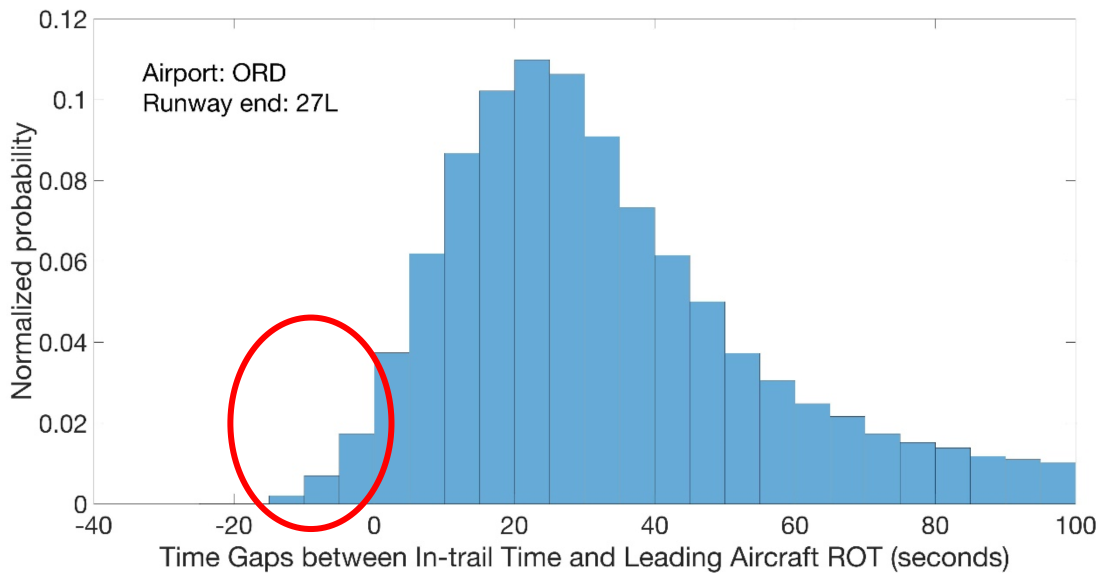


Figure 66 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with Only Wake Vortex Separation Rule at ORD 27L.

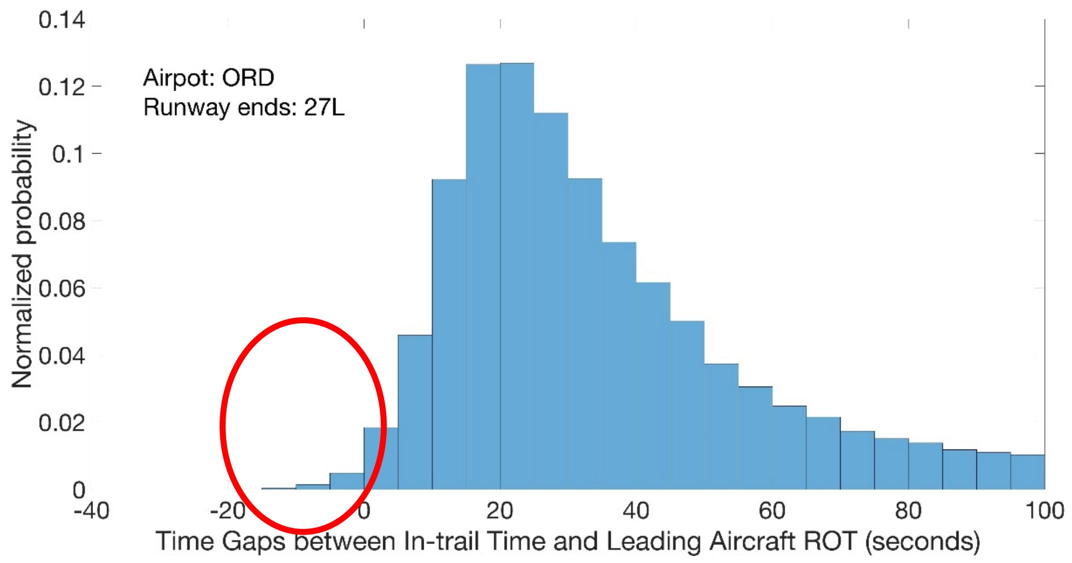


Figure 67 Estimated Time Gaps Histogram Plot between In-trail Intervals and ROTs of Leading Aircraft under RECAT II with an Additional 95 Percentile ROT Separation Rule at ORD 27L.

Table 32 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at ORD.

Runway End	Without the additional ROT separation rule (%)	With the recommended additional ROT separation rule (%)
09L	7. 63	0. 96
10C	5. 26	0. 69
10L	6. 75	0. 09
10R	2. 04	0. 35
27L	2. 19	0. 54
27R	4. 83	0. 79
28C	2. 65	0. 49
28R	3. 41	0. 22

Table 33 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at CLT.

Runway End	Without the additional ROT separation rule (%)	With the recommended additional ROT separation rule (%)
05	0.24	0
18C	0.94	0.15
18L	0.31	0.04
18R	3.57	0.59
23	2.51	0.42
36C	1.01	0.21
36L	3.51	0.71
36R	2.13	0.29

Table 34 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at DEN.

Runway End	Without the additional ROT separation rule (%)	With the recommended additional ROT separation rule (%)
7	3.81	0.62
8	0	0
16L	3.35	0.08
16R	8.02	0.29
17L	5.03	0.00
17R	3.11	0.29
25	1.31	0
26	3.27	0.21
34L	9.68	0
34R	1.72	0.14
35L	3.76	0.38
35R	3.09	0.28

Table 35 Potential Go-arounds Estimation under RECAT II with & without Additional ROT Separation Rule at ATL.

Runway End	Without the additional ROT separation rule (%)	With the recommended additional ROT separation rule (%)
08L	3.41	0.44
08R	0.74	0
09L	1.38	0
09R	2.45	0.35
10	5.95	0.65
26L	1.50	0
26R	6.07	0.71
27L	2.92	0.42
27R	1.48	0
28	3.90	0.71

This section provides a preliminary way to estimate the potential go-arounds with lower separation rules and proves that the ROTs will not cause go-arounds for more than 99% of the times by adding a 95 percentile ROT constraint. Also, it is important to notice the potential go around percentile estimated in this research is only the upper boundary for these reasons: 1) the ROT standard used for this research is ROT from runway threshold to hold bar, which is relatively conservative. In reality, the following aircraft does not need to make a go around if the leading aircraft has clear the runway but not reach the hold bar. 2) There is a correlated human behavior between the in-trail separations which cannot be captured by this study. In other words, the pilots will take reactions (e.g., slow down) when the leading aircraft has not clear the runway. 3) ATC may command the following aircraft to make a detour in the air instead of a go around when the leading aircraft cannot clear the runway in time. In conclusion, adding a 95 percentile ROT constraint can heavily reduce the potential go-arounds caused by ROT under lower separation rules.

6. The Incremental Benefits of RECAT II and TBS under Wind Conditions at ORD

6.1 Wind Data at ORD

Wind has a strong influence on runway throughput for some specific airports. The Time-based Separation (TBS) is developed by EUROCONTROL to diminish the wind effects to airport throughput. For example, London Heathrow Airport (LHR) has applied TBS since March 2015. According to EUROCONTROL report, LHR has gain additional 1.2 arrivals per hour across all wind conditions and additional 2.9 arrivals per hour in strong wind conditions (EUROCONTROL). As Chicago O'Hare Airport is also suffering a capacity loss under strong wind conditions, the benefits from TBS and RECAT II comparing with RECAT I (current operation) under different wind conditions is studied. Based on the wind data from the year 2000 to the year 2014 at ORD from National Oceanic and Atmospheric Administration (NOAA), the different headwind scenarios have created for different runway ends at ORD. For this research, since only the headwind effects are taken into consideration, the headwind is calculated by equation 10. For example, Figure 68 shows that for Runway end 27R, only the wind against runway direction (from 180 degrees to 360 degrees for this case) is taken into consideration.

$$Headwind = |\ cos(Wind\ direction - Runway\ heading) * Wind\ speed\ | \qquad (11)$$

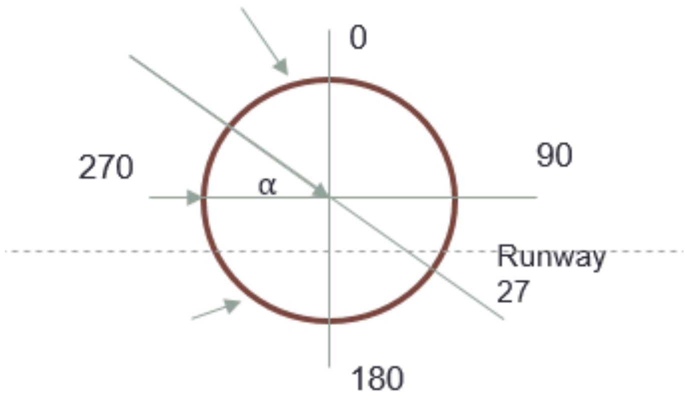


Figure 68 Sample Headwinds for Runway End 27R at ORD.

Based on the runway headings at ORD, three different groups wind inputs are generated based on the wind data for west flow (27L, 27R, 28C and 28R), east flow (09L, 09R, 10L and 10R) and Runway 22R as shown from Table 36 to Table 38.

Table 36 Wind Conditions for West Flow (Runway 27L, 27R, 28C, and 28R) at ORD.

Wind Coverage	Headwind (knots)	Headwind above average (knots)
Normal condition	6.33	0
60%	6.85	0.52
70%	7.85	1.52
80%	9.6	3.27
90%	12.12	5.79
99%	19.05	12.72

Table 37 Wind Conditions for East Flow (Runway 09L, 09R, 10L and 10R) at ORD.

Wind Coverage	Headwind (knots)	Headwind above average (knots)
Normal condition	5.55	0
60%	6.06	0.51
70%	7.12	1.57
80%	8.49	2.94
90%	10.41	4.86
99%	15.7	10.15

Table 38 Wind Conditions for Runway 22R at ORD.

Wind Coverage	Headwind (knots)	Headwind above average (knots)
Normal condition	6.39	0
60%	6.89	0.5
70%	7.85	1.46
80%	9.6	3.21
90%	12.05	5.66
99%	18.08	11.69

6.2 Potential Gains from TBS and RECAT II under Wind Conditions

RUNSIM is used to simulate the potential capacity for each runway at ORD. During wind conditions, the ground approach speed is reduced due to the influence of headwind. In RUNSIM, the approach speed under wind conditions is calculated as equation 12.

$$Groud\ Approach\ Speed = Approach\ Speed - Headwind \qquad (12)$$

Based on the wind data inputs, the potential capacity gains from TBS and RECAT II under wind conditions for different runway ends for west flow (Runway 28C, 27R and

27L), east flow (09L, 28R and 10L) and Runway 22R at ORD are simulated in RUNSIM as shown from Figure 50 to Figure 53. From the simulation result, under extremely windy conditions, the runway throughput is decreased up to 9 arrivals per hour for west flow and 5 arrivals per hour for east flow under RECAT II. Since TBS has taken the ground approach speed into consideration, the throughput keeps the same even under extremely windy conditions. For Runway 22R, under extremely windy conditions, TBS could even bring more potential gains to throughput comparing with RECAT II. The research shows that the headwind will have a strong influence to distance-based separation at ORD. In other words, the distance-based separation RECAT II can still be improved to time-based separation to benefits more under wind conditions at ORD.

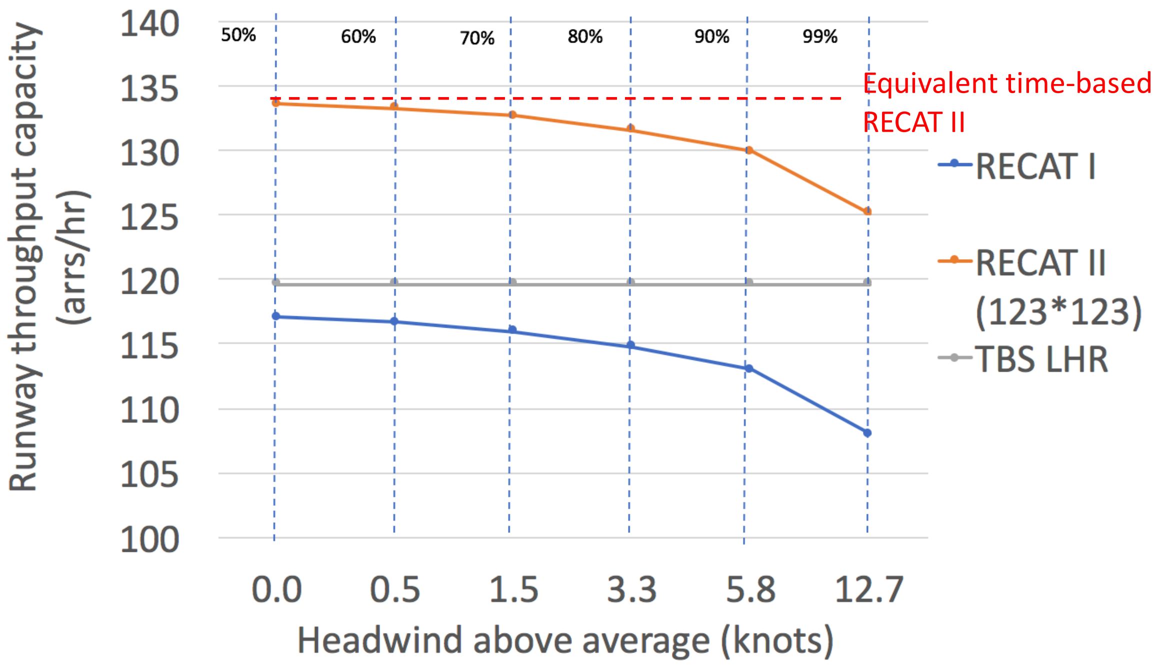


Figure 69 Potential Benefits from RECAT II and TBS under Wind Conditions for West Flow (Runway 28C, 27R, and 27L) at ORD.

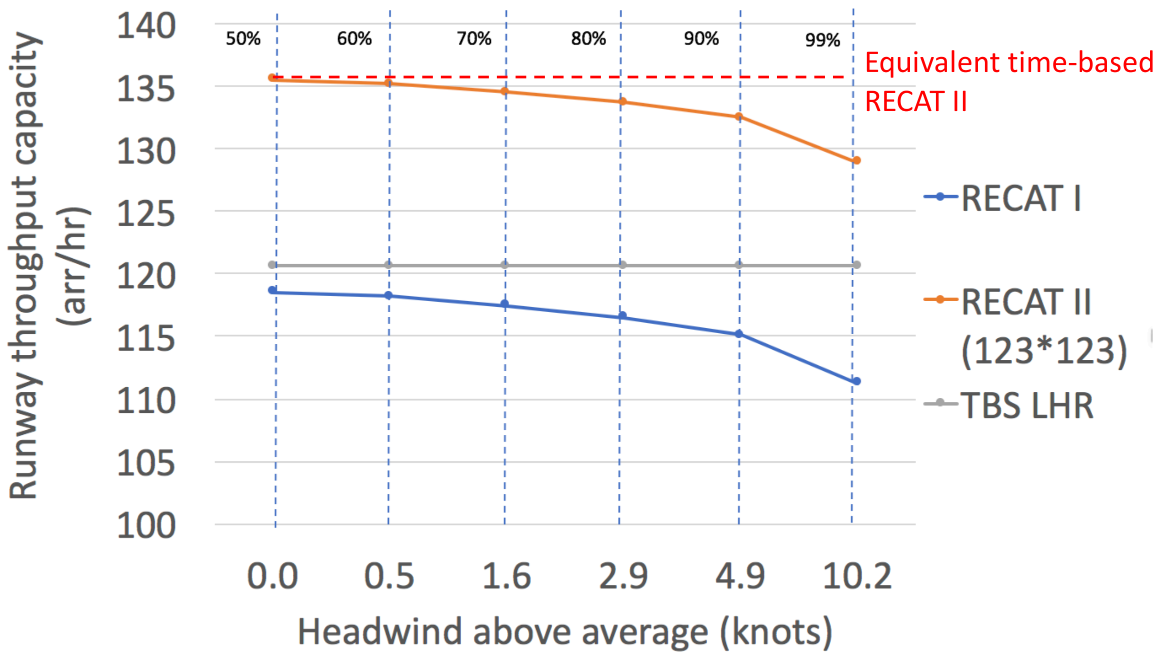


Figure 70 Potential Benefits from RECAT II and TBS under Wind Conditions for East Flow (Runway 09L, 28R, and 10L) at ORD.

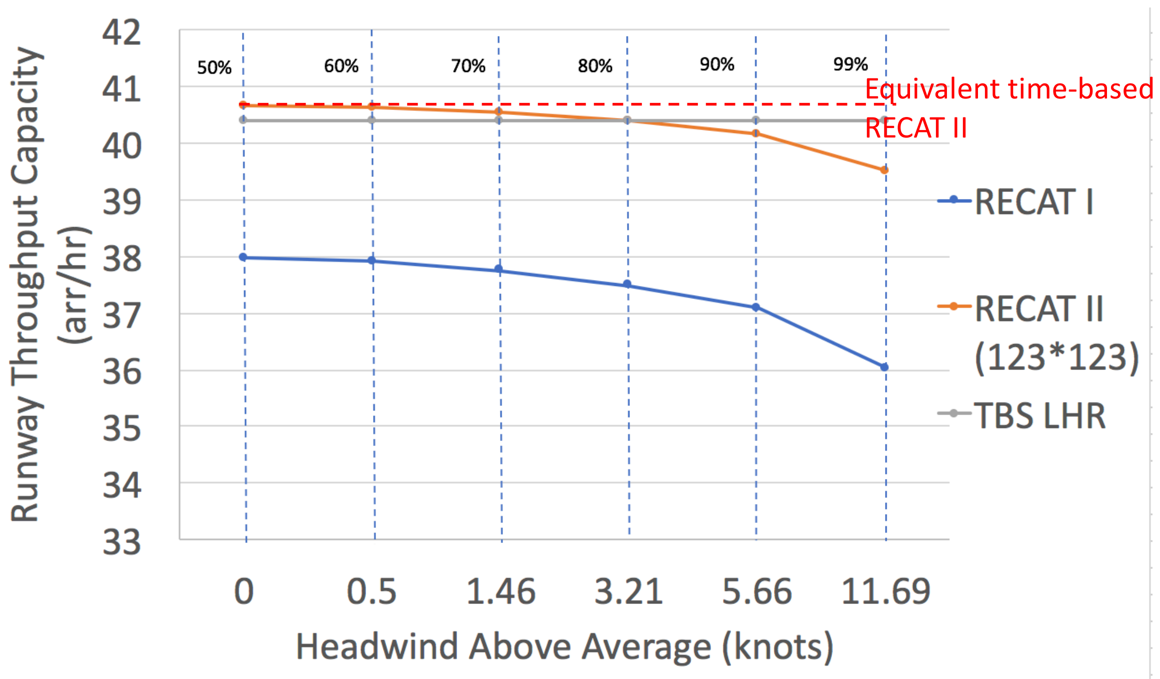


Figure 71 Potential Benefits from RECAT II and TBS under Wind Conditions for Runway end 22R at ORD.

6.3 Sample time-based RECAT II separation at ORD

This section provides a sample time-based RECAT II separation based on some typical aircraft types at ORD. The sample time-based RECAT II separation matrix is also compared with the TBS matrix that is applied at London Heathrow Airport (LHR). One or two most frequently operated aircraft are chosen to standard for each aircraft wake group as shown in Table 34. Based on the distance-based RECAT II separation shown in Table 35 and the approach speed for the last 2.5 nautical miles under normal wind conditions as shown in Table 34, the time-based RECAT II separation can be estimated as shown in Table 36. Comparing with the TBS matrix in Table 37, for most of the cases, the time-based RECAT II separation are much less than the TBS. As the minimum separation for distance-based RECAT II is 2 nautical miles, it is as expected that for these 2 nautical mile separation in-trail groups in RECAT 2, the equivalent time separation is 15 to 20 seconds less than TBS. However, when the leading aircraft is a heavy aircraft such as B744, for some in-trail aircraft types the TBS is less than time-based RECAT II separation, which indicates that TBS may obtain more benefits with more relative large aircraft.

Table 39 Typical Aircraft for Each Wake Group Operates at ORD.

Aircraft type	B744	B722	DC10	MD82	A320	B737	CRJ7	E170
Wake group	B	B	C	D	D	D	E	E
Approach speed for last 2.5 nm (knots)	153	138	135	134	135	134	135	131

Table 40 Distance-based RECAT II Separation for Typical Aircraft Operates at ORD.

RECAT 2 distance based separation (nm)			Following							
			B744	B722	DC10	MD82	A320	B737	CRJ7	E170
			B	B	C	D	D	D	E	E
Leading	B744	B	2	4.7	3.5	4.7	4.6	4.6	4.7	4.7
	B722	B	2	2	2	2	2	2	2	2
	DC10	C	2	3.1	2	3.2	3.1	3.1	3.2	3.2
	MD82	D	2	2	2	2	2	2	2	2
	A320	D	2	2	2	2	2	2	2	2
	B737	D	2	2	2	2	2	2	2	2
	CRJ7	E	2	2	2	2	2	2	2	2
	E170	E	2	2	2	2	2	2	2	2

Table 41 Sample Time-based RECAT II Separation Matrix for Typical Aircraft Types at ORD.

RECAT 2 time based separation (seconds)			Following							
			B744	B722	DC10	MD82	A320	B737	CRJ7	E170
			B	B	C	D	D	D	E	E
Leading	B744	B	47	123	93	126	123	124	125	129
	B722	B	47	52	53	54	53	54	53	55
	DC10	C	47	81	53	86	83	83	85	88
	MD82	D	47	52	53	54	53	54	53	55
	A320	D	47	52	53	54	53	54	53	55
	B737	D	47	52	53	54	53	54	53	55
	CRJ7	E	47	52	53	54	53	54	53	55
	E170	E	47	52	53	54	53	54	53	55

Table 42 Sample TBS Matrix for Typical Aircraft Types at ORD.

TBS separation (seconds)			Following							
			B744	B722	DC10	MD82	A320	B737	CRJ7	E170
			B	B	C	D	D	D	E	E
Leading	B744	B	90	113	113	113	113	113	135	135
	B722	B	70	70	70	70	70	70	68	68
	DC10	C	70	90	68	90	90	90	90	90
	MD82	D	70	70	70	70	70	70	68	68
	A320	D	70	70	70	70	70	70	68	68
	B737	D	70	70	70	70	70	70	68	68
	CRJ7	E	70	70	70	70	70	70	68	68
	E170	E	70	70	70	70	70	70	68	68

As LHR has a relatively high percentage of heavy aircraft operations, almost 50% reduction in wind-related delays at Heathrow is achieved. However, there is a significant difference between the fleet mixes operated at LHR and ORD; this research proves fewer benefits from TBS comparing with an equivalent time-based RECAT II separation.

7. Conclusions and Recommendations

This thesis focus on four main targets based on ASDE-X data and RUNSIM simulations:

- Identifying and re-generate buffer times for different in-trail arrival aircraft groups
- Estimating how many times will ROT limiting the arrival operations under RECAT II, and simulate the potential recovery of runway throughputs under RECAT II with a 95 percentile ROT separation rule comparing with RECAT I
- Estimating the potential go-arounds with and without an additional 95 percentile ROT separation rule under RECAT II
- Estimating the potential gains by applying an equivalent time-based RECAT II separation at ORD

The research shows that there is a significant difference in buffer times for different in-trail groups. For example, based on the buffer time analysis at ORD, CLT and DEN airports, the average buffer time trends to be 10 seconds less when the leading aircraft belongs to wake group ‘E' with the following aircraft belongs to wake group ‘D' (‘E-D' in-trail group) comparing with a ‘D-E' in-trail group.

The research also shows that buffer times differ with airport elevations. The overall buffer time distribution has a mean value of 19 seconds with a standard deviation of 13 at ORD and a mean value of 24 seconds at DEN. The correlated airport elevations are 668 feet for ORD and 5,430 feet for DEN. As a result, two datasets of buffer times are created for airports below 2,500 feet and above 2,500 feet. The buffer times are regenerated based on a data-based kernel distribution for different in-trail arrival groups in RUNSIM.

This thesis assesses the potential operations limited by ROT and throughput under RECAT II with and without an additional 95 percentile ROT separation rule by RUNSIM. The research shows ROT will be a limiting factor to runway throughputs (e.g., up to 9.75% at ORD) under RECAT II. An additional ROT separation rule is suggested to decrease the number of times ROT limits the runway throughput. The research shows by adding a 95 percentile ROT separation rule, less than 2% of the time the separations between in-trail arrivals will be influenced by ROT for each runway end at ORD and

CLT under RECAT II. The simulation results also show that RECAT II can still gain up to 15% and 16% runway throughput for main arrival runways at ORD and CLT respectively with a 95 percentile ROT constraint comparing with RECAT I. In conclusion, an additional 95 percentile ROT separation rule is recommended to reduce potential ROT conflictions with wake vortex separation and can still gain throughput capacities under RECAT II.

The study also estimates the potential go-arounds caused by ROT with and without a 95 percentile ROT constraint under RECAT II for each runway end at ORD, CLT, DEN, and ATL. Based on the current in-trail time intervals, delta separation between RECAT I and RECAT II and the approach speeds, the potential go-arounds caused by ROT with a 95 percentile ROT constraint is less than 1% for any runway ends at these airports. This study validates that a 95 percentile ROT separation rule is necessary to operate RECAT II successfully.

A sample equivalent time-based RECAT II matrix is also generated to diminish the headwind effects to throughput at ORD. The research shows an average of 1.45 and 1.20 arrivals can obtain for east flow and west flow operations respectively under wind conditions at ORD. Comparing with the TBS, the time-based RECAT II separation is decreased up to 20 seconds for some in-trail groups and thus obtain more benefits to throughput capacities. In conclusion, the RECAT II separation performs better than TBS regarding throughput and could still be improved by transferring to an equivalent time-based separation.

REFERENCE

Airnav.com.

Baik, H. and A. Trani (2008). "Framework of a Time-Based Simulation Model for the Analysis of Airfield Operations." Journal of Transportation Engineering.

Boehlert, S. and K. Calvert (2006). Next Generation Air Transportation System: progress and challenges associated with the transformation of the National Airspace System, Washington, D.C.: U.S. Government Accountability Office.

C.L.V.Driessen (2015). Reduced Separation during Final Approach. Aerospace Engineering, Delft University of Technology. **Master of Science**.

Cheng, J., et al. (2016). The Development of Wake Turbulence Recategorization in the United States. AIAA. **2016-3434**.

EUROCONTROL. "Time Based Separation at Heathrow." from https://ec.europa.eu/transport/modes/air/ses/ses-award-2016/projects/time-based-separation-heathrow_en.

EUROCONTROL (2014). "Time Based Separation." from http://www.eurocontrol.int/articles/time-based-separation.

FAA (2008). Advisory Circular. U. S. D. o. Transportation.

FAA (2013b). SAFO 12007. U. S. D. o. Transportation.

FAA (2016). INFO. F. A. Administration.

FAA (2016). NextGen Implementation Plan 2016.

FAA (2016). ORDER JO 7110.123. F. A. Administration. Air Traffic Organization Policy**:** A14.

Fan, Z. and A. Trani (2014). A Computer Model to Predict Potential Turbulence Encounters in the National Airspace System. Civil Engineering, Virginia Tech. **Doctor of Philosophy**.

Gentry, J., et al. (2014). "Airport Capacity Profiles." FAA Report.

Gerz, T., et al. (2002). "Commercial aircraft wake vortices." Progress in aerospace sciences **38**(3): 181-208.

Gu, X., et al. (1995). "Characterization of Gate Location on Aircraft Runway Landing Roll Prediction and Airport Ground Networks Navigation." Transportation Research Record.

Hobeika, A., et al. (1993). "Microcomputer Model for Design and Location of Runway Exits." Journal of Transportation Engineering.

Kim, B., et al. (1996). "Computer Simulation Model for Airplane Landing-Performance Prediction." Transportation Research Record **1562**.

Kolos-Lakatos, T. (2013). The Influence of Runway Occupancy Time and Wake Vortex Separation Requirements on Runway Throughput. Department of Aeronautics & Astronautics. MIT International Center for Air Transportation, Massachusetts Institute of Technology.

Kumar, V. (2010). "Runway Occupancy Time Extraction and Analysis Using Surface Track Data." ResearchGate.

Martinez, J., et al. (2001). "Modeling Airside Airport Operations Using General-Purpose, Activity-Based, Discrete-Event Simulation Tools." Transportation Research Record **1**: 3476.

Mirmohammadsadeghi, N. and A. Trani (2017). Improvements to Airport Systems and Efficiency Using Computer Models and Tools. Civil Engineering, Virginia Tech. **Master of Science**.

Morris, C. and P. Choroba (2013). "Validaion of Time Based Separation concept at London Heathrow Airport." Tenth USA/Europe Air Traffic Management Reserach and Development Seminar.

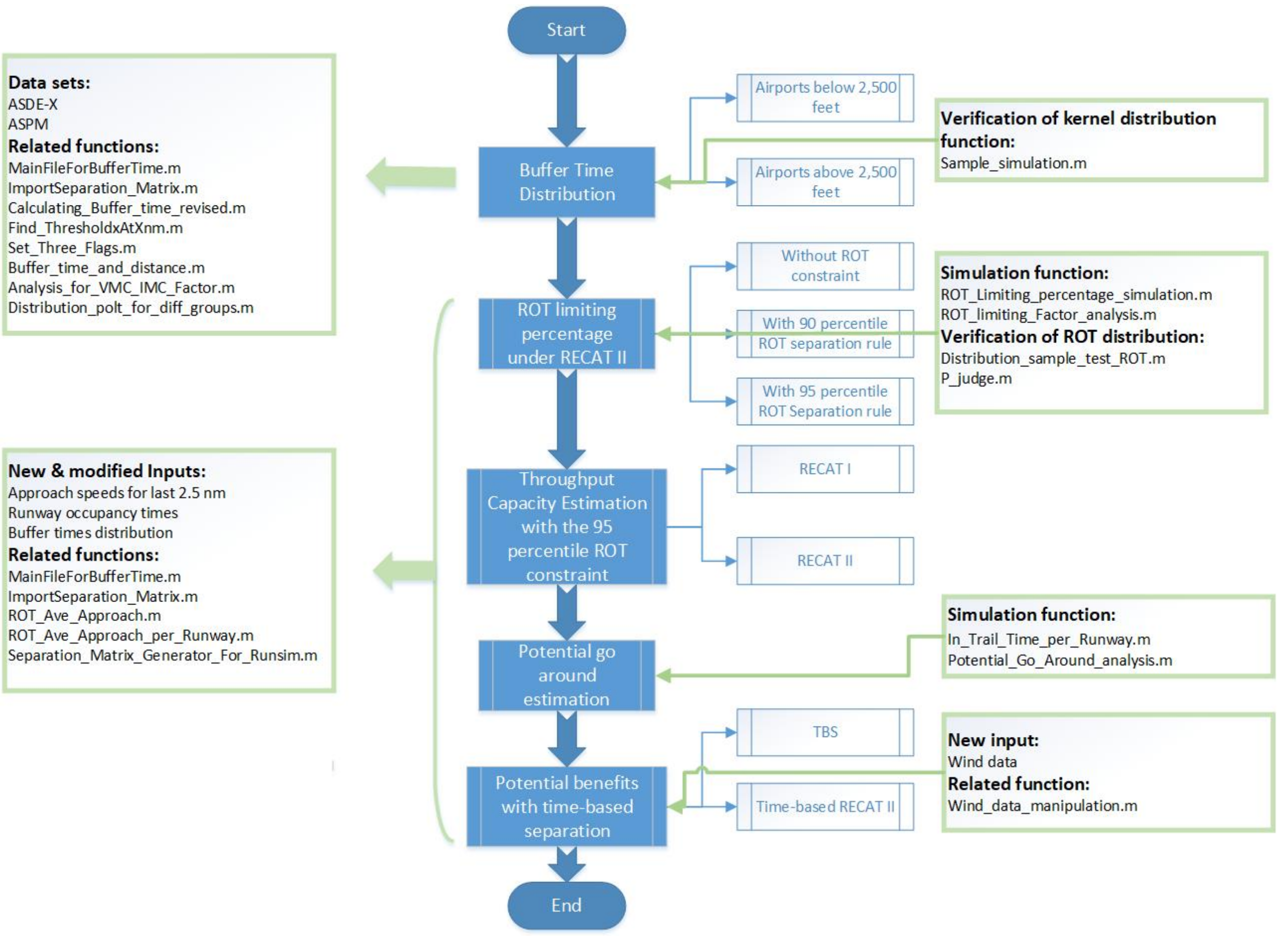
Pu, D. and A. Trani (2014). Demand and Capacity Problems in the Next Generation Air Transportation System. Civil Engineering, Virginia Tech **Master of Science**.

Sherali, H., et al. (1992). "An Integrated Simulation and Dynamic Programming Approach for Determining Optimal Runway Exit Locations." Management Science **38**(7): 1049-1062.

Trani, A., et al. (1992). Runway Exit Designs for Capacity Improvement Demonstrations, Virginia Tech.

Weiss, W. E. and J.N.Barrer (1984). Analysis of Runway Occupancy Time and Separation Data Collection at La Guardia, Boston, And Newark Airports. F. A. Administration.

Appendex 1


Start
Buffer Time Distribution
ROT limiting percentage under RECAT II
Throughput Capacity Estimation with the 95 percentile ROT constraint
Potential go around estimation
Potential benefits with time-based separation
End
Airports below 2,500 feet
Airports above 2,500 feet
Without ROT constraint
With 90 percentile ROT separation rule
With 95 percentile ROT Separation rule
RECAT I
RECAT II
TBS
Time-based RECAT II
Data sets:
ASDE-X
ASPM
Related functions:
MainFileForBufferTime.m
ImportSeparation_Matrix.m
Calculating_Buffer_time_revised.m
Find_ThresholdxAtXnm.m
Set_Three_Flags.m
Buffer_time_and_distance.m
Analysis_for_VMC_IMC_Factor.m
Distribution_polt_for_diff_groups.m
New & modified Inputs:
Approach speeds for last 2.5 nm
Runway occupancy times
Buffer times distribution
Related functions:
MainFileForBufferTime.m
ImportSeparation_Matrix.m
ROT_Ave_Approach.m
ROT_Ave_Approach_per_Runway.m
Separation_Matrix_Generator_For_Runsim.m
Verification of kernel distribution function:
Sample_simulation.m
Simulation function:
ROT_Limiting_percentage_simulation.m
ROT_limiting_Factor_analysis.m
Verification of ROT distribution:
Distribution_sample_test_ROT.m
P_judge.m
Simulation function:
In_Trail_Time_per_Runway.m
Potential_Go_Around_analysis.m
New input:
Wind data
Related function:
Wind_data_manipulation.m

ASDE-X data Manipulation	
MainFileForBufferTime.m	% main file for ASDE-X data manipulation and buffer time analysis
ROT_Ave_Approach.m	% calculate the approach speed and ROT for all of the in-trail arrival operations
ROT_Ave_Approach_per_aircraft_type.m	% calculate the average ROT per each runway end per aircraft type and approach speed per each aircraft type
Calculating_Buffer_time_revised.m	% calculate the buffer time during peak period
Set_Three_Flags.m	% incorporate ASPM data and set three flags for the buffer times
Buffer_time_and_distance.m	% added buffer time and buffer distance. NOTICE that the average buffer time and buffer distance for each in-trail group is GENERATED but NOT SAVED, if want the matrix, need to trace back
analysis_for_VMC_IMC_Factor.m	% plot the kernel distribution of IMC time intervals and VMC time intervals, notice a detailed group analysis is also included in the code, but enabled
Distribution_plot_for_diff_groups.m	% generate the majority of the figures for buffer times
In_Trail_Time_per_Runway.m	% obtain all of the in trail operations for each runway end

Potential_Go_Around_analysis.m	% analysis potential go around under RECAT II with and without the additional ROT separation rule, NOTICE the histogram for time gaps under RECAT II with and without the additional ROT separation rule plot is enabled and need to change for each airport
Sample test file	
Distribution_sample_test_ROT.m	% this file is used to test whether the ROT follows a certain distribution, p_judge.m is used for this file
Sample_simulation.m	% this file is used to test how the number of data points will influence the kernel distribution
box_plot_script.m	% this file is used to make box plots for four main in-trail arrival groups for ORD and CLT
Sub function script file	
ImportSeparation_Matrix.m	% used in several script files, used to import the separation matrix based on separation rule
FindTheIdxAtXnm.m	% a sub-function file in Calculating_Buffer_time_revised.m, aims to identify when the cumdistance is x nm
p_judge.m	% used to test whether a data set flows to a certain distribution, 5 distribution is included (normal, gamma, Poisson, exponential, rayl), could be expanded and used in other files
RUNSIM input script	
Data_generator_for_RUNSIM.m	% this script file generate Grouping_Names.txt Trajectory.txt RECAT_Groups.txt Fleet_Mix.txt based on the manipulated script file, NOTICE this outputs for this file are RUNWAY-END based, need to change runway end name
Separation_Matrix_Generator_For_Runsim.m	% this script generate the arrival and departure separation matrix based on the outs from Data_generator_for_RUNSIM.m, three different separation matrix could be generated: RECAT I, RECAT II, and TBS
ROT_limiting_Factor_analysis.m	% this script file is used to calculate the ROT limiting percentage, NOTICE a in-trail group detailed ROT limiting percentage could also be generated in this code
BufferTimeGenerationDataBased.m	% this script file enables RUNSIM to generate buffer time based on kernel distribution, two data sets are established
Wind input file	
Wind_data_manipulation.m	% this script file generates headwind (from 50% to 99%, for each 10%) for a typical runway, need to change the Runway Heading for each direction. Double check the import root for each airport

MainFileForBufferTime.m

```
% Main script file to obtain buffer time and basic analysis for buffer time
% between different in-trail groups and VMC/IMC factor
clc;
clear all;
close all;

Airport_ID = 'ORD';

if ismac
    ASDE_X_Dir         = '/Users/junqihu/Documents/MATLAB/Research/';
    VMC_Hours_Filename = '/Users/junqihu/ATSL_Git/Wake_Analysis/Input file/VMC_Hours.xlsx';
    Output_Dir         = ['/Users/junqihu/Documents/MATLAB/RESEARCH/Buffer_Analysis/',
Airport_ID, '/'];
else
    ASDE_X_Dir         = 'D:\Datasets\ASDE-X\ORD\';
    VMC_Hours_Filename = '..\Input file\VMC_Hours.xlsx';
    Output_Dir         = ['..\Output\', Airport_ID, '\'];
end

if exist(Output_Dir, 'dir') == 0
   mkdir(Output_Dir);
end

% Obtain approach speed for X nm (8.5 for ORD, 6 for the others)and ROT for
% each operation
ROT_Ave_Approach(Airport_ID, ASDE_X_Dir, Output_Dir);

% Obtain the average approach speed and average ROT based on aircraft type
ROT_Ave_Approach_per_aircraft_type(Output_Dir);

% obtain buffer time from the ADSE-X data
Calculating_Buffer_time_revised(Airport_ID, ASDE_X_Dir, Output_Dir);

% add three flags to obtained buffer matrix (Flag 1: whether location of nearest
% point is at threshold or not, 0 - yes,1 - no but near the threshold, 2 -
% no and far from threshold; % Flag 2 for opening case based on X (approach length) nm average
speed
% 1 - opening case; 0 - closing case; 2 - Marginal case (difference<5); 3
% - lack of data; % Flag 3 for VMC/IMC (1 - VMC, 2 - IMC))
Set_Three_Flags(Airport_ID, VMC_Hours_Filename, Output_Dir);

% Generate buffer time & buffer distance matrix
Buffer_time_and_distance(Output_Dir);

% Obtain the Transfer Factor between VMC/IMC operations
analysis_for_VMC_IMC_Factor(Output_Dir);

% Obtain the distribution plots and the cdf plots for sample in-trail
% Groups and comparing the simulation darta V.S. real data
Distribution_plot_for_diff_groups(Output_Dir);

% calculate in-trail time per runway
In_Trail_Time_per_Runway(Airport_ID, ASDE_X_Dir, Output_Dir);
% potential go around analysis
Potential_Go_Around_analysis(Airport_ID,Output_Dir)
```

ROT_Ave_Approach.m

```
function ROT_Ave_Approach(Airport_ID, ASDE_X_Dir, Output_Dir)

% Data manipulation before analysis
% record the Time for arrival enter runway after midnight
% setup directory and load data by loop

disp('Calculating ROT...');

% prompt                                                              = 'Choose the
airport you want (1 - ORD/2 - DCA/3 - DEN/4 - LGA/ 5 - SAN/6 - IAD/7 - MCO/8 - CLT), give me a
number';
% Airport                                                             = input(prompt);
if strcmp(Airport_ID, 'ORD')                                        == 1
    input_dir                                                       = [ASDE_X_Dir, 'ORD
4 Months'];
    ApproachLength_nm                                               = 2.5;
elseif strcmp(Airport_ID, 'DCA')                                    == 1
```

```
    input_dir                                                              = [ASDE_X_Dir, 'DCA
4 Months'];
elseif strcmp(Airport_ID, 'DEN')                                           == 1
    input_dir                                                              = [ASDE_X_Dir,
'DEN'];
    DEN_heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'LGA')                                           == 1
    input_dir                                                              = [ASDE_X_Dir, '4
Months LGA'];
    LGA_Heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'SAN')                                           == 1
    input_dir                                                              = [ASDE_X_Dir, 'SAN
4 Months'];
    SAN_heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'IAD')                                           == 1
    input_dir                                                              = [ASDE_X_Dir, '4
Months IAD'];
    IAD_Heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'MCO')                                           == 1
    input_dir                                                              = [ASDE_X_Dir, '4
Months MCO'];
    MCO_heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'CLT')                                           == 1
    input_dir                                                              = [ASDE_X_Dir,
'CLT'];
    CLT_heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'ATL')                                           == 1
    input_dir                                                              = [ASDE_X_Dir,
'ATL'];
    ATL_heading
    ApproachLength_nm                                                      = 2.5;
elseif strcmp(Airport_ID, 'LAX')                                           == 1
    input_dir                                                              = [ASDE_X_Dir,
'LAX'];
    LAX_Heading
    ApproachLength_nm                                                      = 2.5;
end
Directory                                                                  = dir(input_dir);
not_wanted                                                                 =
ismember({Directory.name},'.') | ismember({Directory.name},'..') |
ismember({Directory.name},'.DS_Store'); % exclude system dataset
Directory(not_wanted)                                                      = [];
Number_of_data_set                                                         = length(Directory);


for g                                                                      =
1:Number_of_data_set

    disp(['  ', num2str(g), '/', num2str(Number_of_data_set)]);

    load([input_dir,'/' Directory(g).name]); % load data
    % Data manipulation before analysis
    % record the Time for arrival enter runway after midnight
    Num_of_operations                                                      =
length(Airport_Report_File_Pure);
    for n=1:Num_of_operations
            Airport_Report_File_Pure(n).Operational_Runway                 =
char(Airport_Report_File_Pure(n).Operational_Runway);
        if isempty(Airport_Report_File_Pure(n).Index_of_the_Entry_Runway)
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight      = NaN;
        else
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight      =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Airport_Report_File_Pure(n).
Index_of_the_Entry_Runway);
        end % if Airport_Report_File_Pure(n).Index_of_the_Entry_Runway is not empty
    end % for n=1:Num_of_operations
    [~,index]                                                              =
sortrows([Airport_Report_File_Pure.Time_Enter_rwy_After_Midnight].');
    Airport_Report_File_Pure                                               =
Airport_Report_File_Pure(index);
    clear index
    % calculate and record course for each arrival operation
    for idx                                                         =
1:Num_of_operations
        a                                                           =
Airport_Report_File_Pure(idx).Index_of_the_Entry_Runway;
        lat                                                         =
Airport_Report_File_Pure(idx).Latitude_all_Points(1:a);
        lon                                                         =
Airport_Report_File_Pure(idx).Longitude_all_Points(1:a);
        if isempty(lat)
            Airport_Report_File_Pure(idx).course_deg                = [];
        else
            [course,distance]                                       =
legs(lat,lon,'gc');
            Airport_Report_File_Pure(idx).course_deg                = course;
            Airport_Report_File_Pure(idx).cumdistance_nm            =
cumsum(distance,'reverse');
        end % if lat is not empty
    end % for idx=1:Num_of_operations
    clear idx a lat lon course distance
    %------------------------------------------------------
    % find the approach length for each operation, when the first time the navtigation
    % heading is less than 10 degree diverge from the runway, it is the
    % start of approach length

  %  for n                                                                    =
1:Num_of_operations
  %      for i                                                                =
1:length(AllRunwayName)
  %          if strcmp({Airport_Report_File_Pure(n).Operational_Runway},AllRunwayName(i))  ==1
  %              runway_deg                                                   =
runway_deg_input(i);
  %              APHidx_nm(n)                                                 =
FindWhenTheFirstTimeDeviationMoreThan10(Airport_Report_File_Pure(n).course_deg,runway_deg);
  %              if isnan(APHidx_nm(n))
  %                  Airport_Report_File_Pure(n).APHDistance_nm(1)            =
NaN;
  %              else
  %                  Airport_Report_File_Pure(n).APHDistance_nm(1)            =
Airport_Report_File_Pure(n).cumdistance_nm(APHidx_nm(n));
  %              end % if APHidx_nm(n) is not NaN
  %          end
  %      end
  %  end
    %---------------------------------------------------
    % find the time when the aircraft is x nm away from the runway
    % threshold
    for n                                                            =
1:Num_of_operations
        Enter_index                                                  =
Airport_Report_File_Pure(n).Index_of_the_Entry_Runway;
        Airport_Report_File_Pure(n).time_Enter_runway_sec            =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Enter_index);
        Num_of_distance_points_approach                              =
length([Airport_Report_File_Pure(n).cumdistance_nm]);
        time_points_sec                                              =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(1: Num_of_distance_points_approach);
        distance_away_threshold_nm                                   =
Airport_Report_File_Pure(n).cumdistance_nm;
        if isempty(time_points_sec) || isempty(distance_away_threshold_nm)
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec  = NaN;
        elseif length(unique(time_points_sec)) == length(time_points_sec) &&
length(unique(distance_away_threshold_nm)) == length(distance_away_threshold_nm) ...
                && length(unique(time_points_sec))>1 &&
length(unique(distance_away_threshold_nm))>1
            % interp1 need unique time_points_sec and distance_away_threshold_nm
            time_X_nm_away_from_threshold_sec                       =
interp1(distance_away_threshold_nm,time_points_sec,ApproachLength_nm);
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec  =
time_X_nm_away_from_threshold_sec;
        else
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec  = NaN;
            clear time_x_nm_away_from_threshold_sec time_points_sec distance_away_threshold_nm
Enter_index APHidx_nm
        end
        Time_Dur_from_X_nm_to_threshold_sec                                 =
Airport_Report_File_Pure(n).time_Enter_runway_sec -
Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec ;
        if Time_Dur_from_X_nm_to_threshold_sec                              < 0  % time between
two different day
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec + 86400; % change it to the correct time period
        else
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec;
        end
        Airport_Report_File_Pure(n).Ave_Approach_X_nm_knots                 =
ApproachLength_nm./((Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec)./3600);
    end
    clear n Num_of_distance_points_approach time_points_sec
Time_Dur_from_8_5_nm_to_threshold_sec Enter_index distance_away_threshold_nm
time_8_5_nm_away_from_threshold_sec
    %-----------------------------------------------------
    % reord the ROT and average approach speed in a result table
    % Record data for rush hour buffer time for closing cases
    for i=1:Num_of_operations
        Result_ROT_Approach_Speed(i).Aircraft_Type                            =
Airport_Report_File_Pure(i).Aircraft_Type;
        Result_ROT_Approach_Speed(i).Operational_Runway                       =
Airport_Report_File_Pure(i).Operational_Runway;
        Result_ROT_Approach_Speed(i).Operation                                =  Airport_Report
_File_Pure(i).Operation;
        Result_ROT_Approach_Speed(i).Airport                                  =
Airport_Report_File_Pure(i).Airport;
        if isempty(Airport_Report_File_Pure(i).EnterRwy_ExitRwy_Dur_HoldBar)  == 0 &&
length((Airport_Report_File_Pure(i).EnterRwy_ExitRwy_Dur_HoldBar)) == 1
            Result_ROT_Approach_Speed(i).EnterRwy_ExitRwy_Dur_HoldBar         =
Airport_Report_File_Pure(i).EnterRwy_ExitRwy_Dur_HoldBar;
        else
            Result_ROT_Approach_Speed(i).EnterRwy_ExitRwy_Dur_HoldBar         = NaN;
        end
        if  isempty(Airport_Report_File_Pure(i).Ave_Approach_X_nm_knots)      == 0
            Result_ROT_Approach_Speed(i).Ave_Approach_X_nm_knots              =  Airport_Report
_File_Pure(i).Ave_Approach_X_nm_knots;
        else
            Result_ROT_Approach_Speed(i).Ave_Approach_X_nm_knots              = NaN;
        end
    end %  for i=1:Number_of_Intrail_Runway_Names
    if  g                                                                     == 1 % the
initial table
        Result_Table_ROT_Ave_Approach                                         =
struct2table(Result_ROT_Approach_Speed);
    else
        Result_Table_ROT_Ave_Approach                                         =
vertcat(Result_Table_ROT_Ave_Approach,struct2table(Result_ROT_Approach_Speed));
        clear i Airport_Report_File_Pure Num_of_operations Result_ROT_Approach_Speed
    end
end

% Saving Results
save ([Output_Dir, 'Result_Table_ROT_Ave_Approach_1.mat'], 'Result_Table_ROT_Ave_Approach');

return;
```

ROT_Ave_Approach_per_aircraft_type.m

```
function ROT_Ave_Approach_per_aircraft_type(Output_Dir)

% Laoding results
load ([Output_Dir, 'Result_Table_ROT_Ave_Approach_1.mat']);

Result_Table_ROT_Ave_Approach                                              =
table2struct(Result_Table_ROT_Ave_Approach);
RunwayName                                                                 =
unique({Result_Table_ROT_Ave_Approach.Operational_Runway});
Aircraft_List                                                              =
unique({Result_Table_ROT_Ave_Approach.Aircraft_Type})';
Num_of_Aircraft_Type                                                       =
length(Aircraft_List);
Num_of_Operation                                                           =
length(Result_Table_ROT_Ave_Approach);
% do the flitter to ignore the fuzzy numbers for ROT
```

```
for n                                                               = 1:Num_of_Operation
    if Result_Table_ROT_Ave_Approach(n).EnterRwy_ExitRwy_Dur_HoldBar       > 120    % ignore the
unreasonable data
        Result_Table_ROT_Ave_Approach(n).EnterRwy_ExitRwy_Dur_HoldBar      = NaN;
    end
end
%----------------------------------------------------------------------
% Approach speed for all of the operations
Ave_Approach_X_nm_knots                                             =
zeros(Num_of_Aircraft_Type,1);   %preload
Ave_Approach_X_nm_std                                               =
zeros(Num_of_Aircraft_Type,1);    % preload
Num_of_operations_per_Aircraft_type                                 =
zeros(Num_of_Aircraft_Type,1);   %preload
for n                                                               =
1:Num_of_Aircraft_Type
    Find_Aircraft_Operations                                        =
ismember({Result_Table_ROT_Ave_Approach.Aircraft_Type},Aircraft_List(n));
    Ave_Approach_X_nm_knots(n)                                      =
nanmean([Result_Table_ROT_Ave_Approach(Find_Aircraft_Operations).Ave_Approach_X_nm_knots]);
    Ave_Approach_X_nm_std(n)                                        =
nanstd([Result_Table_ROT_Ave_Approach(Find_Aircraft_Operations).Ave_Approach_X_nm_knots]);
    Num_of_operations_per_Aircraft_type(n)                          =
sum(Find_Aircraft_Operations);
end

Ave_Approach_Speed_Aircraft_Result                                  =
zeros(Num_of_Aircraft_Type,3);
for n                                                               =
1:Num_of_Aircraft_Type
    Ave_Approach_Speed_Aircraft_Result(n,1)                         =
Ave_Approach_X_nm_knots(n);
    Ave_Approach_Speed_Aircraft_Result(n,2)                         =
Ave_Approach_X_nm_std(n);
    Ave_Approach_Speed_Aircraft_Result(n,3)                         =
Num_of_operations_per_Aircraft_type(n);
end
clear n i
%---------------------------------------------------------------
% Runway Occupancy Time per runway per aircraft type
%----------------------------
% Split data based on runway information
for n                                                               = 1:Num_of_Operation
    Result_Table_ROT_Ave_Approach(n).RunwayInformation              =
strcat(Result_Table_ROT_Ave_Approach(n).Airport,Result_Table_ROT_Ave_Approach(n).Operational_Ru
nway);
end % for n=1:Num_of_operations
RunwayInformation                                                   =
unique({Result_Table_ROT_Ave_Approach.RunwayInformation});
Num_of_Valid_runway_Name                                            =
length(RunwayInformation)-1; % exclude the No Lat/Alt data
for n                                                               =
1:Num_of_Valid_runway_Name
    index                                                           =
zeros(length(Result_Table_ROT_Ave_Approach),1);
    rwy_id                                                           =
char(RunwayInformation(n));
    for i                                                           = 1:Num_of_Operation
        index(i)                                                    =
strcmp(char(RunwayName(n)),char({Result_Table_ROT_Ave_Approach(i).Operational_Runway}));
    end % for i=1:Num_of_operations
    Location                                                        = find(index); %
identify the location of operations for runway n
    Number_of_Operation_per_runway_name                             = length(Location);
    for j                                                           =
1:Number_of_Operation_per_runway_name
        Data_Split_Runway(1).(rwy_id)(j).Aircraft_Type              =
Result_Table_ROT_Ave_Approach(Location(j)).Aircraft_Type;
        Data_Split_Runway(1).(rwy_id)(j).Operational_Runway         =
Result_Table_ROT_Ave_Approach(Location(j)).Operational_Runway;
        Data_Split_Runway(1).(rwy_id)(j).EnterRwy_ExitRwy_Dur_HoldBar    =
Result_Table_ROT_Ave_Approach(Location(j)).EnterRwy_ExitRwy_Dur_HoldBar;
        % Data_Split_Runway(1).(rwy_id)(j).Ave_Approach_X_nm_knots          =
Result_Table_ROT_Ave_Approach(Location(j)).Ave_Approach_X_nm_knots;
        % Data_Split_Runway(1).(rwy_id)(j).Operation                        =
Result_Table_ROT_Ave_Approach(Location(j)).Operation;
    end % for j=1:Number_of_Operation_per_runway_name
end % for n=1:Num_of_runway_names

clear n i j Number_of_Operation_per_runway_name Location index rwy_id
%------------------------------------------------------------------
% Runway Occupancy Time per runway per aircraft type
for n                                                                     =
1:Num_of_Valid_runway_Name
    rwy_id                                                                =
char(RunwayInformation(n));
    Aircraft_List_per_Runway                                              =
unique({Data_Split_Runway.(rwy_id).Aircraft_Type});
    Num_of_Acft_type_per_rwy                                              =
length(Aircraft_List_per_Runway);
    Num_of_Operation_per_rwy                                              =
length(Data_Split_Runway.(rwy_id));
    if Num_of_Operation_per_rwy                                           > 1
        % Find_Aircraft_Operations                                          =
zeros(Num_of_Operation_per_rwy,1,'logical');    % preload
        Ave_ROT_per_rwy_per_acft_type_holdbar_sec                         =
zeros(Num_of_Acft_type_per_rwy,1);
        %  Ave_Approach_X_nm_per_rwy_per_acft_type_knots                     =
zeros(Num_of_Acft_type_per_rwy,1);
        Ninety_ROT_per_rwy_per_acft_type_holdbar_sec                      =
zeros(Num_of_Acft_type_per_rwy,1);
        Ninety_Five_ROT_per_rwy_per_acft_type_holdbar_sec                 =
zeros(Num_of_Acft_type_per_rwy,1);
        std_ROT_per_rwy_per_acft_type_holdbar_sec                         =
zeros(Num_of_Acft_type_per_rwy,1);
        Frequency_per_rwy_per_acft_type                                   =
zeros(Num_of_Acft_type_per_rwy,1);
        for j                                                             =
1:Num_of_Acft_type_per_rwy
            Find_Aircraft_Operations_per_rwy                              =
strcmp(Aircraft_List_per_Runway(j),{Data_Split_Runway.(rwy_id).Aircraft_Type});
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Aircraft_type   =
Aircraft_List_per_Runway(j);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).ROT_Values_sec  =
[Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).EnterRwy_ExitRwy_Dur_HoldBar];
            Ave_ROT_per_rwy_per_acft_type_holdbar_sec(j)                  =
nanmean([Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).EnterRwy_ExitRwy_Dur_Hold
Bar]);
            Ninety_ROT_per_rwy_per_acft_type_holdbar_sec(j)               =
prctile([Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).EnterRwy_ExitRwy_Dur_Hold
Bar],90);
            Ninety_Five_ROT_per_rwy_per_acft_type_holdbar_sec(j)          =
prctile([Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).EnterRwy_ExitRwy_Dur_Hold
Bar],95);
            std_ROT_per_rwy_per_acft_type_holdbar_sec(j)                  =
std([Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).EnterRwy_ExitRwy_Dur_HoldBar]
,'omitnan');
            % Ave_Approach_X_nm_per_rwy_per_acft_type_knots(j)              =
nanmean([Data_Split_Runway.(rwy_id)(Find_Aircraft_Operations_per_rwy).Ave_Approach_X_nm_knots])
;
            Frequency_per_rwy_per_acft_type(j)                            =
sum(Find_Aircraft_Operations_per_rwy);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Average_ROT_Holdbar_sec =
Ave_ROT_per_rwy_per_acft_type_holdbar_sec(j);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Ninety_ROT_Holdbar_sec  =
Ninety_ROT_per_rwy_per_acft_type_holdbar_sec(j);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Ninety_Five_ROT_Holdbar_sec  =
Ninety_Five_ROT_per_rwy_per_acft_type_holdbar_sec(j);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).std_ROT_Holdbar_sec   =
std_ROT_per_rwy_per_acft_type_holdbar_sec(j);
            %   Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Ave_Approach_x_nm_knots =
Ave_Approach_X_nm_per_rwy_per_acft_type_knots(j);
            Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(j).Frequency       =
Frequency_per_rwy_per_acft_type(j);
        end
    end
end

% Saving Results
save ([Output_Dir, 'Aircraft_List_for_approach_speed'], 'Aircraft_List');
save ([Output_Dir, 'Ave_Approach_speed_per_acft_type_konts.mat'],
'Ave_Approach_Speed_Aircraft_Result');
save ([Output_Dir, 'Ave_ROT_per_rwy_per_acft_type_sec.mat'],
'Ave_ROT_per_rwy_per_acft_type_sec');
```

```
return;
```

Calculating_Buffer_time_revised.m

```
%
% Location: Chicago O'Hare International Airport/ DCA / DEN / LGA / SAN
% / LAX/ IAD / MCO
% Time: From July 1st 2016 to July 31th 2016
% Author: Junqi Hu
%--------------------------------------------------------------------
% This script file aim to manipulate mutipule data for ORD airport to
% analysis the buffer time between arrival-arrival during rush hour and
% prepare for futher analysising about the difference of buffer time between
% different arrival-arrival groups
%--------------------------------------------------------------------

function Calculating_Buffer_time_revised(Airport_ID, ASDE_X_Dir, Output_Dir)

disp('Calculating Buffer Time...');

%------------------------------------------------------------------
% Some constant input

Number_of_15mins_per_day                                            = 96;
Operate_at_Capacity_15mins                                          = 10;
% RECAT_num Choose the RECAT we use, 0 is Legacy, 1 / 1.5 / 2 is RECAT 1 /
% 1.5 / 2, -1 is Legacy revised (the MSR is 2.5 nm)

% prompt                                                             = 'Choose the
airport you want (1 - ORD/2 - DCA/3 - DEN/4 - LGA/5 - SAN/6 - IAD/7 - MCO/8 - CLT/9 - LAX/10 -
SLC), give me a number:>>>';
% Airport                                                             = input(prompt);
if strcmp(Airport_ID, 'ORD')                                        == 1
    input_dir                                                        = [ASDE_X_Dir, 'ORD
4 Months'];
    RECAT_num                                                       = 1.5;
    ApproachLength_nm                                               = 8.5;
elseif strcmp(Airport_ID, 'DCA')                                    == 1    % DCA
    input_dir                                                       = [ASDE_X_Dir, 'DCA
4 Months'];
    RECAT_num                                                       = 0;
    ApproachLength_nm                                               = 6;
elseif strcmp(Airport_ID, 'DEN')                                    == 1    % DEN
    input_dir                                                       = [ASDE_X_Dir,
'DEN'];
    DEN_heading
    ApproachLength_nm                                               = 6;
    RECAT_num                                                       = 1.5;
elseif strcmp(Airport_ID, 'LGA')                                    == 1    % LGA
    input_dir                                                       = [ASDE_X_Dir, '4
Months LGA'];
    LGA_Heading
    RECAT_num                                                       = 1.5;
    ApproachLength_nm                                               = 6;
elseif strcmp(Airport_ID, 'SAN')                                    == 1    % SAN
    input_dir                                                       = [ASDE_X_Dir, 'SAN
4 Months'];
    SAN_heading
    RECAT_num                                                       = 1.5;
    ApproachLength_nm                                               = 6;
elseif strcmp(Airport_ID, 'IAD')                                    == 1    % IAD
    input_dir                                                       = [ASDE_X_Dir, '4
Months IAD'];
    IAD_Heading
    RECAT_num                                                       = -1;
    ApproachLength_nm                                               = 6;
elseif strcmp(Airport_ID, 'MCO')                                    == 1    % MCO
    input_dir                                                       = [ASDE_X_Dir, '4
Months MCO'];
    MCO_heading
    RECAT_num                                                       = -1;
    ApproachLength_nm                                               = 6;
elseif strcmp(Airport_ID, 'CLT')                                    == 1   % CLT
    input_dir                                                       = [ASDE_X_Dir,
'CLT'];
    CLT_heading
    RECAT_num                                                       = 1.5;
    ApproachLength_nm                                                 = 6;
elseif strcmp(Airport_ID, 'LAX')                                      == 1    % LAX
    input_dir                                                         = [ASDE_X_Dir,
'LAX'];
    LAX_Heading
    RECAT_num                                                         = 1.5;
    ApproachLength_nm                                                 = 6;
elseif strcmp(Airport_ID, 'SLC')                                      == 1    % SLC
    input_dir                                                         = [ASDE_X_Dir,
'SLC'];
    SLC_heading
    RECAT_num                                                         = 1;
    ApproachLength_nm                                                 = 6;
elseif strcmp(Airport_ID, 'ATL')                                      == 1    % ATL
    input_dir                                                         = [ASDE_X_Dir,
'ATL'];
    ATL_heading
    RECAT_num                                                         = 1.5;
    ApproachLength_nm                                                 = 6;
end
%------------------------------------------------------------------
% Import the data of RECATISeparation Matrix
RECATMatrix                                                           =
ImportSeparation_Matrix(RECAT_num); %  load the minima distance spearation matrix based on
airport operation
 % Imput data (Aircraft Type/Group)
[~, ~, AircftName]                                                    = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','A2:A254');
[~, ~, AircftGroup]                                                   = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','L2:L254');
[~, ~, WakeClass_Legacy]                                              = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','F2:F254');
%-----------------------------------------------------
% setup directory and load data by loop
Directory                                                             = dir(input_dir);
not_wanted                                                            =
ismember({Directory.name},'.') | ismember({Directory.name},'..') |
ismember({Directory.name},'.DS_Store'); % exclude system dataset
Directory(not_wanted)                                                 = [];
Number_of_files_to_run                                                = length(Directory);
for g                                                                 =
1:Number_of_files_to_run

    disp(['   ', num2str(g), '/', num2str(Number_of_files_to_run)]);

    load([input_dir,'/' Directory(g).name]); % load data
    % Data manipulation before analysis
    % record the Time for arrival enter runway after midnight
    Num_of_operations                                                 =
length(Airport_Report_File_Pure);
    for n                                                             =
1:Num_of_operations
        if isempty(Airport_Report_File_Pure(n).Enter_Rwy_Time)
           Airport_Report_File_Pure(n).Enter_Rwy_Time                 = '.';
        end% if Airport_Report_File_Pure(n).Enter_Rwy_Time is empty
        if isempty(Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar)
           Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar   = NaN;
        end
    end % for n=1:Num_of_operations
    [~,index1]                                                        =
sortrows({Airport_Report_File_Pure.Enter_Rwy_Time}.');
    Airport_Report_File_Pure                                          =
Airport_Report_File_Pure(index1);

    %------------------------------------------------
    %---------------------------------------------
    % run each data set and obtain the  buffer time for peak hour

    % manipulate with the original data set (Airport_Report_File_Pure)
    for n                                                              =
1:Num_of_operations
        if isempty(Airport_Report_File_Pure(n).Index_of_the_Entry_Runway)
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight  = NaN;
        else
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight  =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Airport_Report_File_Pure(n).
Index_of_the_Entry_Runway);
        end % if Airport_Report_File_Pure(n).Index_of_the_Entry_Runway is not empty
    end % for n=1:Num_of_operations

    [~,index]                                                                  =
sortrows([Airport_Report_File_Pure.Time_Enter_rwy_After_Midnight].');
    Airport_Report_File_Pure                                                   =
Airport_Report_File_Pure(index);
    % calculate and record course for each arrival operation
    for idx                                                                    =
1:Num_of_operations
        Index_of_Enter_Rwy                                                     =
Airport_Report_File_Pure(idx).Index_of_the_Entry_Runway;
        lat                                                                    =
Airport_Report_File_Pure(idx).Latitude_all_Points(1:Index_of_Enter_Rwy);
        lon                                                                    =
Airport_Report_File_Pure(idx).Longitude_all_Points(1:Index_of_Enter_Rwy);
        if isempty(lat)
            Airport_Report_File_Pure(idx).course_deg                           = [];
        else
            [course_deg,distance_nm]                                           =
legs(lat,lon,'gc');
            Airport_Report_File_Pure(idx).course_deg                           = course_deg;
            Airport_Report_File_Pure(idx).cumdistance_nm                       =
cumsum(distance_nm,'reverse');
        end % if lat is not empty
    end % for idx=1:Num_of_operations

    %----------------------------------------------------
    % calculate the average approach speed (X nm) per operation

     % find the time when the aircraft is X nm away from the runway
    % threshold
    for n                                                                      =
1:Num_of_operations
        Enter_index                                                            =
Airport_Report_File_Pure(n).Index_of_the_Entry_Runway;
        Airport_Report_File_Pure(n).time_Enter_runway_sec                      =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Enter_index);
        Num_of_distance_points_approach                                        =
length([Airport_Report_File_Pure(n).cumdistance_nm]);
        time_points_sec                                                        =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(1: Num_of_distance_points_approach);
        distance_away_threshold_nm                                             =
Airport_Report_File_Pure(n).cumdistance_nm;
        if isempty(time_points_sec) || isempty(distance_away_threshold_nm)
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec      = NaN;
        elseif length(unique(time_points_sec))                                 ==
length(time_points_sec) && length(unique(distance_away_threshold_nm)) ==
length(distance_away_threshold_nm) ...
                && length(unique(time_points_sec))                             >1 ...
                && length(unique(distance_away_threshold_nm))                  >1
    % interp1 need unique time_points_sec and distance_away_threshold_nm
            time_X_nm_away_from_threshold_sec                                  =
interp1(distance_away_threshold_nm,time_points_sec,ApproachLength_nm);
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec      =
time_X_nm_away_from_threshold_sec;
        else
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec      = NaN;
        end
        Time_Dur_from_X_nm_to_threshold_sec                                    =
Airport_Report_File_Pure(n).time_Enter_runway_sec -
Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec ;
        if Time_Dur_from_X_nm_to_threshold_sec                                 < 0  % time between
two different day
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec + 86400; % change it to the correct time period
        else
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec;
        end
        Airport_Report_File_Pure(n).Ave_Approach_X_nm_knots                    =
ApproachLength_nm./((Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec)./3600);
    end

    %------------------------------------------------------------------
    % Split data based on runway information
    for n                                                                     =
1:Num_of_operations
```

```
        Airport_Report_File_Pure(n).Operational_Runway                    =
char(Airport_Report_File_Pure(n).Operational_Runway);
        Airport_Report_File_Pure(n).RunwayInformation                     =
strcat(Airport_Report_File_Pure(n).Airport,Airport_Report_File_Pure(n).Operational_Runway);
    end % for n=1:Num_of_operations
    RunwayInformation                                                     =
unique({Airport_Report_File_Pure.RunwayInformation});
    RunwayName                                                            =
unique({Airport_Report_File_Pure.Operational_Runway}.');
    AircftGroup_Name                                                      = strmatch('No
Lat/Long Found',RunwayName);
    RunwayInformation(AircftGroup_Name)                                   = {'No_Lat_Long'};
    Num_of_runway_names                                                   =
length(RunwayName);
    for n                                                                 =
1:Num_of_runway_names-1
        index                                                             =
zeros(length({Airport_Report_File_Pure.Operational_Runway}),1);
        rwy_id                                                            =
char(RunwayInformation(n));
        for i                                                             =
1:Num_of_operations
            index(i)                                                      =
strcmp(char(RunwayName(n)),char({Airport_Report_File_Pure(i).Operational_Runway}));
        end % for i=1:Num_of_operations
        Location                                                          = find(index); %
identify the location of operations for runway n
        Number_of_Operation_per_runway_name                               = length(Location);
        for j                                                             =
1:Number_of_Operation_per_runway_name
            Data_Split_Runway(1).(rwy_id)(j).Operational_Runway           =
Airport_Report_File_Pure(Location(j)).Operational_Runway;
            Data_Split_Runway(1).(rwy_id)(j).Speed_Smoothed_knots         =
Airport_Report_File_Pure(Location(j)).Speed_Smoothed_knots;
            Data_Split_Runway(1).(rwy_id)(j).Index_of_the_Entry_Runway    =
Airport_Report_File_Pure(Location(j)).Index_of_the_Entry_Runway;
            Data_Split_Runway(1).(rwy_id)(j).Enter_Rwy_Time               =
Airport_Report_File_Pure(Location(j)).Enter_Rwy_Time;
            Data_Split_Runway(1).(rwy_id)(j).Aircraft_Type                =
Airport_Report_File_Pure(Location(j)).Aircraft_Type;
            Data_Split_Runway(1).(rwy_id)(j).IndexExit                    =
Airport_Report_File_Pure(Location(j)).IndexExit;
            Data_Split_Runway(1).(rwy_id)(j).IndexExitHoldBar             =
Airport_Report_File_Pure(Location(j)).IndexExitHoldBar;
            Data_Split_Runway(1).(rwy_id)(j).Time_Seconds_After_Midnight  =
Airport_Report_File_Pure(Location(j)).Time_Seconds_After_Midnight;
            Data_Split_Runway(1).(rwy_id)(j).Index_Main_File              =
Airport_Report_File_Pure(Location(j)).Index_Main_File;
            Data_Split_Runway(1).(rwy_id)(j).Latitude_all_Points          =
Airport_Report_File_Pure(Location(j)).Latitude_all_Points;
            Data_Split_Runway(1).(rwy_id)(j).Longitude_all_Points         =
Airport_Report_File_Pure(Location(j)).Longitude_all_Points;
            Data_Split_Runway(1).(rwy_id)(j).Time_Enter_rwy_After_Midnight =
Airport_Report_File_Pure(Location(j)).Time_Enter_rwy_After_Midnight;
            Data_Split_Runway(1).(rwy_id)(j).cumdistance_nm               =
Airport_Report_File_Pure(Location(j)).cumdistance_nm;
            Data_Split_Runway(1).(rwy_id)(j).Ave_Approach_X_nm_knots      =
Airport_Report_File_Pure(Location(j)).Ave_Approach_X_nm_knots;
            Data_Split_Runway(1).(rwy_id)(j).EnterRwy_ExitRwy_Dur_HoldBar  =
Airport_Report_File_Pure(Location(j)).EnterRwy_ExitRwy_Dur_HoldBar;
        end % for j=1:Number_of_Operation_per_runway_name
    end % for n=1:Num_of_runway_names

    %-----------------------------------------------------------------
    % pair the Group with arcftlist for arrivals--------------
    Number_of_Valid_runway_name                                           =
length(RunwayInformation)-1; % exclude the 'No Lat/Long Found' data
    for i                                                                 =
1:Number_of_Valid_runway_name
        rwy_id                                                            =
char(RunwayInformation(i));
        Number_of_operations_per_runway                                   =
length(Data_Split_Runway.(rwy_id));
        for n                                                             =
1:Number_of_operations_per_runway
            matches                                                       =
ismember(AircftName,Data_Split_Runway(1).(rwy_id)(n).Aircraft_Type);
            if RECAT_num                                                  == 0 || ...
```

```
               RECAT_num                                                      == -1
            AircftGroup_Name                                                  =
WakeClass_Legacy(matches);
            else
            AircftGroup_Name                                                  =
AircftGroup(matches);
            end
            if isempty(AircftGroup_Name)
                Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group            = char('-');
            else
                Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group            =
char(AircftGroup_Name{1});
            end % if AircftGroup_Name is not empty
        end % for n=1:Number_of_operations_per_runway
    end % for i=1:Number_of_Valid_runway_name

    %------------------------------------------------------------------
    % figure out & record Leading and Folloing aircrft type for runway
    for i                                                                     =
1:Number_of_Valid_runway_name
        rwy_id                                                                =
char(RunwayInformation(i));
        Number_of_Leading_operations_per_runway                               =
length(Data_Split_Runway.(rwy_id))-1; %leading aircraft should exclude the last one as the last
one is the following aircraft
        for n                                                                 =
1:Number_of_Leading_operations_per_runway
            LF_Groups(1).(rwy_id)(n).sequenceLeading                          =
Data_Split_Runway(1).(rwy_id)(n).Aircraft_Type;
            LF_Groups(1).(rwy_id)(n).sequenceFollowing                        =
Data_Split_Runway(1).(rwy_id)(n+1).Aircraft_Type;
            LF_Groups(1).(rwy_id)(n).Leading_Group                            =
Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group;
            LF_Groups(1).(rwy_id)(n).Following_Group                          =
Data_Split_Runway(1).(rwy_id)(n+1).Aircraft_Group;
            LF_Groups(1).(rwy_id)(n).Leading_Ave_Speed_X_nm_knot              =
Data_Split_Runway(1).(rwy_id)(n).Ave_Approach_X_nm_knots;
            LF_Groups(1).(rwy_id)(n).Following_Ave_Speed_X_nm_knot            =
Data_Split_Runway(1).(rwy_id)(n+1).Ave_Approach_X_nm_knots;
            LF_Groups(1).(rwy_id)(n).L_F_Group                                =
strcat(LF_Groups(1).(rwy_id)(n).Leading_Group,LF_Groups(1).(rwy_id)(n).Following_Group);
            LF_Groups(1).(rwy_id)(n).EntertimeLeading                         =
Data_Split_Runway.(rwy_id)(n).Enter_Rwy_Time;
            LF_Groups(1).(rwy_id)(n).EntertimeFollowing                       =
Data_Split_Runway.(rwy_id)(n+1).Enter_Rwy_Time;
            LF_Groups(1).(rwy_id)(n).RunwayInformation                        =
Data_Split_Runway.(rwy_id)(n).Operational_Runway;
            LF_Groups(1).(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar_Leading     =
Data_Split_Runway(1).(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar;
        end % for n=1:Number_of_Leading_operations_per_runway
    end % for i=1:Number_of_Valid_runway_name

    %-----------------------------------------------------
    % Match the Minimum spearation in Operated RECAT/Legacy Group for each sequence
    LF_Runway_names                                                           =
fieldnames(LF_Groups);
    Number_of_Intrail_Runway_Names                                            =
length(LF_Runway_names);
    for i                                                                     =
1:Number_of_Intrail_Runway_Names
        LF_rwy_id                                                             =
char(LF_Runway_names(i));
        Number_of_Intrail_Operations_per_runway                               =
length((LF_Groups(1).(LF_rwy_id)));
        for n                                                                 = 1:
Number_of_Intrail_Operations_per_runway
            [~,Location_of_Acft_Information,~]                                =
intersect(RECATMatrix(:,1),LF_Groups(1).(char(LF_Runway_names(i)))(n).Leading_Group);
            [~,Location_of_RECATMatrix_For_Aircraft,~]                        =
intersect(RECATMatrix(:,1),LF_Groups(1).(char(LF_Runway_names(i)))(n).Following_Group);
            if isempty(Location_of_Acft_Information) ||
isempty(Location_of_RECATMatrix_For_Aircraft)
                LF_Groups(1).(LF_rwy_id)(n).Mini_Separation_RECATI            = NaN;
            else
                LF_Groups(1).(LF_rwy_id)(n).Mini_Separation_RECATI=cell2mat(RECATMatrix(Locatio
n_of_Acft_Information,Location_of_RECATMatrix_For_Aircraft));
            end % if Location_of_Acft_Information & Location_of_RECAT1_5_For_Aircraft is not
empty
```

```
        end % for n=1: Number_of_Intrail_Operations_per_runway
    end % for i=1:Number_of_Intrail_Runway_Names

    %----------------------------------------------------
    % find the location of minimum separation between in-trail groups and
    % calculate the buffer time
    for k                                                                  =
1:Number_of_Intrail_Runway_Names
        Num_of_In_trail_groups_by_Runway                                   =
length(LF_Groups.(char(LF_Runway_names(k))));
        LF_rwy_id                                                          =
char(LF_Runway_names(k));
        for n                                                              = 1:
Num_of_In_trail_groups_by_Runway
            miniSep_nm                                                     = NaN;
            Time_Point_Num_Leading                                         = NaN;
            Time_points_following_sec                                      =
Data_Split_Runway.(LF_rwy_id)(n+1).Time_Seconds_After_Midnight(1:(Data_Split_Runway.(LF_rwy_id)
(n+1).Index_of_the_Entry_Runway-1)); % time of following aircraft
            Time_points_leading_sec                                        =
Data_Split_Runway.(LF_rwy_id)(n).Time_Seconds_After_Midnight(1:(Data_Split_Runway.(LF_rwy_id)(n
).Index_of_the_Entry_Runway-1)); % time of leading aircraft
            Cumdistance_Following_nm                                       =
Data_Split_Runway.(LF_rwy_id)(n+1).cumdistance_nm;% cumdistance of following
            Cumdistance_leading_nm                                         =
Data_Split_Runway.(LF_rwy_id)(n).cumdistance_nm;% cumdistance of leading aircraft
            Num_of_Time_points_for_Leading                                 =
Data_Split_Runway.(LF_rwy_id)(n).Index_of_the_Entry_Runway-1; % Num of time pointes for leading
aircraft to exaNumber_of_operations_per_intrail_runwayming & interplot
            Number_of_time_data_following                                  =
length(Time_points_following_sec);
            Number_of_distance_data_following                              =
length(Cumdistance_Following_nm);
            Num_of_Unique_time_data_following_sec                          =
length(unique(Time_points_following_sec));
            Num_of_Unique_distance_data_following                          =
length(unique(Cumdistance_Following_nm));
            if isempty(Time_points_following_sec) || isempty(Cumdistance_Following_nm) ||
isempty(Time_points_leading_sec) || Number_of_time_data_following==1 ||
Number_of_distance_data_following==1 || Num_of_Unique_time_data_following_sec<
Number_of_time_data_following ||
Num_of_Unique_distance_data_following<Number_of_distance_data_following
                LF_Groups.(LF_rwy_id)(n). MiniSep_nm                       = NaN;
            else
                Cumdistance_following_at_leading_time_point_nm             =
interp1(Time_points_following_sec,Cumdistance_Following_nm,Time_points_leading_sec);  %interplo
ted cumudistance of following
                for i                                                      =
1:Num_of_Time_points_for_Leading
                    % Cumdistance_following_at_leading_time_point_nm         =
interp1(Time_points_following_sec,Cumdistance_Following_nm,Time_points_leading_sec(i));  %inter
ploted cumudistance of following
                    if isnan(Cumdistance_following_at_leading_time_point_nm(i))
                        miniSep_nm                                         = NaN;
                    elseif miniSep_nm                                      <=
Cumdistance_following_at_leading_time_point_nm(i) - Cumdistance_leading_nm(i)
                        miniSep_nm                                         = miniSep_nm;
                    else
                        miniSep_nm                                         =
Cumdistance_following_at_leading_time_point_nm(i) - Cumdistance_leading_nm(i);
                        Time_Point_Num_Leading                             = i;
                    end % if is nan(vq)
                end % i=1:Num_of_Time_points_for_Leading
            end % if time data for leading/following is empty or if time data for
leading/following has repeated value
            LF_Groups.(LF_rwy_id)(n).MiniSep_nm                            = miniSep_nm;
            LF_Groups.(LF_rwy_id)(n).Point_Num                             =
Time_Point_Num_Leading;
            LF_Groups.(LF_rwy_id)(n).Point_threshold                       =
Num_of_Time_points_for_Leading;
            if isnan(miniSep_nm)==0 && isnan(Time_Point_Num_Leading)==0
                LF_Groups.(LF_rwy_id)(n).Distance_Following_Aircraft_nm    =
Data_Split_Runway.(LF_rwy_id)(n).cumdistance_nm(Time_Point_Num_Leading)...
                                                                             +LF_Groups.(LF_rwy
_id)(n).Mini_Separation_RECATI;
                Distance_Following_Aircraft_nm                             =
LF_Groups.(LF_rwy_id)(n).Distance_Following_Aircraft_nm;
```

```
                Time_Following_Aircraft_cross_mini_Sep                     =
interp1(Cumdistance_Following_nm,Time_points_following_sec,Distance_Following_Aircraft_nm);
                Buffer_Time_sec                                            =
Time_Following_Aircraft_cross_mini_Sep-
Data_Split_Runway.(LF_rwy_id)(n).Time_Seconds_After_Midnight(Time_Point_Num_Leading);
                LF_Groups.(LF_rwy_id)(n).Buffer_Time_sec                   = Buffer_Time_sec;
            else
                LF_Groups.(LF_rwy_id)(n).Distance_Following_Aircraft_nm    = NaN;
                LF_Groups.(LF_rwy_id)(n). Buffer_Time_sec                  = NaN;
            end % if minimum separation & time ponit data for leading is available
        end % for n=1: Num_of_In_trail_groups_by_Runway
    end % for k=1:Number_of_Intrail_Runway_Names

    %Num_od_Time_points_for_Leading find the buffer time for the minimum separation between in-
trail
    % groups
    %-----------------------------------------------------
    % count the arrivals for every 15 minutes interval
    Numarr                                                                 = zeros(96,2);
    PeakTimePeriod                                                         =
zeros(length(Numarr),1);
    for i                                                                  =
1:Number_of_Intrail_Runway_Names
        LF_rwy_id                                                          =
char(LF_Runway_names(i));
        Number_of_operations_per_intrail_runway                            =
length((LF_Groups(1).(LF_rwy_id)));
        for k                                                              =
1:Number_of_operations_per_intrail_runway
            LF_Groups(1).(LF_rwy_id)(k).PeakBufferTime_sec                 = NaN;
        end % for k=1:Number_of_operations_per_intrail_runway
    end % for i=1:Number_of_Intrail_Runway_Names

    for
i=                                                                  1:Number_of_Intrail_Runway_N
ames
        LF_rwy_id                                                          =
char(LF_Runway_names(i));
        Number_of_operations_per_intrail_runway                            =
length((LF_Groups(1).(LF_rwy_id)));
        for n                                                              =
1:Number_of_15mins_per_day
            Numarr(n,1)                                                    = 0.25*n;
            Numarr(n,2)                                                    =
sum([Data_Split_Runway.(LF_rwy_id).Time_Enter_rwy_After_Midnight]>=0.0001+900*(n-1) &...
                                                                            [Data_Split_Runway.
(LF_rwy_id).Time_Enter_rwy_After_Midnight]<=900*n);
            if Numarr(n,2)                                                 >=
Operate_at_Capacity_15mins
                PeakTimePeriod(n)                                          = Numarr(n,1);
            else
                PeakTimePeriod(n)                                          = NaN;
            end
            for k                                                          =
1:Number_of_operations_per_intrail_runway
                if (Data_Split_Runway.(LF_rwy_id)(k+1).Time_Enter_rwy_After_Midnight      >=
PeakTimePeriod(n)*3600-15*60 ...
                    && Data_Split_Runway.(LF_rwy_id)(k+1).Time_Enter_rwy_After_Midnight  <=
PeakTimePeriod(n)*3600) ...
                    && Data_Split_Runway.(LF_rwy_id)(k).Time_Enter_rwy_After_Midnight    >=
PeakTimePeriod(n)*3600-15*60 ...
                    && Data_Split_Runway.(LF_rwy_id)(k).Time_Enter_rwy_After_Midnight    <=
PeakTimePeriod(n)*3600
                    LF_Groups(1).(LF_rwy_id)(k).PeakBufferTime_sec         =
LF_Groups(1).(LF_rwy_id)(k).Buffer_Time_sec;
                end % if both the leading aircraft and the following arcraft are within the
peak-15-min-period
            end %  for k=1:Number_of_operations_per_intrail_runway
        end % for n=1:Number_of_15mins_per_day
    end % for i=1:Number_of_Intrail_Runway_Names

    %-----------------------------------------------------------------
    % Record data for rush hour buffer time for closing cases
    for i                                                                  =
1:Number_of_Intrail_Runway_Names
        Number_of_intrail_operations_per_runway                            =
length(LF_Groups(1).(char(LF_Runway_names(i))));
```

```
        PeakBufferTime_is_NaN                                               =
false(Number_of_intrail_operations_per_runway,1);
        for n                                                               =
1:Number_of_intrail_operations_per_runway
            PeakBufferTime_is_NaN(n)                                        =
isnan(LF_Groups(1).(char(LF_Runway_names(i)))(n).PeakBufferTime_sec);
        end % for i=1:Number_of_Intrail_Runway_Names
        index                                                               =
PeakBufferTime_is_NaN==0;
        OperateAtCapacityInTrailInformation.(char(LF_Runway_names(i)))      =
LF_Groups(1).(char(LF_Runway_names(i)))(index);
        if i==1 && g==1 % the initial table
            Result_Table_Buffer_Time                                        =
struct2table(OperateAtCapacityInTrailInformation.(char(LF_Runway_names(1)))); % Create initial
table
        elseif isempty(OperateAtCapacityInTrailInformation.(char(LF_Runway_names(i)))) == 0
            Result_Table_Buffer_Time                                        =
vertcat(Result_Table_Buffer_Time,struct2table(OperateAtCapacityInTrailInformation.(char(LF_Runw
ay_names(i)))));
        end % if i or g is not equal to 1
        clear index PeakBufferTime_is_NaN OperateAtCapacityInTrailInformation
    end %  for i=1:Number_of_Intrail_Runway_Names

    clear RunwayName RunwayInformation PeakTimePeriod OperateAtCapacityInTrailInformation ...
        Numarr n matches Loss_Acft_Info LF_Runway_names LF_Groups Data_Split_Runway APHidx_nm
Airport_Report_File_Pure ...
        Acft_Index Number_of_Intrail_Runway_Names Number_of_intrail_operations_per_runway

end % for g=1:Number_of_data_set

% Saving Results
save ([Output_Dir, 'Result_Table_Buffer_Time_1.mat'], 'Result_Table_Buffer_Time');

return;
```

Set_Three_Flags.m

```
function Set_Three_Flags(Airport_ID, VMC_Hours_Filename, Output_Dir)

disp('Settings Three Flags...');

Round_up_Unit = 5;

% Loading Previous Results
load ([Output_Dir, 'Result_Table_Buffer_Time_1.mat']);

% prompt                                                                    = 'Choose the
airport you want (1 - ORD/2 - DCA/3 - DEN/4 - LGA/5 - SAN/6 - IAD/7 - MCO/8 - CLT), give me a
number';
% Airport                                                                    = input(prompt);
if strcmp(Airport_ID, 'ORD')                                                == 1       % ORD
    [~, ~, raw]                                                             =
xlsread(VMC_Hours_Filename,'ORD');
    raw                                                                     = raw(1:5389,:);
    TimeZone                                                                = 'America/Chicago';
elseif strcmp(Airport_ID, 'DCA')                                            == 1    % DCA
    [~, ~, raw]                                                             =
xlsread(VMC_Hours_Filename,'DCA');
    raw                                                                     = raw(1:7394,:);
    TimeZone                                                                =
'America/New_York';
elseif strcmp(Airport_ID, 'DEN')                                            == 1    % DEN
    [~, ~, raw]                                                             =
xlsread(VMC_Hours_Filename,'DEN');
    raw                                                                     = raw(1:5385,:);
    TimeZone                                                                = 'America/Denver';
elseif strcmp(Airport_ID, 'LGA')                                            == 1    %LGA
    [~, ~, raw]                                                             =
xlsread(VMC_Hours_Filename,'LGA');
    raw                                                                     = raw(1:7416,:);
    TimeZone                                                                =
'America/New_York';
elseif strcmp(Airport_ID, 'SAN')                                            == 1    %SAN
    [~, ~, raw]                                                             =
xlsread(VMC_Hours_Filename,'SAN');
    raw                                                                     = raw(1:7416,:);
    TimeZone                                                                  =
'America/New_York';
elseif strcmp(Airport_ID, 'IAD')                                              == 1    %IAD
    [~, ~, raw]                                                               =
xlsread(VMC_Hours_Filename,'IAD');
    raw                                                                       = raw(1:7146,:);
    TimeZone                                                                  =
'America/New_York';
elseif strcmp(Airport_ID, 'MCO')                                              == 1    %MCO
    [~, ~, raw]                                                               =
xlsread(VMC_Hours_Filename,'MCO');
    raw                                                                       = raw(1:8063,:);
    TimeZone                                                                  =
'America/New_York';
elseif strcmp(Airport_ID, 'CLT')                                              == 1    %CLT
    [~, ~, raw]                                                               =
xlsread(VMC_Hours_Filename,'CLT');
    raw                                                                       = raw(1:7210,:);
    TimeZone                                                                  =
'America/New_York';
elseif strcmp(Airport_ID, 'ATL')                                              == 1    %ATL
    [~, ~, raw]                                                               =
xlsread(VMC_Hours_Filename,'ATL');
    raw                                                                       = raw(1:7320,:);
    TimeZone                                                                  =
'America/New_York';
elseif strcmp(Airport_ID, 'LAX')                                              == 1    %LAX
    [~, ~, raw]                                                               =
xlsread(VMC_Hours_Filename,'LAX');
    raw                                                                       = raw(1:6606,:);
    TimeZone                                                                  =
'America/New_York';
end
% Import the data

raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw))               = {''};

% Replace non-numeric cells with NaN
R                                                                             = cellfun(@(x)
~isnumeric(x) && ~islogical(x),raw(:,2)); % Find non-numeric cells
raw(R,2)                                                                      = {NaN}; % Replace
non-numeric cells

% Create output variable
% data                                                                          =
reshape([raw{:}],size(raw));

% Allocate imported array to column variable names
Local_Date                                                                    = raw(:,1);
Local_Hour                                                                    = raw(:,2);
if ispc % On the Mac, Excel dates are retrieved as datenum, but do not use the same reference
date for 0. On Windows, the date is retrieved as a datestr
    Local_Date_num                                                            =
datenum(Local_Date, 'mm/dd/yyyy');
end

% Clear temporary variables
clearvars data raw R;
%------------------------------------------------------------------
Result_Table_Buffer_Time                                                      =
table2struct(Result_Table_Buffer_Time);
%-------------------------------------------------------
% calculate time interval between two sequence arrivals
Num_of_cases                                                                  =
length(Result_Table_Buffer_Time);
for n                                                                         = 1:Num_of_cases
    Result_Table_Buffer_Time(n).interval                                      =
etime(datevec(Result_Table_Buffer_Time(n).EntertimeFollowing),datevec(Result_Table_Buffer_Time(
n).EntertimeLeading));
end


% Flag 1 for  percentage of last point (Flag 1: whether location of nearest
% point is at threshold or not, 0 - yes,1 - no but near the threshold, 2 -
% no and far from threshold)
for n                                                                         = 1:Num_of_cases
    if
isequal(Result_Table_Buffer_Time(n).Point_Num,Result_Table_Buffer_Time(n).Point_threshold)
```

```
        Result_Table_Buffer_Time(n).Flag_For_Locate_not_Threshold          = 0;
    elseif Result_Table_Buffer_Time(n).Point_threshold-Result_Table_Buffer_Time(n).Point_Num <
10
        Result_Table_Buffer_Time(n).Flag_For_Locate_not_Threshold          = 1;
    else
        Result_Table_Buffer_Time(n).Flag_For_Locate_not_Threshold          = 2;
    end
end
% Flag 2 for opening case based on X (approach length) nm average speed
% 1 - opening case; 0 - closing case; 2 - Marginal case (difference<5); 3
% - lack of data
for n                                                                      = 1:Num_of_cases
    if     abs(Result_Table_Buffer_Time(n).Leading_Ave_Speed_X_nm_knot-
Result_Table_Buffer_Time(n).Following_Ave_Speed_X_nm_knot)  <5
        Result_Table_Buffer_Time(n).Flag_For_Opening_Case                  = 2;  % marginal
case
    elseif Result_Table_Buffer_Time(n).Leading_Ave_Speed_X_nm_knot         >
Result_Table_Buffer_Time(n).Following_Ave_Speed_X_nm_knot && ...
            Result_Table_Buffer_Time(n).Leading_Ave_Speed_X_nm_knot-
Result_Table_Buffer_Time(n).Following_Ave_Speed_X_nm_knot      >5
        Result_Table_Buffer_Time(n).Flag_For_Opening_Case                  = 1;
    elseif Result_Table_Buffer_Time(n).Leading_Ave_Speed_X_nm_knot         <
Result_Table_Buffer_Time(n).Following_Ave_Speed_X_nm_knot && ...
            abs(Result_Table_Buffer_Time(n).Leading_Ave_Speed_X_nm_knot-
Result_Table_Buffer_Time(n).Following_Ave_Speed_X_nm_knot)  >5
        Result_Table_Buffer_Time(n).Flag_For_Opening_Case                  = 0;
    else
        Result_Table_Buffer_Time(n).Flag_For_Opening_Case                  = 3; % NaN
    end
end
%---------------------------------------------------------------------
% change the UTC time to local time for the data set
for n                                                                      = 1:Num_of_cases
    Entertimeleading                                                       =
datetime(Result_Table_Buffer_Time(n).EntertimeLeading,'TimeZone','UTC');
    Entertimeleading.TimeZone                                              = TimeZone;
    Result_Table_Buffer_Time(n).Enter_Rwy_Time_local_date_Leading          =
{datestr(Entertimeleading,'dd-mm-yyyy,hh')};
    %
Result_Table_Buffer_Time(n).Enter_Rwy_Time_local_date_Leading_MM=floor(str2num(datestr(a,'MM'))
/Round_up_Unit)*Round_up_Unit;
    %
Result_Table_Buffer_Time(n).Enter_Rwy_Time_local_date_Leading=strcat(Result_Table_Buffer_Time(n
).Enter_Rwy_Time_local_date_Leading,',',num2str(Result_Table_Buffer_Time(n).Enter_Rwy_Time_loca
l_date_Leading_MM));
end
% Concatenate the ASPM data to accomodate the format with ADSE-X data
% (prepare to matching up)
Num_of_Local_Date                                                          =
length(Local_Date);
Local_Time_str                                                             =
cell(Num_of_Local_Date,1);
Local_Time_str{Num_of_Local_Date,1}                                        = [];

for n                                                                      =
1:Num_of_Local_Date
    if ismac
        Local_Time_str(n)                                                  =
strcat({datestr(693960+Local_Date{n,1},'dd-mm-yyyy')},{','},num2str(Local_Hour{n}));
    elseif ispc
        Local_Time_str(n)                                                  =
strcat({datestr(Local_Date_num(n),'dd-mm-yyyy')},{','},num2str(Local_Hour{n}));
    end
end
% Flag 3 for VMC/IMC (1 - VMC, 2 - IMC)
for n                                                                      = 1:Num_of_cases
    if ismember(Result_Table_Buffer_Time(n).Enter_Rwy_Time_local_date_Leading,Local_Time_str)
        Result_Table_Buffer_Time(n).Flag_for_VMC_1                         = 1;
    else
        Result_Table_Buffer_Time(n).Flag_for_VMC_1                         = 0;
    end
end
%--------------------------------------------------------------------
% Flag 4 for all 3 Flag zero
for n                                                                      = 1:Num_of_cases
    if Result_Table_Buffer_Time(n).Flag_for_VMC_1                          == 0 && ...
            Result_Table_Buffer_Time(n).Flag_For_Opening_Case              == 0 && ...
            Result_Table_Buffer_Time(n).Flag_For_Locate_not_Threshold      == 0
        Result_Table_Buffer_Time(n).Flag_all_zero                          = 1;
    else
        Result_Table_Buffer_Time(n).Flag_all_zero                          = 0;
    end
end
% c                                                                          =
[Result_Table_Buffer_Time.Flag_all_zero];
% b                                                                          = find(c==1);
% figure
% histogram([Result_Table_Buffer_Time(b).PeakBufferTime_sec])
%---------------------------------------------------------------------
% All_Flag_zero_PeakBufferTime_sec=[Result_Table_Buffer_Time(b).PeakBufferTime_sec];
% figure
% cdfplot(All_Flag_zero_PeakBufferTime_sec)
%---------------------------------------------------------------------
% plot the time interval for VMC and IMC
% d = [Result_Table_Buffer_Time.Flag_for_VMC_1];
% VMC_Operations = find(d==1);
% IMC_Operations = find(d==0);
% figure
% cdfplot([Result_Table_Buffer_Time(VMC_Operations).PeakBufferTime_sec]);
% hold on
% cdfplot([Result_Table_Buffer_Time(IMC_Operations).PeakBufferTime_sec]);

% Saving Results
save ([Output_Dir, 'Result_Table_Buffer_Time_2.mat'], 'Result_Table_Buffer_Time');

return;
```

Buffer_time_and_distance.m

```
% To obtain the buffer matrix for buffer time & buffer distance based on in-trail groups for
% IMC conditions
%----------------------------------------------------------
function Buffer_time_and_distance(Output_Dir)

disp('Calculating Buffer Time and Distance...');

% Loading Previous Results
load ([Output_Dir, 'Result_Table_Buffer_Time_2.mat']);

% buffer time
LF_Groups                                                                  =
unique({Result_Table_Buffer_Time.L_F_Group});
L_F_Group_All                                                              =
{Result_Table_Buffer_Time.L_F_Group};
Num_of_LF_Groups                                                           = length(LF_Groups);
% for all buffer time matrix
for n                                                                      = 1:Num_of_LF_Groups
    index_for_each_LF_Group                                                =
ismember([L_F_Group_All],LF_Groups(n));
    BufferTime(n,1)                                                        =
sum(index_for_each_LF_Group);
    BufferTime(n,2)                                                        =
nanmean([Result_Table_Buffer_Time(index_for_each_LF_Group).PeakBufferTime_sec]);
    BufferTime(n,3)                                                        =
nanmedian([Result_Table_Buffer_Time(index_for_each_LF_Group).PeakBufferTime_sec]);
end

% produce IMC buffer matrix
index_IMC                                                                  =
find([Result_Table_Buffer_Time.Flag_for_VMC_1]==0);
Result_Table_Buffer_Time_IMC                                               =
Result_Table_Buffer_Time(index_IMC);
LF_Groups_IMC                                                              =
unique({Result_Table_Buffer_Time_IMC.L_F_Group}');
L_F_Group_All_IMC                                                          =
{Result_Table_Buffer_Time_IMC.L_F_Group};
for n                                                                      = 1:Num_of_LF_Groups
    index_for_each_LF_Group_IMC                                            =
ismember(L_F_Group_All(index_IMC),LF_Groups(n));
    BufferTime_IMC(n,1)                                                    =
sum(index_for_each_LF_Group);
    BufferTime_IMC(n,2)                                                    =
nanmean([Result_Table_Buffer_Time_IMC(index_for_each_LF_Group_IMC).PeakBufferTime_sec]);
    BufferTime_IMC(n,3)                                                    =
nanmedian([Result_Table_Buffer_Time_IMC(index_for_each_LF_Group_IMC).PeakBufferTime_sec]);
```

```
end

%---------------------------------------------------------------------
% buffer distance
Num_of_operations                                                          =
length(Result_Table_Buffer_Time);
for n                                                                      =
1:Num_of_operations
    Result_Table_Buffer_Time(n).bufferdistance_nm = Result_Table_Buffer_Time(n).MiniSep_nm-
Result_Table_Buffer_Time(n).Mini_Separation_RECATI;
end
Num_of_operations_IMC                                                      =
length(Result_Table_Buffer_Time_IMC);
for n                                                                      =
1:Num_of_operations_IMC
    Result_Table_Buffer_Time_IMC(n).bufferdistance_nm =
Result_Table_Buffer_Time_IMC(n).MiniSep_nm-
Result_Table_Buffer_Time_IMC(n).Mini_Separation_RECATI;
end

% for all buffer distance martirx
for n                                                                      = 1:Num_of_LF_Groups
    index_for_each_LF_Group                                                =
ismember(L_F_Group_All,LF_Groups(n));
    BufferDistance(n,1)                                                    =
sum(index_for_each_LF_Group);
    BufferDistance(n,2)                                                    =
nanmean([Result_Table_Buffer_Time(index_for_each_LF_Group).bufferdistance_nm]);
    BufferDistance(n,3)                                                    =
nanmedian([Result_Table_Buffer_Time(index_for_each_LF_Group).bufferdistance_nm]);
end

% produce IMC buffer distance matrix
for n                                                                      =
1:length(LF_Groups)
    index_for_each_LF_Group_IMC                                            =
ismember(L_F_Group_All(index_IMC),LF_Groups(n));
    BufferDistance_IMC(n,1)                                                =
sum(index_for_each_LF_Group_IMC);
    BufferDistance_IMC(n,2)                                                =
nanmean([Result_Table_Buffer_Time_IMC(index_for_each_LF_Group_IMC).bufferdistance_nm]);
    BufferDistance_IMC(n,3)                                                =
nanmedian([Result_Table_Buffer_Time_IMC(index_for_each_LF_Group_IMC).bufferdistance_nm]);
end

% Saving Results
save ([Output_Dir, 'Result_Table_Buffer_Time_3.mat'], 'Result_Table_Buffer_Time');

return;
```

analysis_for_VMC_IMC_Factor.m

```
function analysis_for_VMC_IMC_Factor(Output_Dir)

disp('Analyzing VMC and IMC...');

% Loading Previous Results
load ([Output_Dir, 'Result_Table_Buffer_Time_2.mat']);

LF_Groups                                                                  =
unique({Result_Table_Buffer_Time.L_F_Group});
Num_of_operations_for_PK_period                                            =
length(Result_Table_Buffer_Time);
% clearance of the data
for n                                                                      =
1:Num_of_operations_for_PK_period
    if Result_Table_Buffer_Time(n).interval                                <40 || ...
       Result_Table_Buffer_Time(n).interval                                >300
       Result_Table_Buffer_Time(n).PeakBufferTime_sec                      = NaN;
       Result_Table_Buffer_Time(n).interval                                = NaN;
    end
end


for n                                                                      =
1:Num_of_operations_for_PK_period
    if Result_Table_Buffer_Time(n).Flag_for_VMC_1                          ==0
```

```
       Result_Table_Buffer_Time(n).IMC_PeakBufferTime_sec                  =
Result_Table_Buffer_Time(n).PeakBufferTime_sec;
       Result_Table_Buffer_Time(n).VMC_PeakBufferTime_sec                  = NaN;
    else
       Result_Table_Buffer_Time(n).IMC_PeakBufferTime_sec                  = NaN;
       Result_Table_Buffer_Time(n).VMC_PeakBufferTime_sec                  =
Result_Table_Buffer_Time(n).PeakBufferTime_sec;
    end
end

%--------------------------------------------------
% trying to compare the ks density plot between all VMC and IMC operations
index_IMC                                                                  =
find([Result_Table_Buffer_Time.Flag_for_VMC_1]==0);
Result_Table_Buffer_Time_IMC                                               =
Result_Table_Buffer_Time(index_IMC);

index_VMC                                                                  =
find([Result_Table_Buffer_Time.Flag_for_VMC_1]==1);
Result_Table_Buffer_Time_VMC                                               =
Result_Table_Buffer_Time(index_VMC);

IMC_interval_sec                                                           =
[Result_Table_Buffer_Time_IMC.interval];
VMC_interval_sec                                                           =
[Result_Table_Buffer_Time_VMC.interval];

figure
ksdensity(IMC_interval_sec)
hold on
ksdensity(VMC_interval_sec)
xlim([40,160])




% trying to plot the frequency of DD DE and ED EE group for both VMC and
% IMC
% find out the operations per each group
% index_DD                                                                   =
ismember({Result_Table_Buffer_Time.L_F_Group},'DD');
% index_ED                                                                   =
ismember({Result_Table_Buffer_Time.L_F_Group},'ED');
% index_DE                                                                   =
ismember({Result_Table_Buffer_Time.L_F_Group},'DE');
% index_EE                                                                   =
ismember({Result_Table_Buffer_Time.L_F_Group},'EE');

% if we focus on intervals instead of buffer time
% LF_Groups                                                                  =
unique({Result_Table_Buffer_Time.L_F_Group});
% Num_of_operations_for_PK_period                                            =
length(Result_Table_Buffer_Time);

% for n                                                                      =
1:Num_of_operations_for_PK_period
%    if Result_Table_Buffer_Time(n).Flag_for_VMC_1                          ==0
%      Result_Table_Buffer_Time(n).IMC_interval_sec                        =
Result_Table_Buffer_Time(n).interval;
%      Result_Table_Buffer_Time(n).VMC_interval_sec                        = NaN;
%   else
%      Result_Table_Buffer_Time(n).IMC_interval_sec                        = NaN;
%       Result_Table_Buffer_Time(n).VMC_interval_sec                        =
Result_Table_Buffer_Time(n).interval;
%    end
% end

% if we want to compare with each individual groups
% Find under IMC/VMC conditions, the time interval for DD, ED, DE, EE
% groups
% IMC_DD_interval_sec                                                        =
[Result_Table_Buffer_Time(index_DD).IMC_interval_sec];
% VMC_DD_interval_sec                                                        =
[Result_Table_Buffer_Time(index_DD).VMC_interval_sec];
% IMC_ED_interval_sec                                                        =
[Result_Table_Buffer_Time(index_ED).IMC_interval_sec];
```

```
% VMC_ED_interval_sec                                                    =
[Result_Table_Buffer_Time(index_ED).VMC_interval_sec];
% IMC_DE_interval_sec                                                    =
[Result_Table_Buffer_Time(index_DE).IMC_interval_sec];
% VMC_DE_interval_sec                                                    =
[Result_Table_Buffer_Time(index_DE).VMC_interval_sec];
% IMC_EE_interval_sec                                                    =
[Result_Table_Buffer_Time(index_EE).IMC_interval_sec];
% VMC_EE_interval_sec                                                    =
[Result_Table_Buffer_Time(index_EE).VMC_interval_sec];

% figure
% ksdensity(IMC_DD_interval_sec)
% hold on
% ksdensity(VMC_DD_interval_sec)
% xlim([40,100])

% Saving Results
save ([Output_Dir, 'Result_Table_Buffer_Time_4.mat'], 'Result_Table_Buffer_Time');

return;
```

Distribution_plot_for_diff_groups.m

```
function Distribution_plot_for_diff_groups(Output_Dir)

disp('Plotting...');

% Loading Previous Results
load ([Output_Dir, 'Result_Table_Buffer_Time_4.mat']);

% flitter the data, only pick the data from -20 to 80
Num_of_Operations                                                      =
length(Result_Table_Buffer_Time);
for i                                                                  =
1:Num_of_Operations
    if Result_Table_Buffer_Time(i).PeakBufferTime_sec < -20 ||
Result_Table_Buffer_Time(i).PeakBufferTime_sec > 80
       Result_Table_Buffer_Time(i).PeakBufferTime_sec                  = NaN;
    end
end

index_IMC                                                              =
find([Result_Table_Buffer_Time.Flag_for_VMC_1]==0);
Result_Table_IMC_Buffer_Time                                           =
Result_Table_Buffer_Time(index_IMC);

% find the index for each in-trail group
index_IMC_DD = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'DD');
index_IMC_EF = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'EF');
index_IMC_EE = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'EE');
index_IMC_DE = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'DE');
index_IMC_ED = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'ED');
index_IMC_DB = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'DB');
index_IMC_DC = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'DC');
index_IMC_DF = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'DF');
index_IMC_EB = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'EB');
index_IMC_EC = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'EC');
index_IMC_BD = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'BD');
index_IMC_BE = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'BE');
index_IMC_FD = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'FD');
index_IMC_FE = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'FE');
% Find the index for other groups (changed threshold sthandards)
index_IMC_BB = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'BB');
index_IMC_BC = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'BC');
index_IMC_BF = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'BF');
index_IMC_CB = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'CB');
index_IMC_CC = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'CC');
index_IMC_CD = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'CD');
index_IMC_CE = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'CE');
index_IMC_CF = ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'CF');
% Plot the distribution of buffer time for each in-trail group and show the

% statistic analysis
ORD_four_main_groups =
vertcat([Result_Table_IMC_Buffer_Time(index_IMC_ED)],[Result_Table_IMC_Buffer_Time(index_IMC_EE
)],...
```

```
    [Result_Table_IMC_Buffer_Time(index_IMC_DD)],[Result_Table_IMC_Buffer_Time(index_IMC_DE)]);
buffer_time = [ORD_four_main_groups.PeakBufferTime_sec];
in_trail_group = {ORD_four_main_groups.L_F_Group};
[p,stats] = vartestn(buffer_time',in_trail_group');
% mean and quantile value of buffer time for each group
% DD Group
std_DD                                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec]);
Quantile_DD                                                                       =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail DD groups');
xlabel('Buffer time between D-D in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);


% EE Group
std_EE                                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec]);
Quantile_EE                                                                       =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail EF groups');
xlabel('Buffer time between E-E in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% DE Group
std_DE                                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec]);
Quantile_DE                                                                       =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail DF groups');
xlabel('Buffer time between D-E in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% ED Group
std_ED                                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec]);
Quantile_ED                                                                       =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail ED groups');
xlabel('Buffer time between E-D in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% DB Group
std_DB                                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec]);
Quantile_DB                                                                       =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail DB groups');
```

```
xlabel('Buffer time between D-B in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% DC Group
std_DC                                                                          =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec]);
Quantile_DC                                                                     =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail DC groups');
xlabel('Buffer time between D-C in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% DF Group
std_DF                                                                          =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec]);
Quantile_DF                                                                     =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
hold on
title('Histogram plot for buffer time distributions for in-trail DF groups');
xlabel('Buffer time between D-F in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% EB Group
std_EB                                                                          =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec]);
Quantile_EB                                                                     =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail EB groups');
xlabel('Buffer time between E-B in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% EC Group
std_EC                                                                          =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec]);
Quantile_EC                                                                     =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail EC groups');
xlabel('Buffer time between E-C in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% EF Group
std_EF                                                                          =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec]);
Quantile_EF                                                                     =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail EF groups');
xlabel('Buffer time between E-F in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);
```

```
%BD Group
std_BD                                                                      =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec]);
Quantile_BD                                                                 =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail BD groups');
xlabel('Buffer time between B-D in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

%BE Group
std_BE                                                                      =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec]);
Quantile_BE                                                                 =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec],[.25,.5,.75]);
figure
histogram([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec],'Normalization','prob
ability','BinLimits',[-20,80],'BinWidth',2)
hold on
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec],[.25,.5,.75]),[0.
09 0.09 0.09],'--or')
title('Histogram plot for buffer time distributions for in-trail BE groups');
xlabel('Buffer time between B-E in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Probability (Unitless)','FontSize',30);

% CDF plot all
figure
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec])
title('CDF plot for buffer times for all of the in-trail groups');
xlabel('Buffer time between arrivals during peak period (seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('DB','DC','DD','DE','DF','EB','EC','ED','EE','EF','BD','BE');

% Buffer cdf plot DD, DE, ED, EE
figure
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec])
hold on
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec])
title('CDF plot for buffer times for four major in-trail groups');
xlabel('Buffer time between arrivals during peak period (seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('DD','ED','DE','EE');
%-------------------------------------------------------------
%------------------------------------------------------------
% Comparing Simulated data with real data to validicate the distribution
% DD group random generate data VS real data
```

```
ind                                                                          =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_DD                                                    =
[Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_DD_Cleared                                            =
PeakBufferTime_sec_IMC_DD(ind); % without nan
pd                                                                           =
fitdist(PeakBufferTime_sec_IMC_DD_Cleared','Kernel');
rnd_DD                                                                       =
random(pd,10000,1);
% rnd_DD_for_Kruskal_test                                                      =
random(pd,length(PeakBufferTime_sec_IMC_DD_Cleared),1);
% h_test_DD                                                                    =
chi2gof(rnd_DD,'cdf',pd); % stats test, chi test
% two_groups                                                                   =
[PeakBufferTime_sec_IMC_DD_Cleared' rnd_DD_for_Kruskal_test];
% p_dd                                                                         =
kruskalwallis(two_groups); % kruskalwallis test
% random_number_DD_1000                                                        =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,1000);
figure
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec])
hold on
ecdf(rnd_DD)
title('CDF plot for real & 10,000 simulated buffer time for DD groups (with large data)');
xlabel('Buffer time between D-D in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('Real data', 'Simulation');

% EF group random generate data VS real data
ind                                                                          =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_EF                                                    =
[Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_EF_Cleared                                            =
PeakBufferTime_sec_IMC_EF(ind);
pd                                                                           =
fitdist(PeakBufferTime_sec_IMC_EF_Cleared','Kernel');
rnd_EF                                                                       =
random(pd,10000,1);
% rnd_EF_for_Kruskal_test                                                      =
random(pd,length(PeakBufferTime_sec_IMC_EF_Cleared),1);
% h_test_EF                                                                    =
chi2gof(rnd_EF,'cdf',pd);
% two_groups                                                                   =
[PeakBufferTime_sec_IMC_EF_Cleared' rnd_EF_for_Kruskal_test];
% p_ef                                                                         =
kruskalwallis(two_groups);
% random_number_EF_1000                                                        =
randsample(PeakBufferTime_sec_IMC_EF_Cleared,1000);
figure
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_EF).PeakBufferTime_sec])
hold on
ecdf(rnd_EF)
title('CDF plot for real & 10,000 simulated buffer time for EF groups (with small data)');
xlabel('Buffer time between E-F in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('Real data', 'Simulation');

% DB
ind                                                                          =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_DB                                                    =
[Result_Table_IMC_Buffer_Time(index_IMC_DB).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_DB_Cleared                                            =
PeakBufferTime_sec_IMC_DB(ind);
pd                                                                           =
fitdist(PeakBufferTime_sec_IMC_DB_Cleared','Kernel');
rnd_DB                                                                       =
random(pd,10000,1);
% EE
ind                                                                          =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_EE                                                    =
[Result_Table_IMC_Buffer_Time(index_IMC_EE).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_EE_Cleared                                            =
PeakBufferTime_sec_IMC_EE(ind);
```

```
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_EE_Cleared','Kernel');
rnd_EE                                                                      =
random(pd,10000,1);
% DE
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_DE                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_DE).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_DE_Cleared                                           =
PeakBufferTime_sec_IMC_DE(ind);
pd = fitdist(PeakBufferTime_sec_IMC_DE_Cleared','Kernel');
rnd_DE = random(pd,10000,1);
% ED
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_ED                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_ED).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_ED_Cleared                                           =
PeakBufferTime_sec_IMC_ED(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_ED_Cleared','Kernel');
rnd_ED                                                                      =
random(pd,10000,1);
% DC
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_DC                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_DC).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_DC_Cleared                                           =
PeakBufferTime_sec_IMC_DC(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_DC_Cleared','Kernel');
rnd_DC                                                                      =
random(pd,10000,1);
% DF
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_DF                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_DF).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_DF_Cleared                                           =
PeakBufferTime_sec_IMC_DF(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_DF_Cleared','Kernel');
rnd_DF                                                                      =
random(pd,10000,1);
% EB
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_EB                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_EB).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_EB_Cleared                                           =
PeakBufferTime_sec_IMC_EB(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_EB_Cleared','Kernel');
rnd_EB                                                                      =
random(pd,10000,1);
% EC
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_EC                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_EC).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_EC_Cleared                                           =
PeakBufferTime_sec_IMC_EC(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_EC_Cleared','Kernel');
rnd_EC                                                                      =
random(pd,10000,1);
% BD
ind                                                                         =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_BD                                                   =
[Result_Table_IMC_Buffer_Time(index_IMC_BD).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_BD_Cleared                                           =
PeakBufferTime_sec_IMC_BD(ind);
pd                                                                          =
fitdist(PeakBufferTime_sec_IMC_BD_Cleared','Kernel');
```

```
rnd_BD                                                                =
random(pd,10000,1);
% BE
ind                                                                   =
~isnan([Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec]);
PeakBufferTime_sec_IMC_BE                                             =
[Result_Table_IMC_Buffer_Time(index_IMC_BE).PeakBufferTime_sec];
PeakBufferTime_sec_IMC_BE_Cleared                                     =
PeakBufferTime_sec_IMC_BE(ind);
pd                                                                    =
fitdist(PeakBufferTime_sec_IMC_BE_Cleared','Kernel');
rnd_BE                                                                =
random(pd,10000,1);

% Compare the simulated CDF Plot with real data CDF plot
figure
ecdf(rnd_DB)
hold on
ecdf(rnd_DC)
hold on
ecdf(rnd_DD)
hold on
ecdf(rnd_DE)
hold on
ecdf(rnd_DF)
hold on
ecdf(rnd_EB)
hold on
ecdf(rnd_EC)
hold on
ecdf(rnd_ED)
hold on
ecdf(rnd_EE)
hold on
ecdf(rnd_EF)
hold on
ecdf(rnd_BD)
hold on
ecdf(rnd_BE)
title('CDF plot for 10,000 simulated buffer time for different in-trail groups');
xlabel('Simulated Buffer times between D-D in-trail arrivals during peak period
(seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('DB','DC','DD','DE','DF','EB','EC','ED','EE','EF','BD','BE');

%-----------------------------------------------------------
% identify how many data set will reflect the data polt well
% DD group random generate data VS real data
random_number_50                                                      =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,50);
random_number_100                                                     =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,100);
random_number_300                                                     =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,300);
random_number_500                                                     =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,500);
random_number_1000                                                    =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,1000);
figure
ecdf([Result_Table_IMC_Buffer_Time(index_IMC_DD).PeakBufferTime_sec])
hold on
ecdf(random_number_50)
hold on
ecdf(random_number_100)
hold on
ecdf(random_number_300)
hold on
ecdf(random_number_500)
hold on
ecdf(random_number_1000)
title('CDF plot for buffer times for DD groups under different times of simulations');
xlabel('Buffer time between D-D in-trail arrivals during peak period (seconds)','FontSize',30);
ylabel('Cumulative Probability Density (Unitless)','FontSize',30);
legend('Real Data','Random 50','Random 100','Random 300','Random 500','Random 1000');
% normal fit for All IMC buffer time
ind                                                                   =
~isnan([Result_Table_IMC_Buffer_Time.PeakBufferTime_sec]);
```

```
PeakBufferTime_sec_IMC_Cleared                                             =
[Result_Table_IMC_Buffer_Time(ind).PeakBufferTime_sec];
pd_all_normal                                                              =
fitdist(PeakBufferTime_sec_IMC_Cleared','Normal')
% -----------------------------------------------------------
% For Legacy airport
% HeavyHeavy Group
% index_IMC_LargeLarge                                                       =
ismember({Result_Table_IMC_Buffer_Time.L_F_Group},'LargeLarge');
% std_Large_Large                                                            =
nanstd([Result_Table_IMC_Buffer_Time(index_IMC_LargeLarge).PeakBufferTime_sec]);
% Quantile_LargeLarge                                                        =
quantile([Result_Table_IMC_Buffer_Time(index_IMC_LargeLarge).PeakBufferTime_sec],[.25,.5,.75]);
% figure
%
histogram([Result_Table_IMC_Buffer_Time(index_IMC_LargeLarge).PeakBufferTime_sec],'Normalizatio
n','probability','BinLimits',[-20,80],'BinWidth',2)
% hold on
%
plot(quantile([Result_Table_IMC_Buffer_Time(index_IMC_LargeLarge).PeakBufferTime_sec],[.25,.5,.
75]),[0.09 0.09 0.09],'--or')
% figure
% ecdf([Result_Table_IMC_Buffer_Time(index_IMC_LargeLarge).PeakBufferTime_sec])
%------------------------------
% save data
fid1 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_DD_Cleared.txt'],'w+');
countDD = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_DD_Cleared');
fclose(fid1);
fid2 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_BD_Cleared.txt'],'w+');
countBD = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_BD_Cleared');
fclose(fid2);
fid3 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_DB_Cleared.txt'],'w+');
countDB = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_DB_Cleared');
fclose(fid3);
fid4 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_DE_Cleared.txt'],'w+');
countDE = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_DE_Cleared');
fclose(fid4);
fid5 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_DF_Cleared.txt'],'w+');
countDF = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_DF_Cleared');
fclose(fid5);
fid6 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_EB_Cleared.txt'],'w+');
countEB = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_EB_Cleared');
fclose(fid6);
fid7 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_EC_Cleared.txt'],'w+');
countEC = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_EC_Cleared');
fclose(fid7);
fid8 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_ED_Cleared.txt'],'w+');
countED = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_ED_Cleared');
fclose(fid8);
fid9 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_EE_Cleared.txt'],'w+');
countEE = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_EE_Cleared');
fclose(fid9);
fid10 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_EF_Cleared.txt'],'w+');
countEF = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_EF_Cleared');
fclose(fid10);
fid11 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_BE_Cleared.txt'],'w+');
countBE = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_BE_Cleared');
fclose(fid11);
fid12 = fopen([Output_Dir, 'PeakBufferTime_sec_IMC_DC_Cleared.txt'],'w+');
countDC = fprintf(fid1,'%f\t',PeakBufferTime_sec_IMC_DC_Cleared');
fclose(fid12);

% Saving Results
save ([Output_Dir, 'Result_Table_Buffer_Time_Final.mat'], 'Result_Table_Buffer_Time');

return;
```

In_Trail_Time_per_Runway.m

```
%
% Location: Chicago O'Hare International Airport/ DCA / DEN / LGA / SAN
% / LAX/ IAD / MCO
% Time: From July 1st 2016 to July 31th 2016
% Author: Junqi Hu
%--------------------------------------------------------------------
% This script file aim to manipulate mutipule data for ORD airport to
% analysis the buffer time between arrival-arrival during rush hour and
% prepare for futher analysising about the difference of buffer time between
% different arrival-arrival groups
%--------------------------------------------------------------------

function In_Trail_Time_per_Runway(Airport_ID, ASDE_X_Dir, Output_Dir)

disp('In_Trail_Time_per_Runway...');

%------------------------------------------------------------------
% RECAT_num Choose the RECAT we use, 0 is Legacy, 1 / 1.5 / 2 is RECAT 1 /
% 1.5 / 2, -1 is Legacy revised (the MSR is 2.5 nm)

% prompt                                                                    = 'Choose the
airport you want (1 - ORD/2 - DCA/3 - DEN/4 - LGA/5 - SAN/6 - IAD/7 - MCO/8 - CLT/9 - LAX/10 -
SLC), give me a number:>>>';
% Airport                                                                    = input(prompt);
if strcmp(Airport_ID, 'ORD')                                                == 1
    input_dir                                                                = [ASDE_X_Dir, 'ORD
4 Months'];
    RECAT_num                                                               = 1.5;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'DCA')                                            == 1    % DCA
    input_dir                                                               = [ASDE_X_Dir, 'DCA
4 Months'];
    RECAT_num                                                               = 0;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'DEN')                                            == 1    % DEN
    input_dir                                                               = [ASDE_X_Dir,
'DEN'];
    DEN_heading
    ApproachLength_nm                                                       = 2.5;
    RECAT_num                                                               = 1.5;
elseif strcmp(Airport_ID, 'LGA')                                            == 1    % LGA
    input_dir                                                               = [ASDE_X_Dir, '4
Months LGA'];
    LGA_Heading
    RECAT_num                                                               = 1.5;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'SAN')                                            == 1    % SAN
    input_dir                                                               = [ASDE_X_Dir, 'SAN
4 Months'];
    SAN_heading
    RECAT_num                                                               = 1.5;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'IAD')                                            == 1    % IAD
    input_dir                                                               = [ASDE_X_Dir, '4
Months IAD'];
    IAD_Heading
    RECAT_num                                                               = -1;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'MCO')                                            == 1    % MCO
    input_dir                                                               = [ASDE_X_Dir, '4
Months MCO'];
    MCO_heading
    RECAT_num                                                               = -1;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'CLT')                                            == 1   % CLT
    input_dir                                                               = [ASDE_X_Dir,
'CLT'];
    CLT_heading
    RECAT_num                                                               = 1.5;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'LAX')                                            == 1    % LAX
    input_dir                                                               = [ASDE_X_Dir,
'LAX'];
    LAX_Heading
    RECAT_num                                                               = 1.5;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'SLC')                                            == 1    % SLC
    input_dir                                                               = [ASDE_X_Dir,
'SLC'];
    SLC_heading
    RECAT_num                                                               = 1;
    ApproachLength_nm                                                       = 2.5;
elseif strcmp(Airport_ID, 'ATL')                                            == 1    % SLC
    input_dir                                                               = [ASDE_X_Dir,
'ATL'];
    SLC_heading
    RECAT_num                                                           = 1.5;
    ApproachLength_nm                                                   = 2.5;
end
%------------------------------------------------------------------
% Import the data of RECATISeparation Matrix
RECATMatrix                                                             =
ImportSeparation_Matrix(RECAT_num); %  load the minima distance spearation matrix based on
airport operation
 % Imput data (Aircraft Type/Group)
[~, ~, AircftName]                                                      = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','A2:A254');
[~, ~, AircftGroup]                                                     = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','L2:L254');
[~, ~, WakeClass_Legacy]                                                = xlsread('../Input
file/AircraftInfo(BADA 311).xlsx','Aircraft_Info (RECAT)','F2:F254');
%-----------------------------------------------------
% setup directory and load data by loop
Directory                                                               = dir(input_dir);
not_wanted                                                              =
ismember({Directory.name},'.') | ismember({Directory.name},'..') |
ismember({Directory.name},'.DS_Store'); % exclude system dataset
Directory(not_wanted)                                                   = [];
Number_of_files_to_run                                                  = length(Directory);
for g                                                                   =
1:Number_of_files_to_run

    disp(['  ', num2str(g), '/', num2str(Number_of_files_to_run)]);

    load([input_dir,'/' Directory(g).name]); % load data
    % Data manipulation before analysis
    % record the Time for arrival enter runway after midnight
    Num_of_operations                                                   =
length(Airport_Report_File_Pure);
    for n                                                               =
1:Num_of_operations
        if isempty(Airport_Report_File_Pure(n).Enter_Rwy_Time)
           Airport_Report_File_Pure(n).Enter_Rwy_Time                   = '.';
        end% if Airport_Report_File_Pure(n).Enter_Rwy_Time is empty
        if isempty(Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar)
           Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar     = NaN;
        end
        if length(Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar)> 1
           Airport_Report_File_Pure(n).EnterRwy_ExitRwy_Dur_HoldBar     = NaN;
        end
        if isempty(Airport_Report_File_Pure(n).Aircraft_Type)
           Airport_Report_File_Pure(n).Aircraft_Type                    = 'Not_Recorded';
        end
    end % for n=1:Num_of_operations
    [~,index1]                                                          =
sortrows({Airport_Report_File_Pure.Enter_Rwy_Time}.');
    Airport_Report_File_Pure                                            =
Airport_Report_File_Pure(index1);

    %------------------------------------------------
    %---------------------------------------------
    % run each data set and obtain the  buffer time for peak hour

    % manipulate with the original data set (Airport_Report_File_Pure)
    for n                                                                =
1:Num_of_operations
        if isempty(Airport_Report_File_Pure(n).Index_of_the_Entry_Runway)
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight    = NaN;
        else
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight    =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Airport_Report_File_Pure(n).
Index_of_the_Entry_Runway);
        end % if Airport_Report_File_Pure(n).Index_of_the_Entry_Runway is not empty
    end % for n=1:Num_of_operations

    [~,index]                                                            =
sortrows([Airport_Report_File_Pure.Time_Enter_rwy_After_Midnight].');
    Airport_Report_File_Pure                                             =
Airport_Report_File_Pure(index);


    % manipulate with the original data set (Airport_Report_File_Pure)
    for n                                                                =
1:Num_of_operations
```

```
        if isempty(Airport_Report_File_Pure(n).Index_of_the_Entry_Runway)
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight       = NaN;
        else
            Airport_Report_File_Pure(n).Time_Enter_rwy_After_Midnight       =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Airport_Report_File_Pure(n).
Index_of_the_Entry_Runway);
        end % if Airport_Report_File_Pure(n).Index_of_the_Entry_Runway is not empty
    end % for n=1:Num_of_operations

    [~,index]                                                               =
sortrows([Airport_Report_File_Pure.Time_Enter_rwy_After_Midnight].');
    Airport_Report_File_Pure                                                =
Airport_Report_File_Pure(index);
   %^---------------------------------------------------------------------------------
    % calculate and record course for each arrival operation
    for idx                                                                 =
1:Num_of_operations
        Index_of_Enter_Rwy                                                  =
Airport_Report_File_Pure(idx).Index_of_the_Entry_Runway;
        lat                                                                 =
Airport_Report_File_Pure(idx).Latitude_all_Points(1:Index_of_Enter_Rwy);
        lon                                                                 =
Airport_Report_File_Pure(idx).Longitude_all_Points(1:Index_of_Enter_Rwy);
        if isempty(lat)
            Airport_Report_File_Pure(idx).course_deg                        = [];
        else
            [course_deg,distance_nm]                                        =
legs(lat,lon,'gc');
            Airport_Report_File_Pure(idx).course_deg                        = course_deg;
            Airport_Report_File_Pure(idx).cumdistance_nm                    =
cumsum(distance_nm,'reverse');
        end % if lat is not empty
    end % for idx=1:Num_of_operations

    %----------------------------------------------------
    % calculate the average approach speed (X nm) per operation
     % find the time when the aircraft is X nm away from the runway
    % threshold
    for n                                                                   =
1:Num_of_operations
        Enter_index                                                         =
Airport_Report_File_Pure(n).Index_of_the_Entry_Runway;
        Airport_Report_File_Pure(n).time_Enter_runway_sec                   =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(Enter_index);
        Num_of_distance_points_approach                                     =
length([Airport_Report_File_Pure(n).cumdistance_nm]);
        time_points_sec                                                     =
Airport_Report_File_Pure(n).Time_Seconds_After_Midnight(1: Num_of_distance_points_approach);
        distance_away_threshold_nm                                          =
Airport_Report_File_Pure(n).cumdistance_nm;
        if isempty(time_points_sec) || isempty(distance_away_threshold_nm)
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec   = NaN;
        elseif length(unique(time_points_sec))                              ==
length(time_points_sec) && length(unique(distance_away_threshold_nm)) ==
length(distance_away_threshold_nm) ...
                && length(unique(time_points_sec))                          >1 ...
                && length(unique(distance_away_threshold_nm))               >1
    % interp1 need unique time_points_sec and distance_away_threshold_nm
            time_X_nm_away_from_threshold_sec                               =
interp1(distance_away_threshold_nm,time_points_sec,ApproachLength_nm);
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec   =
time_X_nm_away_from_threshold_sec;
        else
            Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec   = NaN;
        end
        Time_Dur_from_X_nm_to_threshold_sec                                 =
Airport_Report_File_Pure(n).time_Enter_runway_sec -
Airport_Report_File_Pure(n).time_X_nm_away_from_threshold_sec ;
        if Time_Dur_from_X_nm_to_threshold_sec                              < 0  % time between
two different day
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec + 86400; % change it to the correct time period
        else
            Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec =
Time_Dur_from_X_nm_to_threshold_sec;
        end
        Airport_Report_File_Pure(n).Ave_Approach_X_nm_knots                 =
ApproachLength_nm./((Airport_Report_File_Pure(n).Time_Dur_from_X_nm_to_threshold_sec)./3600);
```

```
    end
    %-------------------------------------------------------------------
    % Split data based on runway information
    for n                                                                   =
1:Num_of_operations
        Airport_Report_File_Pure(n).Operational_Runway                      =
char(Airport_Report_File_Pure(n).Operational_Runway);
        Airport_Report_File_Pure(n).RunwayInformation                       =
strcat(Airport_Report_File_Pure(n).Airport,Airport_Report_File_Pure(n).Operational_Runway);
    end % for n=1:Num_of_operations
    RunwayInformation                                                       =
unique({Airport_Report_File_Pure.RunwayInformation});
    RunwayName                                                              =
unique({Airport_Report_File_Pure.Operational_Runway}.');
    AircftGroup_Name                                                        = strmatch('No
Lat/Long Found',RunwayName);
    RunwayInformation(AircftGroup_Name)                                     = {'No_Lat_Long'};
    Num_of_runway_names                                                     =
length(RunwayName);
    for n                                                                   =
1:Num_of_runway_names-1
        index                                                               =
zeros(length({Airport_Report_File_Pure.Operational_Runway}),1);
        rwy_id                                                              =
char(RunwayInformation(n));
        for i                                                               =
1:Num_of_operations
            index(i)                                                        =
strcmp(char(RunwayName(n)),char({Airport_Report_File_Pure(i).Operational_Runway}));
        end % for i=1:Num_of_operations
        Location                                                            = find(index); %
identify the location of operations for runway n
        Number_of_Operation_per_runway_name                                 = length(Location);
        for j                                                               =
1:Number_of_Operation_per_runway_name
            Data_Split_Runway(1).(rwy_id)(j).Operational_Runway             =
Airport_Report_File_Pure(Location(j)).Operational_Runway;
            Data_Split_Runway(1).(rwy_id)(j).Index_of_the_Entry_Runway      =
Airport_Report_File_Pure(Location(j)).Index_of_the_Entry_Runway;
            Data_Split_Runway(1).(rwy_id)(j).Enter_Rwy_Time                 =
Airport_Report_File_Pure(Location(j)).Enter_Rwy_Time;
            Data_Split_Runway(1).(rwy_id)(j).Aircraft_Type                  =
Airport_Report_File_Pure(Location(j)).Aircraft_Type;
            Data_Split_Runway(1).(rwy_id)(j).IndexExit                      =
Airport_Report_File_Pure(Location(j)).IndexExit;
            Data_Split_Runway(1).(rwy_id)(j).IndexExitHoldBar               =
Airport_Report_File_Pure(Location(j)).IndexExitHoldBar;
            Data_Split_Runway(1).(rwy_id)(j).Time_Seconds_After_Midnight    =
Airport_Report_File_Pure(Location(j)).Time_Seconds_After_Midnight;
            Data_Split_Runway(1).(rwy_id)(j).Index_Main_File                =
Airport_Report_File_Pure(Location(j)).Index_Main_File;
            Data_Split_Runway(1).(rwy_id)(j).Latitude_all_Points            =
Airport_Report_File_Pure(Location(j)).Latitude_all_Points;
            Data_Split_Runway(1).(rwy_id)(j).Longitude_all_Points           =
Airport_Report_File_Pure(Location(j)).Longitude_all_Points;
            Data_Split_Runway(1).(rwy_id)(j).Time_Enter_rwy_After_Midnight =
Airport_Report_File_Pure(Location(j)).Time_Enter_rwy_After_Midnight;
            Data_Split_Runway(1).(rwy_id)(j).EnterRwy_ExitRwy_Dur_HoldBar   =
Airport_Report_File_Pure(Location(j)).EnterRwy_ExitRwy_Dur_HoldBar;
            Data_Split_Runway(1).(rwy_id)(j).Operation                      =
Airport_Report_File_Pure(Location(j)).Operation;
            Data_Split_Runway(1).(rwy_id)(j).Approach_Speed_last_X_nm_knots=
Airport_Report_File_Pure(Location(j)).Ave_Approach_X_nm_knots;
        end % for j=1:Number_of_Operation_per_runway_name
    end % for n=1:Num_of_runway_names

    %--------------------------------------------------------------------
    % pair the Group with arcftlist for arrivals--------------
    Number_of_Valid_runway_name                                             =
length(RunwayInformation)-1; % exclude the 'No Lat/Long Found' data
    for i                                                                   =
1:Number_of_Valid_runway_name
        rwy_id                                                              =
char(RunwayInformation(i));
        Number_of_operations_per_runway                                     =
length(Data_Split_Runway.(rwy_id));
        for n                                                               =
1:Number_of_operations_per_runway
```

```
            matches                                                           =
ismember(AircftName,Data_Split_Runway(1).(rwy_id)(n).Aircraft_Type);
            if RECAT_num                                                      == 0 || ...
               RECAT_num                                                      == -1
            AircftGroup_Name                                                  =
WakeClass_Legacy(matches);
            else
            AircftGroup_Name                                                  =
AircftGroup(matches);
            end
            if isempty(AircftGroup_Name)
                Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group               = char('-');
            else
                Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group               =
char(AircftGroup_Name{1});
            end % if AircftGroup_Name is not empty
        end % for n=1:Number_of_operations_per_runway
    end % for i=1:Number_of_Valid_runway_name

    %------------------------------------------------------------------
    % figure out & record Leading and Folloing aircrft type for runway
    for i                                                                     =
1:Number_of_Valid_runway_name
        rwy_id                                                                =
char(RunwayInformation(i));
        Number_of_Leading_operations_per_runway                               =
length(Data_Split_Runway.(rwy_id))-1; %leading aircraft should exclude the last one as the last
one is the following aircraft
        for n                                                                 =
1:Number_of_Leading_operations_per_runway
            LF_Groups(1).(rwy_id)(n).sequenceLeading                          =
Data_Split_Runway(1).(rwy_id)(n).Aircraft_Type;
            LF_Groups(1).(rwy_id)(n).sequenceFollowing                        =
Data_Split_Runway(1).(rwy_id)(n+1).Aircraft_Type;
            LF_Groups(1).(rwy_id)(n).Leading_Group                            =
Data_Split_Runway(1).(rwy_id)(n).Aircraft_Group;
            LF_Groups(1).(rwy_id)(n).Following_Group                          =
Data_Split_Runway(1).(rwy_id)(n+1).Aircraft_Group;
            LF_Groups(1).(rwy_id)(n).L_F_Group                                =
strcat(LF_Groups(1).(rwy_id)(n).Leading_Group,LF_Groups(1).(rwy_id)(n).Following_Group);
            LF_Groups(1).(rwy_id)(n).EntertimeLeading                         =
Data_Split_Runway.(rwy_id)(n).Enter_Rwy_Time;
            LF_Groups(1).(rwy_id)(n).EntertimeFollowing                       =
Data_Split_Runway.(rwy_id)(n+1).Enter_Rwy_Time;
            LF_Groups(1).(rwy_id)(n).RunwayInformation                        =
Data_Split_Runway.(rwy_id)(n).Operational_Runway;
            LF_Groups(1).(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar_Leading     =
Data_Split_Runway(1).(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar;
            LF_Groups(1).(rwy_id)(n).Leading_Operation                        =
Data_Split_Runway(1).(rwy_id)(n).Operation;
            LF_Groups(1).(rwy_id)(n).Following_Operation                      =
Data_Split_Runway(1).(rwy_id)(n+1).Operation;
            if isempty(Data_Split_Runway(1).(rwy_id)(n+1).Approach_Speed_last_X_nm_knots)
            LF_Groups(1).(rwy_id)(n).Following_Approach_Speed_Last_X_nm_knots    = NaN;
            else
            LF_Groups(1).(rwy_id)(n).Following_Approach_Speed_Last_X_nm_knots    =
Data_Split_Runway(1).(rwy_id)(n+1).Approach_Speed_last_X_nm_knots;
            end
            if strcmp('.',LF_Groups(1).(rwy_id)(n).EntertimeLeading) == 1 ||
strcmp('.',LF_Groups(1).(rwy_id)(n).EntertimeFollowing) == 1
            LF_Groups(1).(rwy_id)(n).Time_Interval_sec                        =NaN;
            else
            LF_Groups(1).(rwy_id)(n).Time_Interval_sec                        =
etime(datevec(LF_Groups(1).(rwy_id)(n).EntertimeFollowing,'yyyy-mm-dd
HH:MM:SS'),datevec(LF_Groups(1).(rwy_id)(n).EntertimeLeading,'yyyy-mm-dd HH:MM:SS'));
            end
        end % for n=1:Number_of_Leading_operations_per_runway
    end % for i=1:Number_of_Valid_runway_name

    %-----------------------------------------------------
    % Match the Minimum spearation in Operated RECAT/Legacy Group for each sequence
    LF_Runway_names                                                           =
fieldnames(LF_Groups);
    Number_of_Intrail_Runway_Names                                            =
length(LF_Runway_names);
    for i                                                                     =
1:Number_of_Intrail_Runway_Names
```

```
        LF_rwy_id                                                    =
char(LF_Runway_names(i));
        Number_of_Intrail_Operations_per_runway                      =
length((LF_Groups(1).(LF_rwy_id)));
        for n                                                        = 1:
Number_of_Intrail_Operations_per_runway
            [~,Location_of_Acft_Information,~]                       =
intersect(RECATMatrix(:,1),LF_Groups(1).(char(LF_Runway_names(i)))(n).Leading_Group);
            [~,Location_of_RECATMatrix_For_Aircraft,~]               =
intersect(RECATMatrix(:,1),LF_Groups(1).(char(LF_Runway_names(i)))(n).Following_Group);
            if isempty(Location_of_Acft_Information) ||
isempty(Location_of_RECATMatrix_For_Aircraft)
                LF_Groups(1).(LF_rwy_id)(n).Mini_Separation_RECATI         = NaN;
            else
                LF_Groups(1).(LF_rwy_id)(n).Mini_Separation_RECATI         =
cell2mat(RECATMatrix(Location_of_Acft_Information,Location_of_RECATMatrix_For_Aircraft));
            end % if Location_of_Acft_Information & Location_of_RECAT1_5_For_Aircraft is not
empty
        end % for n=1: Number_of_Intrail_Operations_per_runway
    end % for i=1:Number_of_Intrail_Runway_Names
    % Record data
    for i                                                            =
1:Number_of_Intrail_Runway_Names
        Number_of_intrail_operations_per_runway                      =
length(LF_Groups(1).(char(LF_Runway_names(i))));
        OperateInTrailInformation.(char(LF_Runway_names(i)))         =
LF_Groups(1).(char(LF_Runway_names(i)));
        if i==1 && g==1 % the initial table
            In_Trail_Time_per_Runway                                 =
struct2table(OperateInTrailInformation.(char(LF_Runway_names(1)))); % Create initial table
        else
            In_Trail_Time_per_Runway                                 =
vertcat(In_Trail_Time_per_Runway,struct2table(OperateInTrailInformation.(char(LF_Runway_names(i
)))));
        end % if i or g is not equal to 1
        clear index PeakBufferTime_is_NaN OperateAtCapacityInTrailInformation
    end %  for i=1:Number_of_Intrail_Runway_Names
    clear LF_Groups Airport_Report_File_Pure RunwayInformation RunwayName index1
Data_Split_Runway LF_Runway_names Location
end % for g=1:Number_of_data_set


% Saving Results
save ([Output_Dir, 'In_Trail_Time_per_Runway_1.mat'], 'In_Trail_Time_per_Runway');

return
```

Potential_Go_Around_analysis.m

```
%-----------------------------------------------------------------------
function Potential_Go_Around_analysis(Airport_ID, ASDE_X_Dir, Output_Dir)
% import RECAT 2 matrix
[~, ~, RECATMatrix]                                                  = xlsread('../Input
file/RECAT2_Adjusted.xlsx','Sheet1');
RECATMatrix                                                          =
RECATMatrix(3:end,:);
RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix)) = {''};
Aircraft_List_RECAT                                                  = RECATMatrix(:,1);
%----------------------------------------------------------------------
% manipulate with data sets
load ([Output_Dir, 'In_Trail_Time_per_Runway_1.mat']);

In_Trail_Time_per_Runway                                             =
table2struct(In_Trail_Time_per_Runway);
Num_of_Operations                                                    =
length(In_Trail_Time_per_Runway);
for i                                                                =
1:Num_of_Operations
    if  In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading   > 120
        In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading   = NaN;
    end
    if In_Trail_Time_per_Runway(i).Time_Interval_sec                           <=
In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading
        In_Trail_Time_per_Runway(i).Flag_indicator_ROT_Large_than_intrail       = 1;
        In_Trail_Time_per_Runway(i).Time_Interval_sec                           = 10000;
    elseif In_Trail_Time_per_Runway(i).Time_Interval_sec                       <= 60
```

```
        In_Trail_Time_per_Runway(i).Flag_indicator_ROT_Large_than_intrail       = 2;
        In_Trail_Time_per_Runway(i).Time_Interval_sec                           = 10000;
    elseif isnan(In_Trail_Time_per_Runway(i).Time_Interval_sec)
        In_Trail_Time_per_Runway(i).Time_Interval_sec                           = 10000;
    else
        In_Trail_Time_per_Runway(i).Flag_indicator_ROT_Large_than_intrail       = 0;
    end

    if isnan(In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading)
            In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_sec         = NaN;
            In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_RECATII_sec= NaN;
        else
            In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_sec         =
In_Trail_Time_per_Runway(i).Time_Interval_sec -
In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading;
    end
        Location_Leading                                                            =
ismember(RECATMatrix(:,1),In_Trail_Time_per_Runway(i).sequenceLeading);
        Location_Following                                                          =
ismember(RECATMatrix(:,1),In_Trail_Time_per_Runway(i).sequenceFollowing);
        In_Trail_Time_per_Runway(i).Mini_Separation_RECATII                         =
cell2mat(RECATMatrix(Location_Leading,Location_Following));
        In_Trail_Time_per_Runway(i).Delta_Separation_under_RECATII                  =
In_Trail_Time_per_Runway(i).Mini_Separation_RECATI -
In_Trail_Time_per_Runway(i).Mini_Separation_RECATII;
        In_Trail_Time_per_Runway(i).In_Trail_time_under_RECATII_sec                 =
In_Trail_Time_per_Runway(i).Time_Interval_sec -
In_Trail_Time_per_Runway(i).Delta_Separation_under_RECATII/...
                                                                                In_Trail_T
ime_per_Runway(i).Following_Approach_Speed_Last_X_nm_knots*3600;
        In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_sec             =
In_Trail_Time_per_Runway(i).Time_Interval_sec -
In_Trail_Time_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading;
        In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_RECATII_sec     =
In_Trail_Time_per_Runway(i).In_Trail_time_under_RECATII_sec - ...
                                                                                In_Trail_T
ime_per_Runway(i).EnterRwy_ExitRwy_Dur_HoldBar_Leading;
      if ismember(In_Trail_Time_per_Runway(i).sequenceLeading,RECATMatrix(:,1))      == false
||...
            ismember(In_Trail_Time_per_Runway(i).sequenceFollowing,RECATMatrix(:,1)) == false
        In_Trail_Time_per_Runway(i).Mini_Separation_RECATII                         = NaN;
        In_Trail_Time_per_Runway(i).Delta_Separation_under_RECATII                  = NaN;
        In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_RECATII_sec     = NaN;
        In_Trail_Time_per_Runway(i).In_Trail_time_under_RECATII_sec                 = NaN;
    end
  if In_Trail_Time_per_Runway(i).Delta_Time_between_In_Trail_ROT_RECATII_sec <0
       In_Trail_Time_per_Runway(i).Flag_of_violation                            = 1;
    else
       In_Trail_Time_per_Runway(i).Flag_of_violation                            = 0;
    end
    end
%-----------------------------------------------------------------------------

Runway_Names                                                                    =
unique({In_Trail_Time_per_Runway.RunwayInformation});
Num_of_runway_names                                                             =
length(Runway_Names);
for i                                                                           =
1:Num_of_runway_names
    Runway_ID(i)                                                                =
{strcat(Airport_ID,char(Runway_Names(i)))};
end


for n                                                                           =
1:Num_of_runway_names
    index                                                                       =
zeros(Num_of_Operations,1);
    rwy_id                                                                      =
char(Runway_ID(n));
    for i                                                                       =
1:Num_of_Operations
        index(i)                                                                =
strcmp(char(Runway_Names(n)),char({In_Trail_Time_per_Runway(i).RunwayInformation}));
    end % for i=1:Num_of_operations
    Location                                                                    = find(index); %
identify the location of operations for runway n
    Number_of_Operation_per_runway_name                                         = length(Location);
    for j                                                                    =
1:Number_of_Operation_per_runway_name
        Data_Split_Runway(1).(rwy_id)(j)                                     =
In_Trail_Time_per_Runway(Location(j));

    end % for j=1:Number_of_Operation_per_runway_name
end % for n=1:Num_of_runway_names
%----------------------------------------------------------------------------------
figure
histogram([In_Trail_Time_per_Runway.Delta_Time_between_In_Trail_ROT_sec],[-
30:5:100],'Normalization','probability');
figure
histogram([In_Trail_Time_per_Runway.Delta_Time_between_In_Trail_ROT_RECATII_sec],[-
30:5:100],'Normalization','probability');

%----------------------------------------------------------------------------------
disp(['Overall CLT ROT viloation percentile is:
',num2str(sum([In_Trail_Time_per_Runway.Flag_of_violation])/Num_of_Operations*100)])
for i                                                                        =
1:Num_of_runway_names
    disp([char(Runway_ID(i)),' ROT violation percentile is:
',num2str(sum([Data_Split_Runway.(char(Runway_ID(i))).Flag_of_violation])...
        /length(Data_Split_Runway.(char(Runway_ID(i))))*100)]);
end
%---------------------------------------------------------------------------------
% by adding a 95 percentile constraint
% load ROT and approach speed data
load([Output_Dir,'Ave_ROT_per_rwy_per_acft_type_sec.mat']);
for i                                                                        =
1:Num_of_runway_names
    rwy_id                                                                   =
char(Runway_ID(i));
    Num_of_operations_per_runway                                             =
length(Data_Split_Runway.(rwy_id));
    for n                                                                    =
1:Num_of_operations_per_runway
        Location_of_acft                                                     =
ismember([Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id).Aircraft_type],...
                                                                              Data_Split_Runway.
(rwy_id)(n).sequenceLeading);
        if sum(Location_of_acft)                                             == 0
            Data_Split_Runway.(rwy_id)(n).Time_Interval_sec_with_constrain_RECATII =
Data_Split_Runway.(rwy_id)(n).In_Trail_time_under_RECATII_sec;
        elseif Data_Split_Runway.(rwy_id)(n).In_Trail_time_under_RECATII_sec   <
Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(Location_of_acft).Ninety_Five_ROT_Holdbar_sec
            Data_Split_Runway.(rwy_id)(n).Time_Interval_sec_with_constrain_RECATII =
Ave_ROT_per_rwy_per_acft_type_sec.(rwy_id)(Location_of_acft).Ninety_Five_ROT_Holdbar_sec;
        else
            Data_Split_Runway.(rwy_id)(n).Time_Interval_sec_with_constrain_RECATII =
Data_Split_Runway.(rwy_id)(n).In_Trail_time_under_RECATII_sec;
        end

        if isnan(Data_Split_Runway.(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar_Leading) ||
isnan(Data_Split_Runway.(rwy_id)(n).Delta_Separation_under_RECATII) ||...
           isnan(Data_Split_Runway.(rwy_id)(n).Following_Approach_Speed_Last_X_nm_knots)
        Data_Split_Runway.(rwy_id)(n).Flag_of_violation_with_ROT_constrain = 0;
        elseif Data_Split_Runway.(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar_Leading   >
Data_Split_Runway.(rwy_id)(n).Time_Interval_sec_with_constrain_RECATII
        Data_Split_Runway.(rwy_id)(n).Flag_of_violation_with_ROT_constrain = 1;
        else
        Data_Split_Runway.(rwy_id)(n).Flag_of_violation_with_ROT_constrain = 0;
        end
       Data_Split_Runway.(rwy_id)(n).Delta_Time_between_In_Trail_ROT_RECATII_sec_with_ROT_const
rain =  Data_Split_Runway.(rwy_id)(n).Time_Interval_sec_with_constrain_RECATII ...
           - Data_Split_Runway.(rwy_id)(n).EnterRwy_ExitRwy_Dur_HoldBar_Leading;
       end
end

% needs to change for each airport
% figure
% histogram([Data_Split_Runway.ORD27L.Delta_Time_between_In_Trail_ROT_RECATII_sec],[-
30:5:100],'Normalization','probability');
% figure
%
histogram([Data_Split_Runway.ORD27L.Delta_Time_between_In_Trail_ROT_RECATII_sec_with_ROT_constr
ain],[-30:5:100],'Normalization','probability');

% calculating the new violation percentile
```

```
for i                                                                   =
1:Num_of_runway_names
    disp([char(Runway_ID(i)),' ROT violation percentile with 95 percentile ROT constrain is:
',num2str(sum([Data_Split_Runway.(char(Runway_ID(i))).Flag_of_violation_with_ROT_constrain])...
        /length(Data_Split_Runway.(char(Runway_ID(i))))*100)]);
end

return
```

Distribution_sample_test_ROT.m

```
load('/Users/junqihu/Documents/MATLAB/Data_set/ORD/Ave_ROT_per_rwy_per_acft_type_sec_ORD.mat')
for i = 1:length(Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec)
    if Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec(i) >
Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).Ninety_Five_ROT_Holdbar_sec
        index(i) = true;
    else
        index(i) = false;
    end
end

a = Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec(index);

p_judge(a',0.1)

b = normrnd(100,1,[100,1]);

p_judge(b,0.1)


for i = 1:length(Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec)
    if isnan(Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec(i))
        index_1(i) = false;
    else
        index_1(i) = true;
    end
end

c = Ave_ROT_per_rwy_per_acft_type_sec.ORD10C(6).ROT_Values_sec(index_1);

p_judge(c',0.1)
```

Sample_simulation.m

```
% sample simulation to check time consuming
[~, ~, AircftName]                                                      =
xlsread('/Users/junqihu/Documents/MATLAB/RESEARCH/AircraftInfo(BADA 311).xlsx','Aircraft_Info
(RECAT)','A2:A253');
[~, ~, AircftGroup]                                                     =
xlsread('/Users/junqihu/Documents/MATLAB/RESEARCH/AircraftInfo(BADA 311).xlsx','Aircraft_Info
(RECAT)','L2:L253');
[~, ~, WakeClass_Legacy]                                                =
xlsread('/Users/junqihu/Documents/MATLAB/RESEARCH/AircraftInfo(BADA 311).xlsx','Aircraft_Info
(RECAT)','F2:F253');
% create sample group of aircrafts
% sample                                                                  =
AircftName(1:100,1);
sample                                                                  =
{Result_Table_Buffer_Time(1:1001).sequenceLeading};
Num_of_operations                                                       = length(sample);
% random_number_DD_500                                                    =
randsample(PeakBufferTime_sec_IMC_DD_Cleared,500);
% random_number_DE_500                                                    =
randsample(PeakBufferTime_sec_IMC_DE_Cleared,500);
% random_number_ED_500                                                    =
randsample(PeakBufferTime_sec_IMC_ED_Cleared,500);
% random_number_EE_500                                                    =
randsample(PeakBufferTime_sec_IMC_EE_Cleared,500);
% pd_dd                                                                   =
fitdist(random_number_DD_500','Kernel');
% pd_de                                                                   =
fitdist(random_number_DE_500','Kernel');
% pd_ed                                                                   =
fitdist(random_number_ED_500','Kernel');
% pd_ee                                                                   =
fitdist(random_number_EE_500','Kernel');
% pd_ef                                                                   =
fitdist(PeakBufferTime_sec_IMC_EF_Cleared','Kernel');% EF 227 data points
```

```
% pd_dc                                                                         =
fitdist(PeakBufferTime_sec_IMC_DC_Cleared','Kernel');% DC 76 data points
% pd_db                                                                         =
fitdist(PeakBufferTime_sec_IMC_DB_Cleared','Kernel');% DB 173 data points
% pd_df                                                                         =
fitdist(PeakBufferTime_sec_IMC_DF_Cleared','Kernel');% DF 114 data points
% pd_eb                                                                         =
fitdist(PeakBufferTime_sec_IMC_EB_Cleared','Kernel');% EB 245 data points
% pd_ec                                                                         =
fitdist(PeakBufferTime_sec_IMC_EC_Cleared','Kernel');% EC 128 data points
% pd_bd                                                                         =
fitdist(PeakBufferTime_sec_IMC_BD_Cleared','Kernel');% BD 154 data points
% pd_be                                                                         =
fitdist(PeakBufferTime_sec_IMC_BE_Cleared','Kernel');% BE 128 data points

Buffertime                                                                    =
zeros(Num_of_operations-1,1);
Leading_Group                                                                 =
cell(Num_of_operations-1,1);
Following_Group                                                               =
cell(Num_of_operations-1,1);
for i = 1:(Num_of_operations-1)
     if ismember(sample(i),AircftName) == 1 && ismember(sample(i+1),AircftName) == 1
        Leading_Group(i)                                                      =
AircftGroup(ismember(AircftName,sample(i)));
        Following_Group(i)                                                    =
AircftGroup(ismember(AircftName,sample(i+1)));
        if ismember(Leading_Group(i),'D') && ismember(Following_Group(i),'B')
            Buffertime(i)                                                        =
random(pd_db,1,1);
        elseif ismember(Leading_Group(i),'D') && ismember(Following_Group(i),'C')
            Buffertime(i)                                                        =
random(pd_dc,1,1);
        elseif ismember(Leading_Group(i),'D') && ismember(Following_Group(i),'D')
            Buffertime(i)                                                        =
random(pd_dd,1,1);
        elseif ismember(Leading_Group(i),'D') && ismember(Following_Group(i),'E')
            Buffertime(i)                                                        =
random(pd_de,1,1);
        elseif ismember(Leading_Group(i),'D') && ismember(Following_Group(i),'F')
            Buffertime(i)                                                        =
random(pd_df,1,1);
        elseif ismember(Leading_Group(i),'E') && ismember(Following_Group(i),'B')
            Buffertime(i)                                                        =
random(pd_eb,1,1);
        elseif ismember(Leading_Group(i),'E') && ismember(Following_Group(i),'C')
            Buffertime(i)                                                        =
random(pd_ec,1,1);
        elseif ismember(Leading_Group(i),'E') && ismember(Following_Group(i),'D')
            Buffertime(i)                                                        =
random(pd_ed,1,1);
        elseif ismember(Leading_Group(i),'E') && ismember(Following_Group(i),'E')
            Buffertime(i)                                                       =
random(pd_ee,1,1);
        elseif ismember(Leading_Group(i),'E') && ismember(Following_Group(i),'F')
            Buffertime(i)                                                        =
random(pd_ef,1,1);
        elseif ismember(Leading_Group(i),'B') && ismember(Following_Group(i),'D')
            Buffertime(i)                                                        =
random(pd_bd,1,1);
        elseif ismember(Leading_Group(i),'B') && ismember(Following_Group(i),'E')
            Buffertime(i)                                                        =
random(pd_be,1,1);
        else
            Buffertime(i)                                                        = 18 + 13*randn;
        end
     else
        Buffertime(i)                                                          = 18 + 13*randn;
     end
end

for i = 1:(Num_of_operations-1)
Random_Number(i)                                                              = randn(1,1);
Buffertime_Normal(i)                                                          = 18 +
13*Random_Number(i);
end

for i = 1:(Num_of_operations-1)
```

```
Buffertime_In_Runsim(i)                                             = normrnd(26.4,16);
end

histogram(Buffertime,-20:5:80)
hold on
histogram(Buffertime_Normal,-20:5:80)

hold on
histogram([Result_Table_IMC_Buffer_Time(1:1000).PeakBufferTime_sec],-20:5:80);

figure
ecdf(Buffertime);
hold on
ecdf(Buffertime_Normal)
hold on
ecdf([Result_Table_IMC_Buffer_Time(1:1000).PeakBufferTime_sec])
hold on
ecdf(Buffertime_In_Runsim)
```

box_plot_script.m

```
figure
cdfplot(Buffer_time_DD_Group_ORD);
hold on
cdfplot(Buffer_time_DE_Group_ORD);
hold on
cdfplot(Buffer_time_EE_Group_ORD);
hold on
cdfplot(Buffer_time_ED_Group_ORD);
hold on
cdfplot(Buffer_time_DD_Group_CLT);
hold on
cdfplot(Buffer_time_DE_Group_CLT);
hold on
cdfplot(Buffer_time_EE_Group_CLT);
hold on
cdfplot(Buffer_time_ED_Group_CLT);

Buffer_time_DD_Group = Result_Table_IMC_Buffer_Time.IMC_PeakBufferTime_sec;

% DD
Buffer_time_DD_Group = Buffer_time_DD_Group_ORD';
for i = 1:length(Buffer_time_DD_Group_ORD)
  Buffer_time_DD(i).buffer_time = Buffer_time_DD_Group_ORD(i);
  Buffer_time_DD(i).in_trail_group = 'ORD_DD';
end
Buffer_time_DD_Group_CLT = Buffer_time_DD_Group_CLT';
for i = length(Buffer_time_DD_Group_ORD)+1:(length(Buffer_time_DD_Group_ORD) +
length(Buffer_time_DD_Group_CLT))
  Buffer_time_DD(i).buffer_time = Buffer_time_DD_Group_CLT(i-length(Buffer_time_DD_Group_ORD));
  Buffer_time_DD(i).in_trail_group = 'CLT_DD';
end
buffer_time_DD = [Buffer_time_DD.buffer_time];
in_trail_group_DD = {Buffer_time_DD.in_trail_group};

[p,stats] = vartestn(buffer_time_DD',in_trail_group_DD');

% EE
Buffer_time_EE_Group_ORD = Buffer_time_EE_Group_ORD';
for i = 1:length(Buffer_time_EE_Group_ORD)
  Buffer_time_EE(i).buffer_time = Buffer_time_EE_Group_ORD(i);
  Buffer_time_EE(i).in_trail_group = 'ORD_EE';
end
Buffer_time_EE_Group_CLT = Buffer_time_EE_Group_CLT';
for i = length(Buffer_time_EE_Group_ORD)+1:(length(Buffer_time_EE_Group_ORD) +
length(Buffer_time_EE_Group_CLT))
  Buffer_time_EE(i).buffer_time = Buffer_time_EE_Group_CLT(i-length(Buffer_time_EE_Group_ORD));
  Buffer_time_EE(i).in_trail_group = 'CLT_EE';
end
buffer_time_EE = [Buffer_time_EE.buffer_time];
in_trail_group_EE = {Buffer_time_EE.in_trail_group};

[p,stats] = vartestn(buffer_time_EE',in_trail_group_EE');

% ED
Buffer_time_ED_Group_ORD = Buffer_time_ED_Group_ORD';
for i = 1:length(Buffer_time_ED_Group_ORD)
```

```
  Buffer_time_ED(i).buffer_time = Buffer_time_ED_Group_ORD(i);
  Buffer_time_ED(i).in_trail_group = 'ORD_ED';
end
Buffer_time_ED_Group_CLT = Buffer_time_ED_Group_CLT';
for i = length(Buffer_time_ED_Group_ORD)+1:(length(Buffer_time_ED_Group_ORD) +
length(Buffer_time_ED_Group_CLT))
  Buffer_time_ED(i).buffer_time = Buffer_time_ED_Group_CLT(i-length(Buffer_time_ED_Group_ORD));
  Buffer_time_ED(i).in_trail_group = 'CLT_ED';
end
buffer_time_ED = [Buffer_time_ED.buffer_time];
in_trail_group_ED = {Buffer_time_ED.in_trail_group};

[p,stats] = vartestn(buffer_time_ED',in_trail_group_ED');

% DE
Buffer_time_DE_Group_ORD = Buffer_time_DE_Group_ORD';
for i = 1:length(Buffer_time_DE_Group_ORD)
  Buffer_time_DE(i).buffer_time = Buffer_time_DE_Group_ORD(i);
  Buffer_time_DE(i).in_trail_group = 'ORD_DE';
end
Buffer_time_DE_Group_CLT = Buffer_time_DE_Group_CLT';
for i = length(Buffer_time_DE_Group_ORD)+1:(length(Buffer_time_DE_Group_ORD) +
length(Buffer_time_DE_Group_CLT))
  Buffer_time_DE(i).buffer_time = Buffer_time_DE_Group_CLT(i-length(Buffer_time_DE_Group_ORD));
  Buffer_time_DE(i).in_trail_group = 'CLT_DE';
end
buffer_time_DE = [Buffer_time_DE.buffer_time];
in_trail_group_DE = {Buffer_time_DE.in_trail_group};

[p,stats] = vartestn(buffer_time_DE',in_trail_group_DE');
```

ImportSeparation_Matrix.m

```
% Import the data of RECAT 1/ 1.5 / 2 / 0 (Legacy) Separation Matrix
function RECATMatrix = ImportSeparation_Matrix(RECAT_num)

if RECAT_num == 1.5
    [~, ~, RECATMatrix]                                                          =
xlsread('../Input file/RECAT1.5_Separation.xlsx','Sheet1');
    RECATMatrix                                                                      =
RECATMatrix(1:7,:);
    RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix))          = {''};
elseif RECAT_num                                                                 == 1
    [~, ~, RECATMatrix]                                                          =
xlsread('../Input file/RECAT1_Separation.xlsx','Sheet1');
    RECATMatrix                                                                      =
RECATMatrix(1:7,:);
    RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix))          = {''};
elseif RECAT_num                                                                 == 2
    [~, ~, RECATMatrix]                                                          =
xlsread('../Input file/RECAT2_Adjusted.xlsx','Sheet1');
    RECATMatrix                                                                      =
RECATMatrix(3:end,:);
    RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix)) = {''};
elseif RECAT_num                                                                 == 0
    [~, ~, RECATMatrix]                                                              =
xlsread('../Input file/Legacy_Separation.xlsx','Sheet1');
    RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix))    = {''};
elseif RECAT_num                                                                 == -1
    [~, ~, RECATMatrix]                                                              =
xlsread('../Input file/Legacy_Separation_rev.xlsx','Sheet1');
    RECATMatrix(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix))    = {''};
end

return;
```

Separation_Matrix_Generator_For_RUNSIM.m

```
clc;
clear all;

Runway_End = 'CLT36R';
```

```
if ismac
    load('/Users/junqihu/ATSL_Git/RUNSIM_MODEL/DATA for Individual Runs/BADA_Grouping.mat');
    load('/Users/junqihu/ATSL_Git/RUNSIM_GUI/ACM_GUI/DATA/Recat_Data/v2/SeparationTimeBased_rev
ised.mat');

    Output_dir = '/Users/junqihu/Desktop/New input for recat 2/';
else
    load('D:\RUNSIM_Git\RUNSIM_MODEL\DATA for Individual Runs\BADA_Grouping.mat');
    load('D:\RUNSIM_Git\RUNSIM_GUI\ACM_GUI\DATA\Recat_Data\v2\SeparationTimeBased_revised.mat')
;

    Output_dir = ['..\Output\', Runway_End, '\'];
end

Initial_Big_matrix_TimeBased(1,1) = {'-'};
Initial_Big_matrix_TimeBased(2,1) = {'-'};
Initial_Big_matrix_TimeBased(1,2) = {'-'};
Initial_Big_matrix_TimeBased(2,2) = {'-'};
Separation_Rule = 'TBS'; % 'TBS', 'RECAT 1', 'RECAT 2'
Weather_Condition = 'IMC';
IMC_VMC_Converting_Factor = 0.99;
% Runway 10C
% Aircraft_Name =
{'A319','A320','B737','B738','B744','B763','B772','CRJ2','CRJ7','E135','E145','E170','MD82','MD
83'};
% Aircraft_Groups = {'D','D','D','D','B','C','B','E','E','E','E','E','D','D'};
%-----------------
% Runway 09L
% Aircraft_Name = {'A319','A320','B737','B738','CRJ2','CRJ7','E135','E145','E170','MD83'};
% Aircraft_Groups = {'D','D','D','D','E','E','E','E','E','D'};
%-----------------
% Runway 10L
% Aircraft_Name =
{'A319','A320','B737','B738','B744','B753','B772','CRJ7','DC10','E145','E170'};
% Aircraft_Groups = {'D','D','D','D','B','D','B','E','C','E','E'};
%-----------------
% Runway 10R
% Aircraft_Name = {'A319','A320','B738','CRJ2','CRJ7','E135','E145','E170','MD82'};
% Aircraft_Groups = {'D','D','D','E','E','E','E','E','D'};
%-----------------
% Runway 27R
% Aircraft_Name = {'A319','A320','B738','CRJ2','CRJ7','E135','E145','E170','MD83'};
% Aircraft_Groups = {'D','D','D','E','E','E','E','E','D'};
%-----------------
% Runway 27R
% Aircraft_Name = {'A319','A320','B738','CRJ2','CRJ7','E135','E145','E170'};
% Aircraft_Groups = {'D','D','D','E','E','E','E','E'};
%----------------
% Runway 28C
 % Aircraft_Name =
{'A319','A320','B737','B738','B744','B763','B772','CRJ2','CRJ7','E135','E145','E170','MD83'};
 % Aircraft_Groups = {'D','D','D','D','B','C','B','E','E','E','E','E','D'};
%-----------------
% Runway 28R
% Aircraft_Name =
{'A319','A320','B737','B738','B744','B753','B772','CRJ2','CRJ7','DC10','E145','E170'};
% Aircraft_Groups = {'D','D','D','D','B','D','B','E','E','C','E','E'};
%---------------
 % Runway 09R
 % Aircraft_Name = {'A319','A320','B738','CRJ2','CRJ7','E145','E170','MD82','MD83'};
 % Aircraft_Groups = {'D','D','D','E','E','E','E','D','D'};
 %---------------
 % Runway 22R
 % Aircraft_Name = {'A319','A320','B738','B753','B772','CRJ2','CRJ7','DC10','E145','E170'};
 % Aircraft_Groups = {'D','D','D','D','B','E','E','C','E','E'};

 %import aircraft name and aircraft groups
fid = fopen([Output_dir, '/Grouping_Names.txt']); %filePointer
Initial_Aircraft_Name = textscan(fid,'%s','Delimiter',',');
Aircraft_Name = Initial_Aircraft_Name{1,1}';
fclose(fid);
fid = fopen([Output_dir, '/Recat_Groups.txt']);
Initial_Aircraft_Groups = textscan(fid,'%s','Delimiter',',');
Aircraft_Groups = Initial_Aircraft_Groups{1,1}';
fclose(fid);
```

```
%---------------
Num_of_acfts = length(Aircraft_Name);
Leading_Aircraft = Aircraft_Name;
Following_Aircraft = Aircraft_Name;
[~, ~, SeparationDeptDept] = xlsread('../Input file/Dep_Dep_Matrix_RECAT_1.xlsx','Sheet1');
SeparationDeptDept(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),SeparationDeptDept)) =
{''};
%-----------------------------------------------------------------------------------------
%% import Minimum_separation matrix
% import RECAT 2 arrival arrival matrix
if strcmp(Separation_Rule,'RECAT 2')
[~, ~, RECATMatrix_II]                                                    = xlsread('../Input
file/RECAT2_Adjusted.xlsx','Sheet1');
RECATMatrix_II                                                            =
RECATMatrix_II(3:end,:);
RECATMatrix_II(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix_II)) = {''};
% import Dep_Dep matrix for RECAT 2

%find the arr-arr separation matrix and dep-dep matrix for RECAT II
for i = 1:length(Leading_Aircraft)
    index_Leading_arr = ismember(RECATMatrix_II(:,1),Leading_Aircraft(i));
    index_Leading_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(i));
    for j = 1:length(Following_Aircraft)
        index_Following_arr = ismember(RECATMatrix_II(:,1),Following_Aircraft(j));
        index_Following_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(j));
Minimum_Separation_Arr_Arr(i,j) =
cell2mat(RECATMatrix_II(index_Leading_arr,index_Following_arr));
Minimum_Separation_Dep_Dep(i,j) =
cell2mat(SeparationDeptDept(index_Leading_dep,index_Following_dep));
    end
end
dlmwrite([Output_dir, '/ Minimum_Separation_Arr_Arr_2.txt'], Minimum_Separation_Arr_Arr,
'newline', 'pc');
dlmwrite([Output_dir, '/ Minimum_Separation_Dep_Dep_2.txt'], Minimum_Separation_Dep_Dep,
'newline', 'pc');

% fid = fopen('Arr_Arr_Separation','a+');
% Arr_Arr_Separation = fprintf(fid,'%f/t',Minimum_Separation_Arr_Arr');
%fclose(fid);

% For RECAT 1.5
elseif strcmp(Separation_Rule,'RECAT 1')
[~, ~, RECATMatrix_1_5]                                                          =
xlsread('../Input file/RECAT1.5_Separation.xlsx','Sheet1');
RECATMatrix_1_5                                                                  =
RECATMatrix_1_5(1:7,:);
RECATMatrix_1_5(cellfun(@(RECAT_num) ~isempty(RECAT_num) && isnumeric(RECAT_num) &&
isnan(RECAT_num),RECATMatrix_1_5))          = {''};
[~, ~, SeparationDeptDept] = xlsread('../Input file/Dep_Dep_Matrix_RECAT_1.xlsx','Sheet1');
SeparationDeptDept(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),SeparationDeptDept)) =
{''};

for i = 1:length(Leading_Aircraft)
    index_Leading_arr = ismember(RECATMatrix_1_5(:,1),Aircraft_Groups(i));
    index_Leading_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(i));
    for j = 1:length(Following_Aircraft)
        index_Following_arr = ismember(RECATMatrix_1_5(:,1),Aircraft_Groups(j));
        index_Following_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(j));
        if strcmp(Weather_Condition, 'VMC') == 1
        Minimum_Separation_Arr_Arr_1_5(i,j) =
IMC_VMC_Converting_Factor*cell2mat(RECATMatrix_1_5(index_Leading_arr,index_Following_arr));
        Minimum_Separation_Dep_Dep_1_5(i,j) =
cell2mat(SeparationDeptDept(index_Leading_dep,index_Following_dep));
        else
        Minimum_Separation_Arr_Arr_1_5(i,j) =
cell2mat(RECATMatrix_1_5(index_Leading_arr,index_Following_arr));
        Minimum_Separation_Dep_Dep_1_5(i,j) =
cell2mat(SeparationDeptDept(index_Leading_dep,index_Following_dep));
        end
    end
end
dlmwrite([Output_dir, '/ Minimum_Separation_Arr_Arr_1_5_', Weather_Condition, '.txt'],
Minimum_Separation_Arr_Arr_1_5, 'newline', 'pc');
dlmwrite([Output_dir, '/ Minimum_Separation_Dep_Dep_1_5_', Weather_Condition, '.txt'],
Minimum_Separation_Dep_Dep_1_5, 'newline', 'pc');
% For TBS
```

```
elseif strcmp(Separation_Rule,'TBS')
[~, ~, TBS] = xlsread('../Input file/TBS.xlsx','Sheet1');
TBS(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),TBS)) = {''};
for i = 1:length(Leading_Aircraft)
    index_Leading_arr = ismember(SeparationTimeBased(:,1),Aircraft_Groups(i));
    index_Leading_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(i));
    for j = 1:length(Following_Aircraft)
        index_Following_arr = ismember((SeparationTimeBased(:,1)),Aircraft_Groups(j));
        index_Following_dep = ismember(SeparationDeptDept(:,1),Aircraft_Groups(j));
Minimum_Separation_Arr_Arr(i,j) =
cell2mat(SeparationTimeBased(index_Leading_arr,index_Following_arr));
Minimum_Separation_Dep_Dep(i,j) =
cell2mat(SeparationDeptDept(index_Leading_dep,index_Following_dep));
    end
end
dlmwrite([Output_dir, '/ Minimum_Separation_Arr_Arr_TBS.txt'], Minimum_Separation_Arr_Arr,
'newline', 'pc');
dlmwrite([Output_dir, '/ Minimum_Separation_Dep_Dep_TBS.txt'], Minimum_Separation_Dep_Dep,
'newline', 'pc');
end
```

ROT_limiting_Factor_analysis.m

```
% calculate the percent of ROT being the limiting factor to arrival
% capacity
for i = 1:length(Flag_indicator)
    a(i) = sum(Flag_indicator(i).Flag_ROT_limiting_Factor(:));
end

b = sum(a);

c = b/3990;

% find the percent of in-trail group of ROT being the limiting factor
for i = 1:length(Flag_indicator)
    for j = 1:399
        if Flag_indicator(i).Flag_ROT_limiting_Factor(j) == 1
fid = fopen('Violation_In_TrailGroup','a');
count = fprintf(fid,'%s\r',char(IntrailWakeGroup(i).Info(j)));
fclose(fid);
        end
    end
end

fid = fopen('/Users/junqihu/ATSL_Git/Wake_Analysis/Violation_In_TrailGroup'); %filePointer
% Violation_In_TrailGroup_txt_scan = [];
Violation_In_TrailGroup_txt_scan = textscan(fid,'%s');

Unique_intrail_groups = unique(Violation_In_TrailGroup_txt_scan{1,1});
Num_of_total_violatons = length(Violation_In_TrailGroup_txt_scan{1,1});
Num_of_intrail_groups = length(Unique_intrail_groups);

Num_of_violation_per_in_trail_group = zeros(Num_of_intrail_groups,1);
Percentage_violation_per_in_trail_group = zeros(Num_of_intrail_groups,1);
for i = 1:Num_of_intrail_groups
    a = ismember(Violation_In_TrailGroup_txt_scan{1,1},Unique_intrail_groups(i));
    Num_of_violation_per_in_trail_group(i) = sum(a);
    Percentage_violation_per_in_trail_group(i) =
Num_of_violation_per_in_trail_group(i)/Num_of_total_violatons;
end
```

BufferTimeGenerationDataBased.m

```
% scrpit file to include a data-based distribution of buffer time into
% Runsim model

function buffer_sec = BufferTimeGenerationDataBased(Airport_Altitude,ArrAircraft,k)
global pd_bd_Lower_than_2500_feet pd_db_Lower_than_2500_feet pd_dd_Lower_than_2500_feet
pd_dd_Over_2500_feet ...
    pd_de_Lower_than_2500_feet pd_de_Over_2500_feet pd_df_Lower_than_2500_feet
pd_df_Over_2500_feet pd_eb_Lower_than_2500_feet ...
    pd_ec_Lower_than_2500_feet pd_ed_Lower_than_2500_feet pd_ed_Over_2500_feet
pd_ee_Lower_than_2500_feet pd_ee_Over_2500_feet ...
    pd_ef_Lower_than_2500_feet  pd_ef_Over_2500_feet
if Airport_Altitude <= 2500
    if ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'B')
```

```
        buffer_sec                                                    =
random(pd_db_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'D')
        buffer_sec                                                    =
random(pd_dd_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'E')
        buffer_sec                                                    =
random(pd_de_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'F')
        buffer_sec                                                    =
random(pd_df_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'B')
        buffer_sec                                                    =
random(pd_eb_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'C')
        buffer_sec                                                    =
random(pd_ec_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'D')
        buffer_sec                                                    =
random(pd_ed_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'E')
        buffer_sec                                                    =
random(pd_ee_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'F')
        buffer_sec                                                    =
random(pd_ef_Lower_than_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'B') &&
ismember(ArrAircraft(k+1).WakeGroupName,'D')
        buffer_sec                                                    =
random(pd_bd_Lower_than_2500_feet,1,1);
    else
        buffer_sec                                                    = 19 + 13*randn; %
without large amount of data, we use the distribution of overall buffer times
    end
elseif Airport_Altitude >2500
    if ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'D')
        buffer_sec                                                    =
random(pd_dd_Over_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'E')
        buffer_sec                                                    =
random(pd_de_Over_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'D') &&
ismember(ArrAircraft(k+1).WakeGroupName,'F')
        buffer_sec                                                    =
random(pd_df_Over_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'D')
        buffer_sec                                                    =
random(pd_ed_Over_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'E')
        buffer_sec                                                    =
random(pd_ee_Over_2500_feet,1,1);
    elseif ismember(ArrAircraft(k).WakeGroupName,'E') &&
ismember(ArrAircraft(k+1).WakeGroupName,'F')
        buffer_sec                                                    =
random(pd_ef_Over_2500_feet,1,1);
    else
        buffer_sec                                                    = 27.66+ 14*randn;
    end
end
```

Wind_data_manipulation.m

```
Runway_Heading = 228;

% import the wind data
```

```
[~, ~, raw] = xlsread('/Users/junqihu/ATSL_Git/Wake_Analysis/Input
file/CLT_wind_2000_2014_excel.xlsx','CLT_wind_2010_2014.txt');
raw = raw(2:end,[4,7,end]);
raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {''};

% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non-numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

% Create output variable
data = reshape([raw{:}],size(raw));

% Allocate imported array to column variable names
HrMin = data(:,1);
Dirdeg = data(:,2);
Speedknots = data(:,3);

% Clear temporary variables
clearvars data raw R;
% Clear unreasonable results
Num_of_wind_data                                                            =
length(Speedknots);
for i = 1:Num_of_wind_data
    if Speedknots(i) > 100
        Speedknots(i) = NaN;
    end
end

%-------------------------------------------------------------------------
% calculate the time duration for each wind period
Time_duration = zeros (Num_of_wind_data,1);
for i = 2:Num_of_wind_data
    if HrMin(i) - HrMin(i-1) > 0
    Time_duration(i-1) = HrMin(i) - HrMin(i-1);
    else
    Time_duration(i-1) = HrMin(i) - HrMin(i-1) + 2400;
    end
end
Time_duration(Num_of_wind_data) = 2400 - HrMin(Num_of_wind_data);
% adding heading of the runway
Runway_Heading_deg = Runway_Heading*ones(Num_of_wind_data,1);
% calculating headwind
HeadWind = zeros(Num_of_wind_data,1);
for i = 1:Num_of_wind_data
if (Dirdeg(i) > 0 && Dirdeg(i) < 138) || (Dirdeg(i) > 318 && Dirdeg(i) <= 360)
    % (Dirdeg(i) > 0 && Dirdeg(i) < 86) || (Dirdeg(i) > 266 && Dirdeg(i) <= 360)
HeadWind(i) = abs(cos(deg2rad(Dirdeg(i) - Runway_Heading))* Speedknots(i));
else
HeadWind(i) = NaN;
end
end
% calculate mean headweind
Time_duration_times_Wind = zeros (Num_of_wind_data,1);
Time_duration(isnan(HeadWind)) = NaN;
for i = 11:Num_of_wind_data
    if isnan(HeadWind(i))
        Time_duration_times_Wind(i) = NaN;
    else
        Time_duration_times_Wind(i) = HeadWind(i)*Time_duration(i);
    end
end
Average_HeadWind_kt = nansum(Time_duration_times_Wind)/nansum(Time_duration);
% distribution of the HeadWind
figure
cdfplot(HeadWind);

Average_HeadWind = nanmean(HeadWind);
Median_Percentile_HeadWind_kt = prctile(HeadWind,50);
Sixty_Percentile_HeadWind_kt = prctile(HeadWind,60);
Seventy_Percentile_HeadWind_kt = prctile(HeadWind,70);
Eighty_Percentile_HeadWind_kt = prctile(HeadWind,80);
Ninety_Percentile_HeadWind_kt = prctile(HeadWind,90);
Ninety_Nine_Percentile_HeadWind_kt = prctile(HeadWind,99);
disp(Average_HeadWind_kt)
disp(Median_Percentile_HeadWind_kt)
disp(Sixty_Percentile_HeadWind_kt)
disp(Seventy_Percentile_HeadWind_kt)
disp(Eighty_Percentile_HeadWind_kt)
disp(Ninety_Percentile_HeadWind_kt)
disp(Ninety_Nine_Percentile_HeadWind_kt)
```