NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZATION OF CUBESAT GROUND STATIONS FOR INCREASED SATELLITE NUMBERS

by

Andrea J. Writt

June 2018

Thesis Advisor: James H. Newman
Co-Advisor: Giovanni Minelli

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2018	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE OPTIMIZATION OF CUBESAT GROUND STATIONS FOR INCREASED SATELLITE NUMBERS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Andrea J. Writt	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis contributes to the growing body of research concerned with the issue of the increasing numbers of government small satellites (SmallSats) and the limited number of ground stations available to support these missions. To understand this "many satellite, few ground station" problem, a Monte Carlo simulation is used to identify the point at which a single ground station is expected to be overwhelmed, specifically looking at the case of the Mobile CubeSat Command and Control (MC3) ground station at the Naval Postgraduate School (NPS). Ground station saturation is defined in terms of data downlink requirements and the increasing number of conflicting passes as the number of SmallSats grows. An assessment of when one ground station becomes insufficient for a growing number of SmallSats is the result. The MATLAB software tools created to generate these scenarios are generic and can be used to extend this work to investigate other scenarios of SmallSats and multiple ground stations.

14. SUBJECT TERMS small satellites, SmallSats, cube satellites, Cubesats, Monte Carlo simulation, Mobile CubeSat Command and Control, MC3, satellite operations center, SOC, pass conflicts, ground station optimization, ground station saturation	**15. NUMBER OF PAGES** 83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

OPTIMIZATION OF CUBESAT GROUND STATIONS FOR INCREASED SATELLITE NUMBERS

Andrea J. Writt
Captain, United States Marine Corps
BSME, University of Arkansas, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SPACE SYSTEMS OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
June 2018

Approved by: James H. Newman
Advisor

Giovanni Minelli
Co-Advisor

James H. Newman
Chair, Department of Space Systems Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis contributes to the growing body of research concerned with the issue of the increasing numbers of government small satellites (SmallSats) and the limited number of ground stations available to support these missions. To understand this "many satellite, few ground station" problem, a Monte Carlo simulation is used to identify the point at which a single ground station is expected to be overwhelmed, specifically looking at the case of the Mobile CubeSat Command and Control (MC3) ground station at the Naval Postgraduate School (NPS). Ground station saturation is defined in terms of data downlink requirements and the increasing number of conflicting passes as the number of SmallSats grows. An assessment of when one ground station becomes insufficient for a growing number of SmallSats is the result. The MATLAB software tools created to generate these scenarios are generic and can be used to extend this work to investigate other scenarios of SmallSats and multiple ground stations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AOS	acquisition of signal
APT	average pass time
COM	component objective model
CubeSat	cube satellite
DoD	Department of Defense
DR	data rate
EF	efficiency factor
GEO	geostationary Earth orbit
GUI	globally unique identifiers
LEO	low Earth orbit
LOS	loss of signal
MATLAB	Matrix Laboratory
MBSE	model based systems engineering
MC3	Mobile CubeSat Command and Control
NPS	Naval Postgraduate School
ODD	objective data download
OSF	objective saturation factor
RAAN	right ascension of the ascending node
RF	radio frequency
ROT	required objective time for data downlink
RTT	required threshold time for data downlink
SmallSat	small satellite
STK 10	Systems Tool Kit 10
TDD	threshold data download
TSF	threshold saturation factor

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor and co-advisor for their endless positivity, tireless encouragement, and ruthless support.

To my family, friends, peers, and small fuzzy creatures who have kept me in high spirits along the way, thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The population of small satellites in low Earth orbit (LEO) will continue growing over the next several years due to falling development costs and increased launch opportunities. The population increase may strain the communication capacity of these satellites and terrestrial stations that support them. For users monitoring and utilizing data from cube satellites (CubeSats) [1] and other very small satellites (SmallSats) [2] that can be launched *en masse*, ground station operations will become significantly more difficult as more satellites compete for limited resources. The innovative engineering focus has been largely devoted to designing and launching these new satellites, but the ground support systems have not yet received as much attention. This thesis examines how those systems will be affected as the number of satellites each station must track increases many-fold.

This increase in the CubeSat population quickly overwhelms the ground infrastructure supporting them. Accommodating larger numbers of satellites with few ground stations will consequently lead to conflicting contacts. The term *contact* is synonymous with the terms *access* or *pass* throughout this thesis. It is defined as the duration of time from the acquisition of signal (AOS) to the loss of signal (LOS) of a satellite communicating to its corresponding ground station. All of these contacts assume line-of-sight from the ground station to the satellite for radio frequency (RF) communications. Consequently, AOS/LOS often overlaps with the time spent above the local horizon. Satellites in a geostationary Earth orbit (GEO) would be an exception. This research focuses on LEO satellites, which have fast-moving, low-altitude orbits. The LEO regime will see the largest population increase of SmallSats.

It is assumed that each satellite has data to deliver to a ground station. A problem arises when there are multiple satellites in view of this ground station, as it can only service one satellite at a time. This creates a scenario where not every one of them may be able to downlink its full-required data for an extended period of time. An example

shown in Figure 1 depicts four satellites competing for ground station availability to relay data to the ground and accomplish their missions. This simulation is generated through a program called Systems Tool Kit (STK). Additionally, it is assumed that each ground station only has one antenna available that can only service one satellite at a time.

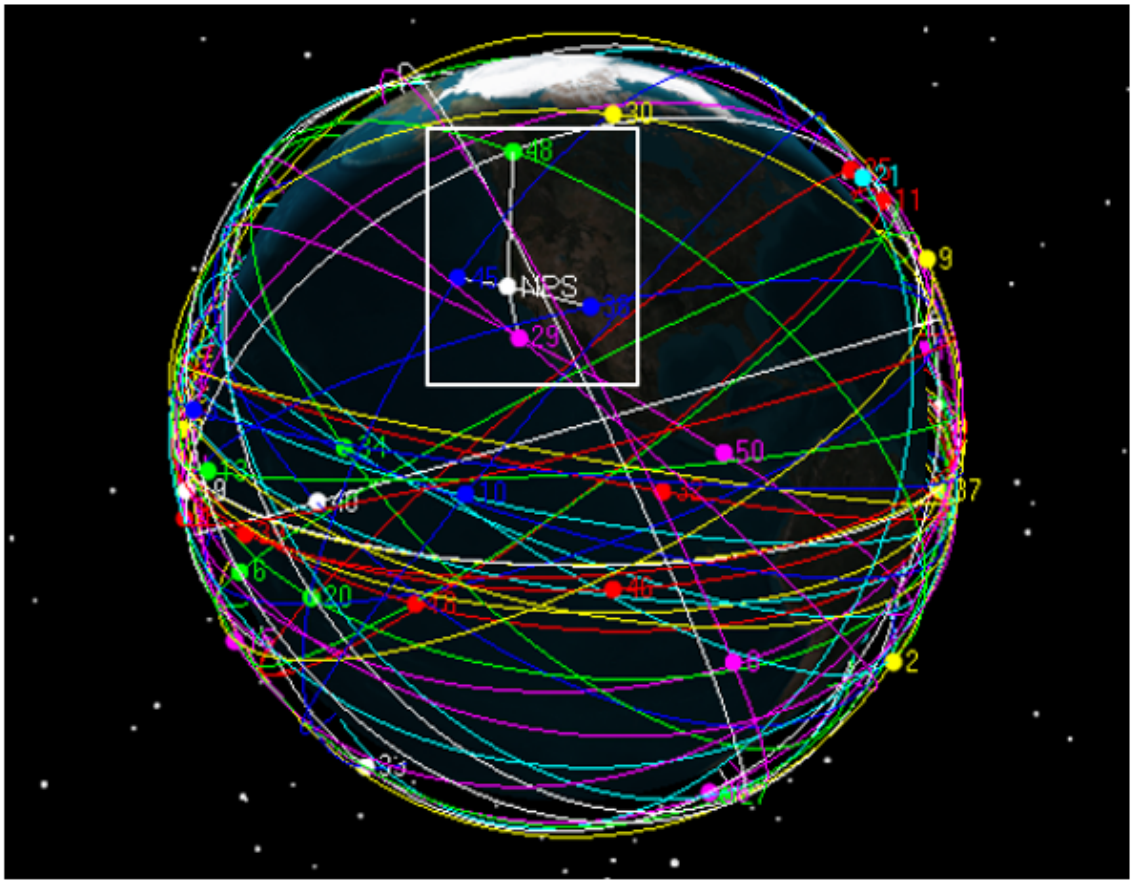

Figure 1. STK Simulation of Many Conflicting Satellite Contacts Over One Ground Station

Depending on how crowded LEO becomes, a single satellite may not have the opportunity to deliver its data for multiple consecutive orbits, causing it to be out of communication with the ground station for extended periods. Optimizing passes requires allotting adequate attention and services for each satellite, based on capability, data rates, and mission priority. Knowing the saturation point for a particular ground station network can help determine the approach to utilize for maximizing its capacity.

B. PURPOSE AND OBJECTIVE

This thesis determines the satellite population quantity likely to overwhelm a ground station network, degrading it to a point where it is no longer able to guarantee minimum dedicated communications passes to the satellites it is responsible for tracking. Specifically, the intent of this thesis is to define what saturation looks like for a single ground station. The process selected to conduct this analysis is a *Monte Carlo* method that is general enough to be applied to multiple orbital constructs. A Monte Carlo algorithm runs a simulation repeatedly, with random values, to demonstrate the wide array of possible answers with minimal amounts of risk involved. The specifics of the program will be discussed in further detail in Chapter III. Knowing when a ground station is likely to saturate should help civil, commercial, and Department of Defense (DoD) organizations be able to project the need for optimization of existing ground stations and procurement of new ground stations.

C. LITERATURE REVIEW

Government, the commercial sector, and academia all agree on the issues posed for the LEO orbital regime in the near future. The problem of excessive satellites per ground station has become a topic of discussion among operators and planners within this growing field of small satellite users. Aspects of this situation to include algorithms, mission architecture design, and scheduling solutions have been investigated, but much of the issue still remains unsolved.

Acknowledged in the joint university study on "Emergent Trends for CubeSat Ground Systems" [3], the one-to-one correlation of a single satellite for a single ground station will quickly disappear as we learn to juggle the complications of multiple satellites operating on the same frequencies and communicating with the same ground station. This work identifies commonly noted trouble areas and analyzes where machine learning algorithms and further development of the architectures could be employed to improve operations. Continuing work to expand upon the generic framework will advance integration of new programs within the architecture to contribute to the

development of common applications, similar to the ones we use daily on our phones, seamlessly handling larger numbers of satellites.

The emerging technology of model based systems engineering (MBSE) is also a new, useful way to look at the entire design trade space through multiple analysis tools, design methods, optimization and verification of the systems [4]. Partners from the academic, commercial, and government sectors are working together to create frameworks to improve the mission design and operations of an individual satellite and its individual ground station. A program that is commonly used in this field simulates satellites' orbits and their communication with supporting ground networks. Through this tool and its interface with the aforementioned program, STK, specific extraction of a portion of the system can be analyzed. STK is utilized as the orbital propagator while MBSE helps to address the mission-specific design and parameters. This analysis is then integrated back into the scenario to improve overall mission effectiveness.

Separately, Dr. Sara Spangelo investigated the common constraints experienced by ground stations in her dissertation by identifying aspects such as mass, size, volume, power and funding levels of these small satellite missions [5]. Through the dynamically varying durations of CubeSat missions and their correlated ground stations, average access time of the network is calculated. Her work created a scheduling formulation to maximize satellites' data relay to the ground, based on current mission parameters from existing small satellites with varying downlink requirements.

Finally, the body of research being conducted at the Naval Postgraduate School, related to the *many satellite, few ground station* problem, aims to contribute by framing solutions to reconcile some of the known operating constraints. Doctoral candidate Mr. Giovanni Minelli's work [6] addresses the collection of aspects that define this optimization problem and creates a formulation to best communicate with multiple satellites concurrently in view of a ground station. His work inspired the basis for the question this thesis is answering, as well as the work of another graduate student, Major John Leone III. Leone's thesis contains an in-depth analysis on how to quantitatively assess PicoSats Realizing Orbital Propagation Calibrations Using Beacon Emitters (PropCube) satellite passes as a function of initial azimuth and time during the pass based

on historical pass data [7]. Leone identifies which satellite accesses provide the most data downlink, and Minelli's shows how to efficiently slew between the satellites in view. This thesis characterizes the missing piece of determining when ground station saturation occurs and optimization of passes needs to be considered.

The concerns outlined by each author are not unique to their separate locations, sponsors, or missions, but rather pieces of a larger puzzle. Technologists worldwide aim to solve this puzzle over the next few years as these issues are amplified by the increasing numbers of small satellites used to perform novel scientific and technological research. Together these efforts will help identify and mitigate the *many satellite, few ground station* problem as the number of small satellites continues to increase for research and operational purposes.

D. SUMMARY

Chapter I has discussed the issue of the increasing ratio of CubeSats to ground stations, which should directly correlate to an increase in overlapping downlink contacts these satellites have with the ground station. First, it provided the context for this concern. It then described the purpose and intent of this thesis, which is to identify the point at which increased numbers of satellites overwhelm a ground site. Finally, this chapter covered the studies conducted by other researchers pertaining to the growing interest in small satellites, which support and enable this work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SCOPING THE PROBLEM

This chapter describes in more detail the formulation of the assumptions that the Monte Carlo simulation is based upon, and explores the preliminary thoughts of what factors contribute to saturation. It provides the foundational concepts to understand the methodology in the following chapters.

A. INITIAL CONSIDERATIONS

At first, the simulation of increasing numbers of satellites and determining their behavior with respect to one ground station seemed rather straightforward. As the investigation of the problem progressed however, many more variables presented their impact on the issue of a ground station reaching its maximum capacity. Therefore, a versatile program that can simulate a simple scenario initially, and build to more complex models, was the best fit. The goal was to identify the range of where there is a slight degradation, a more obvious degradation, and complete saturation to the ground station capability, where it is impossible to meet mission objectives. Reference satellites are utilized to look at the threshold and objective time-in-view of a ground station needed for the satellite to complete its mission. Looking at the amount of time it takes for each satellite to conduct a data transfer to a ground station, is there enough time for the task to be completed before another satellite needs to conduct its data transfer?

Based on existing research, original simulation cases provide the data to perform this analysis. Through studying comparable problems, their methods, and results, this thesis intends to provide a solution for this specific facet of ground station network optimization. A series of test scenarios with differing time periods collects the following data shown in this report. Trends in scenarios become obvious through the numerous iterations of this method. Giovanni Minelli, a researcher at the Naval Postgraduate School (NPS) SmallSat lab, has been investigating this problem for the past few years and produced the basis on which this thesis is written. The Matrix Laboratory (MATLAB) scripts are his original work and create the foundation for the master Monte Carlo code conducting these simulations. The in-depth description of the benefits of the Monte Carlo

simulation, are explained in the Methodology and Approach section in Chapter III. Each situation is conducted for a single facility, the NPS Mobile CubeSat Command and Control (MC3) ground station, in Monterey, California. The different cases observed start simple and grow more complex as the number of satellites is increased. Scenarios with satellites in random orbits are modeled, but these issues are not unique to random, disaggregated, individual satellites. The issues are shared with satellites communication with one or many ground stations in diverse latitudes.

To create a scenario with random satellites, all of the orbital elements of each satellite needed to be taken into consideration. Evaluating which parameters provide the most realistically generated orbits leads to the conclusion that all of the six main orbital elements had a significant influence on the outcome of the program. Figure 2 displays these classical Keplerian orbital elements [8]. The six orbital parameters that were varied include the eccentricity, semimajor axis, inclination, right ascension of the ascending node (RAAN), argument of perigee, and the mean anomaly of each satellite's orbit.

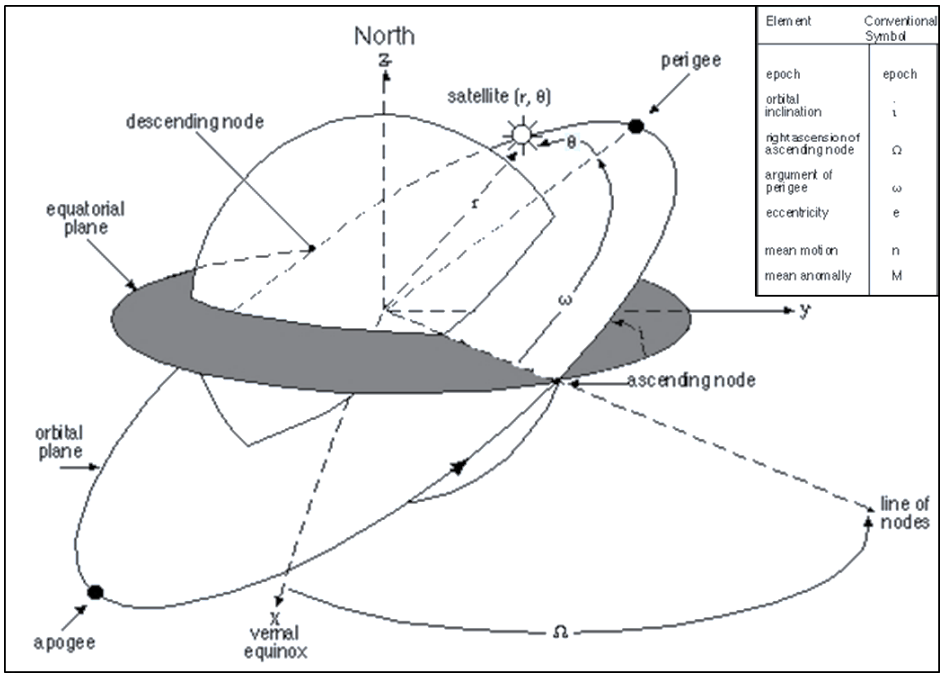

Figure 2. Classical Orbital Elements. Source: [8].

The randomization of these variables was conducted through using a preprogrammed function in MATLAB, known as the *rand* function. This function randomly selects a uniformly distributed number between one and zero. Although the regularity of the *uniform* distribution removes some of the randomness from the scenario, it still provided the best fit for the problem. The rand function was used extensively to create arbitrary numbers for each of the orbital parameters, within a certain range, for each individual satellite created in the random scenarios.

As more time was invested in creating the script in MATLAB, which controls the scenario in STK 10, a multitude of different factors surfaced that begged for further consideration. It became clear that collection of data for different time intervals such as one day, one week, one month, and one year produced very diverse results. The number of satellites naturally had an impact on the number of conflicts recorded, as well as the number of iterations performed by the Monte Carlo loop. Once selections were made for which elements should remain constant, noticeable trends began emerging in the collected data.

B. ASSUMPTIONS

This section describes the assumptions used to provide a baseline for the scenarios. As other considerations were accounted for, these assumptions were adjusted throughout the simulations and are noted in greater detail in the test scenarios, contained in Chapter IV. Figure 3 depicts the altitude of the perigees for all recorded CubeSat missions from the year 2000 through present day [9]. Based on this data, the LEO altitude window that was selected for the purpose of this thesis was between 400 and 700 kilometers, since the vast majority of small satellite missions fall within that range.

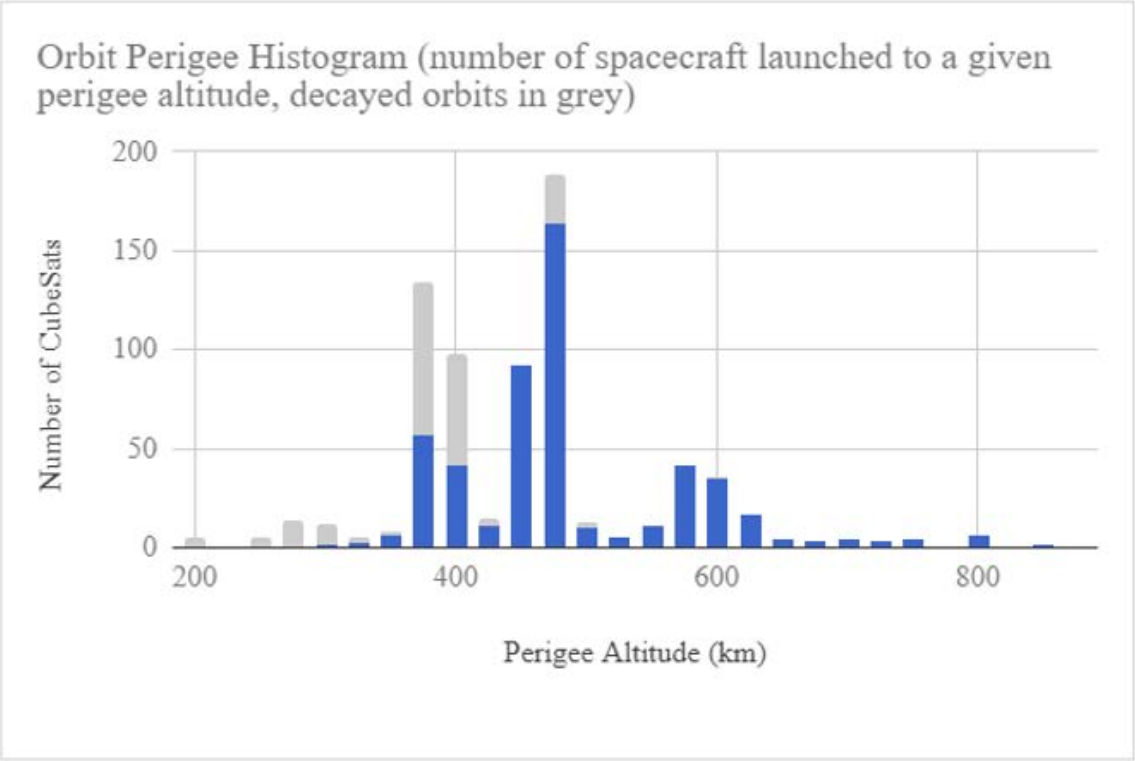

Figure 3. Orbital Altitudes of CubeSat Missions Since 2000.
Source: [9].

Eccentricity is calculated based on the perigee and apogee of an orbit, which are the nearest and farthest points from Earth, respectively. With the set minimum and maximum altitudes noted above, the most extreme eccentricity an orbit could have is 0.02. When the randomizing function from MATLAB is applied, orbits will be constructed to have an eccentricity between 0.02 and zero, where an orbit with a zero eccentricity is also referred to as a *circular orbit*.

Early in the scenario generation, another assumption was the number selected for the maximum inclination of the orbits. The orbital inclination was initially set to 70 degrees. This was based on the fact that the majority of the world population lives between 70°N and 70°S latitude [10]; therefore, a ground station monitoring satellite missions would have a greater chance of being placed in a populated region. Later in the course of the research, the inclination was modified to 98 degrees, to account for polar and sun-synchronous orbits as well.

Communication only happens when the satellite is within line-of-sight, above the local horizon. Once the spacecraft is above the horizon, due to geographical considerations and terrestrial noise, it is reasonable to mask the elevation angle at 10 degrees. Within the scenario, the elevation angle for the simulated ground station is manually set at 10 degrees and saved in the base scenario. A pass begins when the satellite is at 10 degrees above the horizon at AOS, it then proceeds through its maximum elevation of the pass and concludes at LOS, 10 degrees above the local horizon. The duration between when the satellite rises and sets at above 10 degrees is considered *overhead time*. Figure 4 provides a representation that clearly shows the elevation angle selected for the ground station.

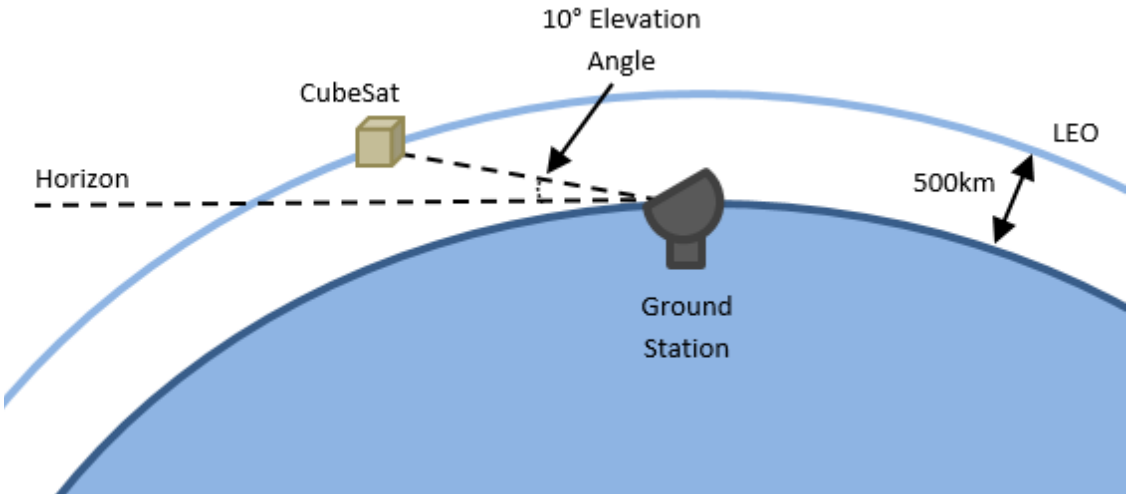

Figure 4. Depiction of 10 Degree Elevation Angle

C. DEFINING SATURATION

The complexity of the term saturation became apparent as the test cases began to provide useful data. To gain a better understanding of the depth of the issue, several facets called for further investigation. First, it has been determined that a single number or point cannot define saturation, but rather an average factor can be calculated to assess the range in which saturation occurs. Larger numbers of satellites provide an obvious correlated increase in conflicting accesses, but an interesting characteristic was the

number of satellites that had zero accesses to the ground station for the entire scenario. With the random orbit generator, a percentage of the satellites were put into orbits with an inclination that would never pass within ten degrees elevation, line of sight of the NPS ground station at 36.6°N, 121.9°W [11]. It was found that an average of approximately 23% of all randomly generated satellites for the 1000 satellite case did not have orbital parameters permitting radio contact. The 500 satellite and 100 satellite cases fell within one percent point of this value as well. Satellite scenarios discussed in this thesis will be referred to by their simulated number of *N satellites*, but to provide meaningful results it is important to note that these case will contain *N*(1-.23)* satellites that have at least some contact time with the NPS ground station. Therefore, it is reasonable to conclude that each *N satellite* case is actually a *.77N satellite* case and each "N satellite" scenario will be placed in quotations to indicate that it is describing *N random satellites, but only .77N satellites that have contact with the ground station*. This factor is discussed in further detail in Chapter IV. Placing a ground station at a different latitude, would clearly yield different results. Saturation of the station is caused by only those satellites whose orbital geometry enables contact. When determining where to place a ground station, operators are not typically looking at a random number of "N satellites," but this factor may be useful to know how many randomly distributed satellites a ground station could service based on its latitude.

The next aspect of saturation worth considering involved the competing interests of the conflicting satellites. All satellite owners and operators want the full access time they were assured they would receive to conduct their missions, when overhead the ground station. The term this thesis uses to describe this concept is *required threshold time*. Required threshold time should be smaller than the previously mentioned *overhead time*; if not, satellites would rarely, if ever, meet their threshold data requirements. As the term hints, required threshold time is the absolute minimum time a satellite needs to successfully complete its mission, in this case over the duration of one week. Therefore, the required threshold time for a single pass is a fraction of that weekly requirement and is dependent on the number of passes the satellite will make in that timeframe. Figure 5

provides a clearer depiction of the relation between overhead time and required threshold time.

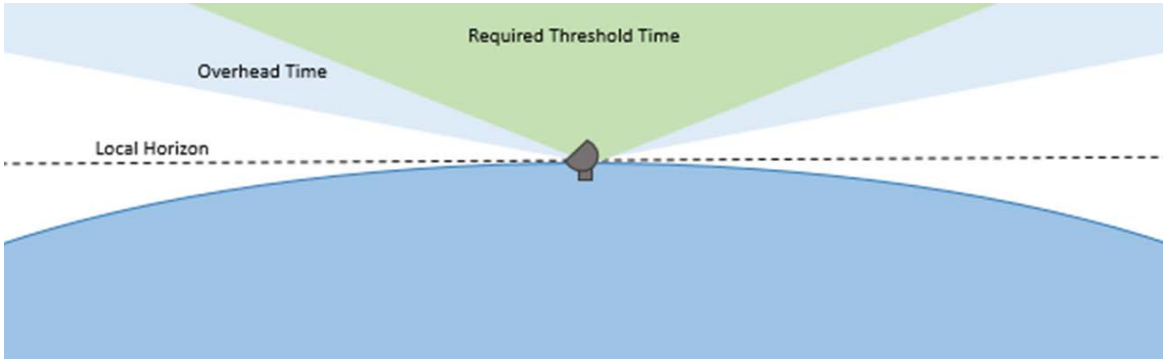

Figure 5. Depiction of Overhead Time Vs. Mission Required Threshold Time

To frame this idea, consider that when there is one satellite overhead, that satellite has more than 100% of the time it requires to communicate with the ground. If there are two identical satellites overhead simultaneously, the required threshold time is now doubled, because each satellite still needs its full required time, but the ground station can only communicate with one at a time. Therefore, the ground is only providing half of the overhead time for each satellite, and that may be less than that required by each of the two satellites. This, of course, continues as the number of conflicting satellites increases.

Lastly, to analyze the concept of saturation, low, medium, and high complexity mission parameters were used to evaluate what a minimum required threshold time looks like for varying CubeSat missions. Reference satellites were selected to give real-world examples. Table 1 displays the satellites utilized to provide the baseline requirements and an initial look at the required threshold rates. The data rates mirror current CubeSat missions and the minimum and maximum data values were approximated based on the size of the products each satellite type is downloading. The one-week period is used as a benchmark to assess the absolute minimum amount of data needed within that period to be considered a mission success. This timeframe was selected because a one-day analysis period does not provide enough contact time for some of the lower complexity missions

to achieve the possibility of adequate data attainment. One week provided enough data to average out the daily variations in the pass schedule due to the differing orbital geometries. From a Monte Carlo scenario of "100 random satellites" performed for 10 iterations at the NPS MC3 ground station, the average number of times each satellite passes over the ground station was measured at 3.09 times per day or 21.65 per week. On average, a satellite pass lasted 394.9 seconds. The intent of displaying the data in this manner is to provide the requirements in tangible terms for mission planning. The low-complexity mission is based on known historical data from one of the NPS PropCubes. The medium and high-complexity missions are modeled after expected data usage for geolocation missions and CubeSats providing high-definition streaming-video capabilities.

Table 1. Reference Missions Data Downlink Requirements for One Week. Sources: [7], [12], [13], [14], [15], [16].

Mission Complexity	Low	Medium	High
Satellite Name/Mission	Ex: PropCube-Fauna	Ex: RF Geolocation	Ex: CubeSat w/ Streaming Video
Threshold Data Download (TDD) (bytes per week)	5 KB	50 MB	50 GB
Objective Data Download (ODD) (bytes per week)	12.5 KB	200 MB	200 GB
Data Rate (DR) (bits per second)	9.6 kbps	1 Mbps	100 Mbps
Average Number of Passes (per week)	21.65		

For each case, the threshold data download (TDD) and objective data download (ODD) requirements were established. From the data rate capability, a minimum amount of downlink time, per week, was established. If the simulation shows that the average satellite does not receive its minimum required time, then the saturation point has been reached. This problem becomes even more convoluted when higher complexity missions, which have the ability to vary their data rate, are taken into account. Future work will need to address aspects such as high complexity satellites with complex downlink

capabilities. Such missions can transfer data in a more efficient manner by varying their data rates throughout a pass, depending on the link margin. However, variable data downlink rates are not addressed in this thesis.

Table 2 uses the information from Table 1 to scale the data requirements from one-week, down to the threshold and objective data requirements per pass. A satellite efficiency factor is utilized to calculate the amount of time it will take for a satellite to downlink its minimum data. These efficiency factors differ based on the knowledge that higher data rate communications systems are usually more sophisticated, enabling such systems to use more of each pass over the ground station. Looking at the data rates and the time required to achieve them, the required threshold time (RTT) for data downlink and required objective time (ROT) for data downlink, provide values to calculate the saturation factors for each mission complexity.

Table 2. Pass Requirements for Data Downlink Capacity. Sources: [7], [12], [13], 14], [15], [16].

Mission Complexity	Low	Medium	High
Satellite Name/Mission	Ex: PropCube-Fauna	Ex: RF Geolocation	Ex: CubeSat w/ Streaming Video
Threshold Data Download (TDD) (bits per pass)	1.27 kb	12.70 Mb	12.70 Gb
Objective Data Download (ODD) (bits per pass)	3.18 kb	50.79 Mb	50.79 Gb
Data Rate (DR) (bits per sec)	9.6 kbps	1 Mbps	100 Mbps
Efficiency Factor (EF)	0.001	0.3	0.5
Average Pass Time (APT) (sec)	394.9	394.9	394.9
Required Threshold Time (RTT) for Data Downlink (sec per pass)	132.2	42.33	254
Required Objective Time (ROT) for Data Downlink (sec per pass)	331.2	169.3	1015
Threshold Saturation Factor (TSF)	0.33	0.11	0.64
Objective Saturation Factor (OSF)	0.84	0.43	2.57

The following equations were used to calculate the numbers in Table 2. Equations (1) and (2) were used to calculate the numbers for the RTT and ROT, based on the values in Table 2 for the mission complexity. This equation is based solely on the mission requirements and is not related to the orbital geometry.

$$RTT = \frac{TDD}{DR\ x\ EF} \quad (1) \qquad ROT = \frac{ODD}{DR\ x\ EF} \quad (2)$$

The third (3) and fourth (4) equations display the method used to produce the saturation factors for the threshold (TSF) and objective (OSF), which incorporates the mission requirements and orbital geometry. The *threshold saturation factor* (TSF) is the percentage of the 6.6-minute pass (on average) that a single satellite needs to be in contact with the ground station to achieve its mission. The *objective saturation factor* (OSF) is the percentage of the average pass needed to achieve the objective data download. Table 2 shows these saturation factors.

$$TSF = \frac{RTT}{APT} \quad (3) \qquad OSF = \frac{ROT}{APT} \quad (4)$$

A saturation factor of less than 1.0 says that the data download goal, either threshold or objective, can be accomplished for a single satellite of that complexity using that ground station. So, the PropCube and the Geolocation mission examples have data download margin for both threshold and objective downloads for at least one satellite, whereas the streaming video satellite can achieve its threshold for one satellite, but not its objective data downloads. When a satellite has data download margin, it is clear that more satellites of that complexity can be supported before the system becomes saturated. Therefore, saturation is defined as occurring when the number of average satellites increases to the point where the average satellite can no longer receive its threshold data download. The Monte Carlo simulations are used to determine this quantity of satellites.

D. SUMMARY

The consideration of *overhead time* and *required threshold time* with respect to the selected reference satellites are each elements of determining ground station saturation. These aspects begin to create a clearer understanding of the need for optimization, providing the foundation of the problem this thesis addresses. This chapter provided the initial considerations of what at first may have seemed a simple, one-to-one correlation as the satellite-to-ground-station ratio grows. It described the early assumptions of the stated problem, and the relevant variables based on these assumptions. In summary, Chapter II defined factors that contribute to the definition of saturation.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY AND APPROACH

This section explores the logic behind the programming, scenario selection, and data collection. Before fully automating the program, manually inputted test cases were run. These initial scenarios were used to assess the most appropriate time period, number of iterations, and different-sized cases with varying satellite quantities, to accurately display meaningful data. Through these procedures, the hope was to be able to identify when the communications capability went from being slightly degraded to completely overwhelmed.

A. SCENARIO DURATIONS

The data collected for scenarios lasting one day and one week did not provide enough contacts from the satellites to the ground station to allow for a statistically consistent depiction of trends over time. However, at the one-month timeframe, the data collected behaved in a more repeatable manner. Further tests were conducted at three months and longer timeframes. The very slight deviations shown in these longer time periods showed that statistical significance was achieved for scenarios run for one month. The single-month scenario provided the information needed to begin observing trends. For this reason, the scenario times for the bulk of the cases discussed in this thesis all lasted one month to reduce the necessary processing time, but still provide enough data for accurate analysis. Figure 6 shows two plots, where the graph on the top depicts a scenario with a time interval of one day. The plot on the bottom shows the same scenario with a time interval of one week. The following two plots in Figure 7 show that for the same scenario shown in Figure 6, the plotted points appear to show less deviation between the one-month scenario time on the top and the three-month scenario on the bottom. Therefore, the selected timeframe is outlined red.

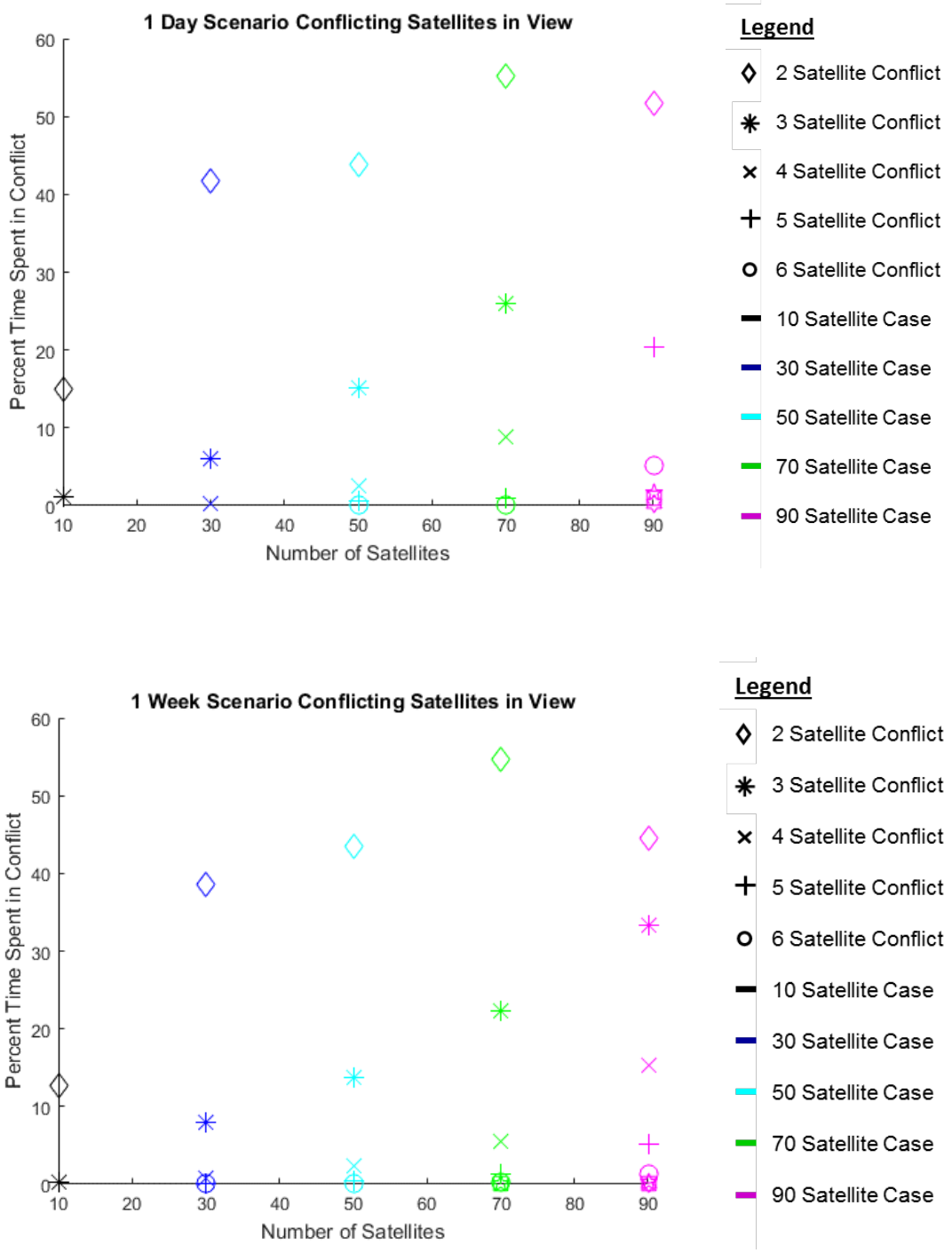

Figure 6. Comparison of Same Scenarios with Different Time Intervals: 1 Day (Top) Versus 1 Week (Bottom)

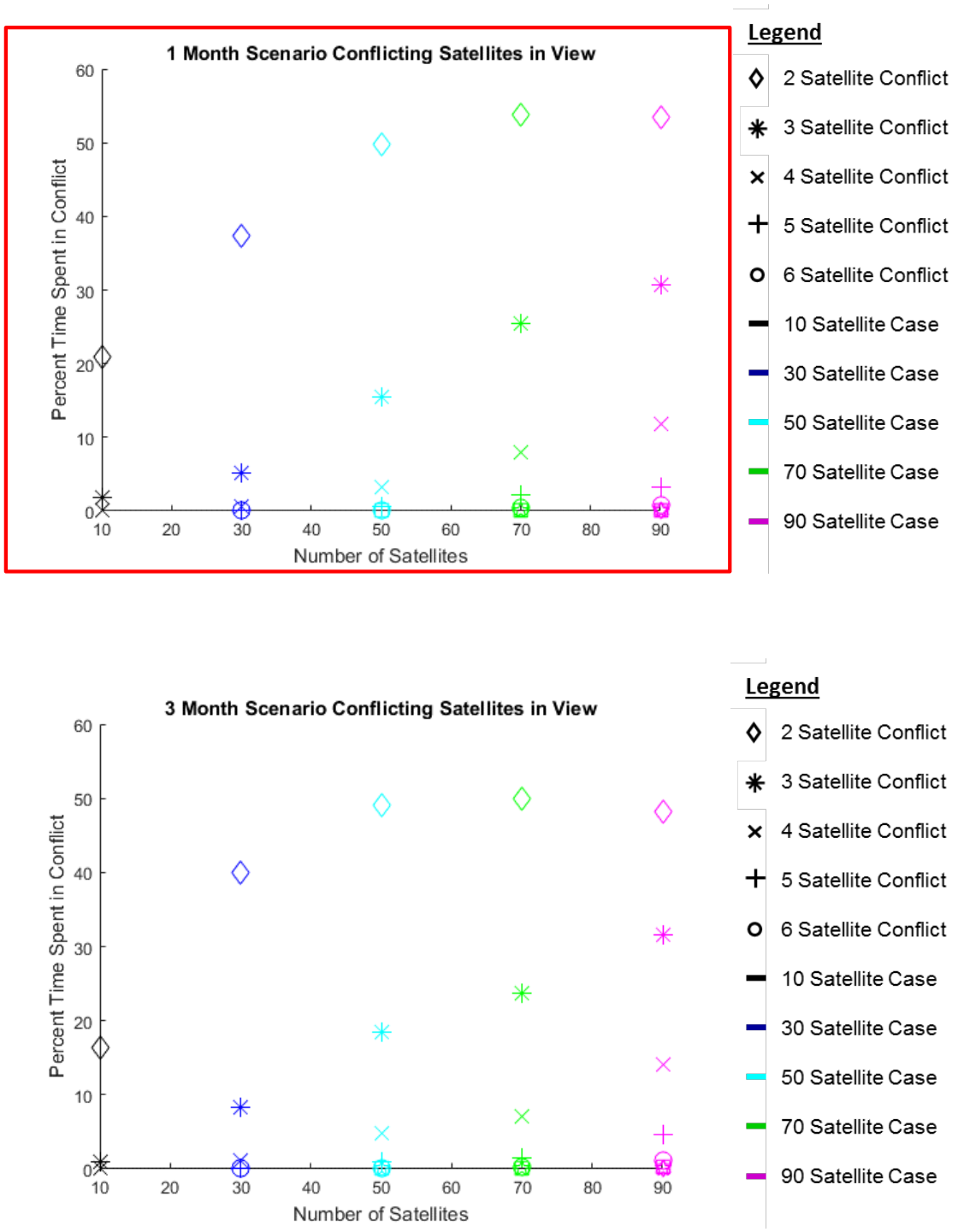

Figure 7. Comparison of Same Scenarios with Different Time Intervals: 1 Month (Top) Versus 3 Months (Bottom)

A Monte Carlo algorithm of repeated random samplings generated these simulations, which created the increasing number of satellites by varying orbital parameters. This algorithm was implemented in MATLAB, which interfaces with and opens a new scenario in STK, calculating the accesses of each satellite to the ground station and relaying this data back to MATLAB to then create files that store the data chronologically. The manipulation of the data for numerical and graphical analysis was then conducted in MATLAB. The following sections describe how the two programs interact and the user inputs required to vary parameters within the simulations.

B. MATLAB-STK INTERFACE

Before the user initiates the main code titled "Monte_Carlo_Loop.m," included in the appendix, there are a few variables that must be selected to create a scenario. The script provides a generic platform to run *N iterations of Y satellites*, where the operator can select both variables, *N* is the number of iterations, and *Y* is the number of random satellites. Figure 8 displays the variables that must be set, as they appear in the MATLAB script. The yellow highlighted portions show the values that the user must input and the green text describes their purpose.

```
MAXSATS = 100;                          % Set number of randomly generated satellites
MAX_ITERATIONS = 15;                    % Set number of iterations for Monte Carlo Loop to run
SAT_STEP = 10;                          % Set step size to increment number of sats to increase
altlow = 400000;                        % Set minimum orbital altitude in m
althigh = 700000;                       % Set maximum orbital altitude in m
max_incl = 98;                          % Set maximum inclination in degrees
start_time = '19 Jul 2017 02:00:00';    % Set scenario start time
end_time = '19 Aug 2017 02:00:0';       % Set scenario end time
GS = [{'NPS'}];                         % Set ground station to be observed
```

Figure 8. Monte Carlo Loop Input Variables

The first portion of the code allows the operator to select the maximum number of satellites to be created for the scenario. The step size, which is the next input value, determines the increments at which the satellite numbers grow within the scenario. For example, if the step size is 10, with a maximum number of "100 satellites," the program

will create 10 satellites, followed by 20 new satellites, followed by 30, and so forth, until it has reached the desired maximum. After this, the number of iterations of the Monte Carlo loop is then set.

The next values the user must input are the minimum and maximum altitude, as well as the maximum inclination desired for the creation of the randomly generated orbits. The next three lines in the program allow the user to determine the desired start and stop time for the scenario, as well as specify the name of the ground station that will be observed. The script then allows the operator to type the desired location for the data files to be stored on the computer.

At this point, the MATLAB script directly opens STK through a component object model (COM) interface [17]. The commands used in the MATLAB code operate as function calls that emulate button clicks in the graphical user interface (GUI), which configures and runs the propagation software that drives STK. Code snippets are provided to assist users with connecting and customizing through the programming interface. Figure 9 shows an example of the MATLAB code samples provided online to open STK, create a scenario, and define properties within the scenario. If there is a separate, previously-saved scenario file the user desires to open and populate, instead of creating a new one, that modification that can be done at this point in the program as well.

```
filepath = '\\special\ssagcommon$\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization Threshold)\Results\';
folderpath = strcat(filepath,int2str(MAX_ITERATIONS), 'Iterations_', char(datetime('now','TimeZone','local','Format','y-M-d_HH-mm-ss\')));
mkdir(folderpath);

disp('Starting STK10')
app = actxserver('STK10.application');
root = app.Personality2;
root.Children.Import('\\special\ssagcommon$\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization Threshold)\Scenario\Monte_Carlo_MC3.sc');
sc = root.CurrentScenario;
station = [root.GetObjectFromPath('/Facility/NPS')];
sc.SetTimePeriod(start_time, end_time);
root.ExecuteCommand('Animate * Reset');


for NUMSATS = 10:SAT_STEP:MAXSATS
    fprintf('starting: NUMSATS = %d at %s\n',NUMSATS, char(datetime('now')))
    for iteration = 1:MAX_ITERATIONS
        fprintf('Starting iteration: %d at %s\n', iteration, char(datetime('now')));
        Real_Monte_Carlo_Random(iteration, NUMSATS, altlow, althigh, max_incl, start_time, end_time, GS, app, root, sc, station, MAX_ITERATIONS, folderpath);
    end
end

T2 = now;
Runtime = T2 - T1;                  % In days
Run_min  = (Runtime * 86400) / 60;  % In minutes
Run_hrs  = Run_min / 60;
fprintf('Program ran for %6.2f minutes, or %6.2f hours, or %6.2f days\n',Run_min, Run_hrs, Run_hrs/24)

clear app                           % closes STK 10
```

Figure 9. MATLAB to STK Scenario Creation Code Snippet

The main file script now calls on two subroutine files known as functions. These files require no inputs from the user, but are solely there to support the overarching Monte Carlo Loop script. The first file that is called upon is the "Random_Satellite_Generator.m," contained in the appendix. This script contains all of the detailed commands to create the satellites in STK for the specified number of iterations. Once the program has stored the data for each satellite's pass over the ground station, it deletes all of the satellites to make room for the next iteration of random SmallSats. The data collected and stored contains the AOS, LOS, duration of each pass, the satellite's assigned number for tracking purposes, and the name of the ground station. This script also displays the number of satellites for an entire scenario, whose random orbital elements never allowed for the spacecraft to pass over the ground station at all. This output is called the *number of misses*.

The second subroutine file, additionally provided in the appendix, is called the "Conflict_Compiler.m." This is where the program compares the AOS and LOS from one satellite to the AOS and LOS of another to look for overlapping time periods. The term used for each period of time where there is at least one satellite in view is a *slice*. These *slices* are separated by the time when there are no satellites in view, and each *slice* contains anywhere from one satellite all the way up to dozens of satellites all in view of

the ground station at the same time. This program then prints a few lines of text, the first describing the total duration with *at least* one satellite in view. That output is followed by a note that displays the duration in time, as well as percentage of total time, that there is one satellite in view, two satellites in view, three, four, and so forth. This data is then stored as a *.mat* file through MATLAB, and able to be accessed for later analysis.

In Figure 10, the access report shows a week's worth of passes for a simulation run with "20 random satellites." Of the twenty satellites, fourteen satellites had passes over the NPS ground station and Figure 10 shows that many conflicts can be seen for these satellites. Figure 11 gives a closer look, showing a segment of time over the one-week scenario period. The images depict the overlap of the satellites simultaneously in view of the ground station. The box outlined in red focuses on the overlapping durations of the satellite passes, for one set of conflicted passes.

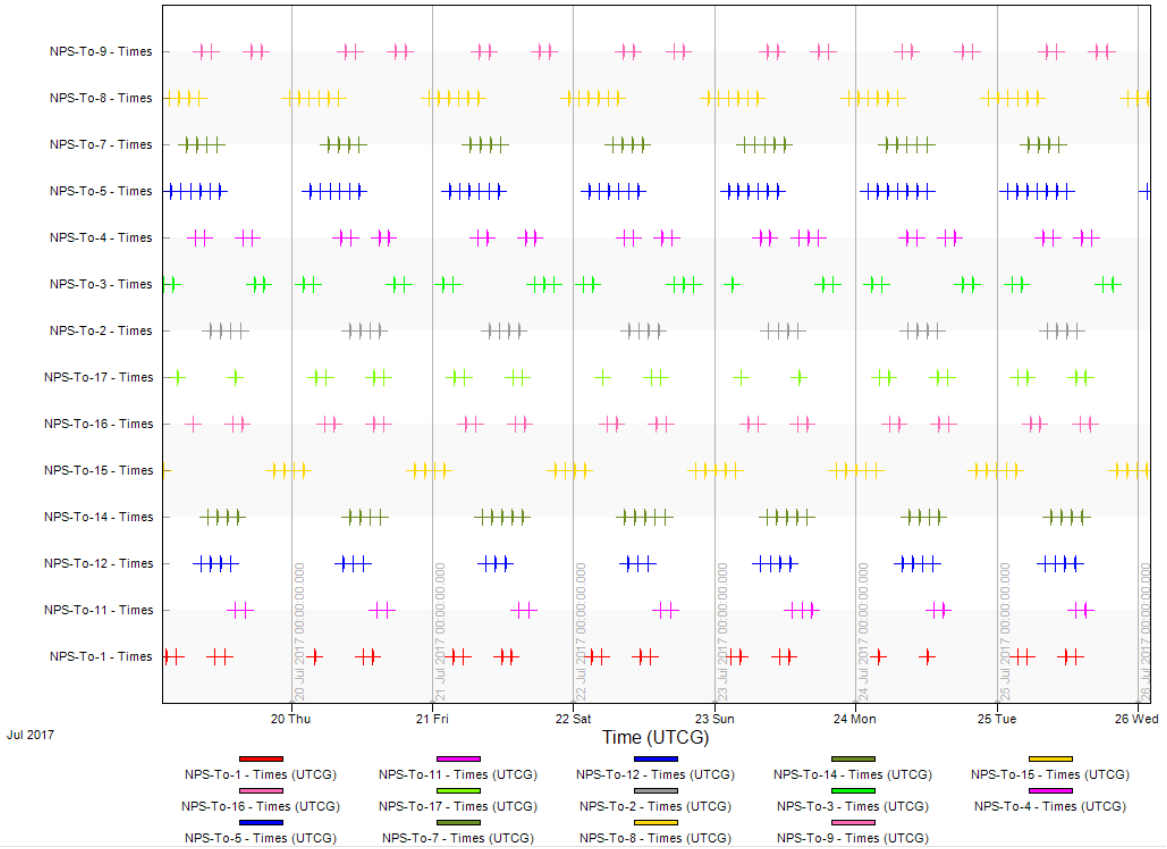

Figure 10. STK Access Report for 1 Week Scenario of "20 Random Satellites"

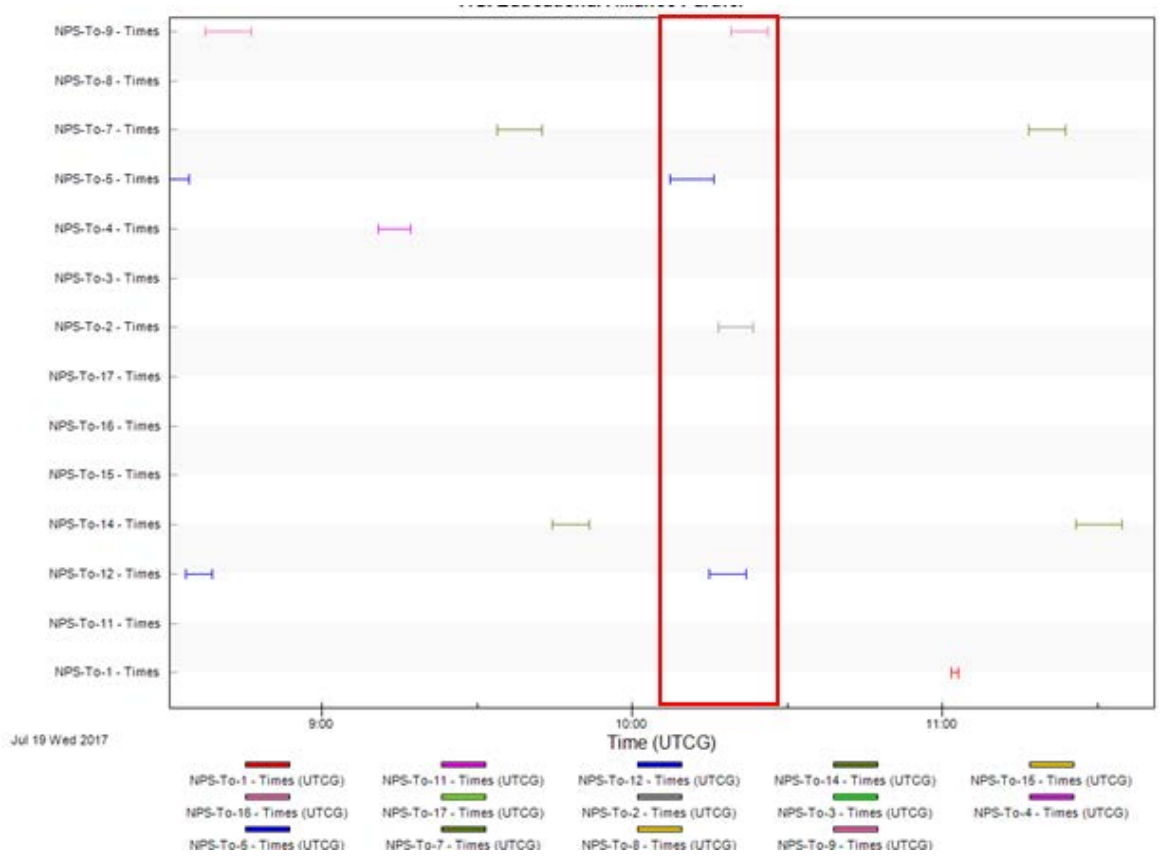

Figure 11. Slice of Overlapping Accesses in STK

Figure 12 further displays the breakdown of the concept of these slices. It shows three individual examples of slices of intersecting CubeSat access times with the ground station. The *Conflict Compiler* script accounts for every new AOS by adding a *one* to the number of conflicting satellites in that *slice* and subtracts the number *one*, at the next chronological LOS, from the next *slice*. The first slice in the figure depicts two satellites with overlapping durations. The second and third slices display increasingly more complex scenarios, where conflicting accesses begin to pile up, ultimately creating more and more scheduling issues. The *Conflict Compiler* script is designed so as not to double count conflicts by accounting for two or more *satellites-in-view* separately and not including one *satellite-in-view* in that number.

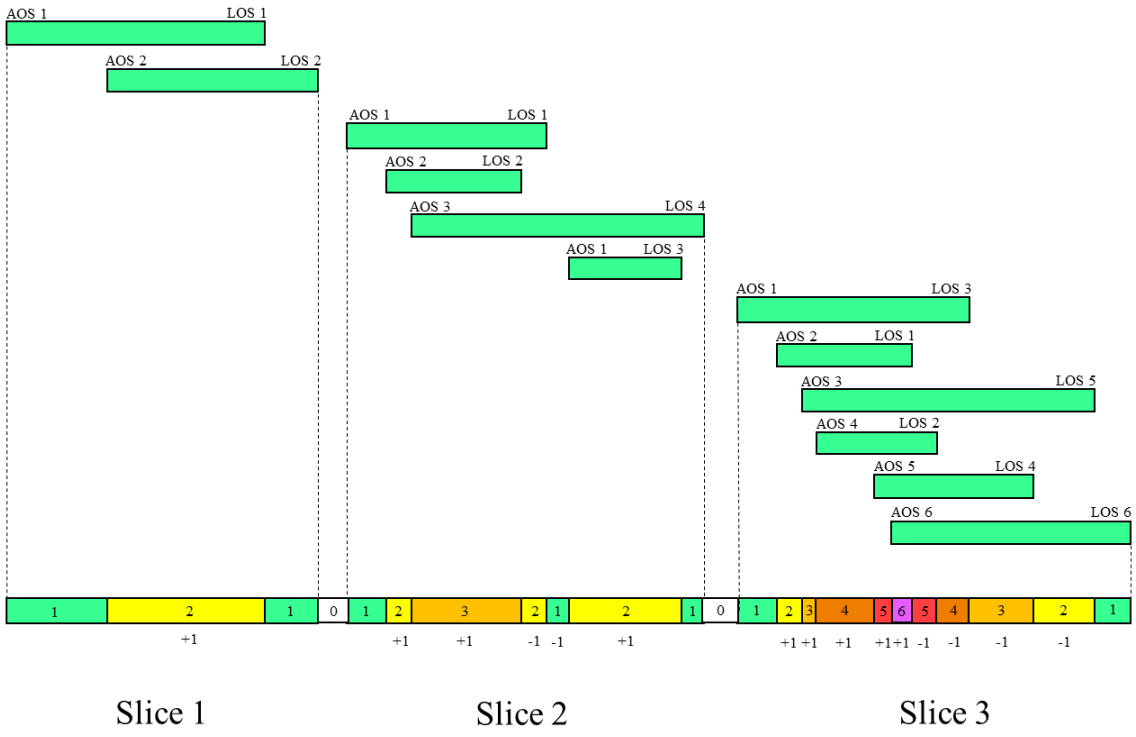

Figure 12. Chronological Satellite Access Conflicts

Since these subroutines are contained within the *Monte Carlo Loop* code and repeated continuously, run time was decreased by allowing STK to remain open for each iteration, rather than completely closing the program to start a new scenario each time after the satellites were deleted, however the time to run a large scenario is still not insignificant. To further understand the capability of this program and the reasoning behind the design, the simulation style is outlined below.

C. MONTE CARLO OPTIMIZATION

1. What Monte Carlo Simulations Do

According to Palisade, one of the leading software organizations in the risk and decision making tools industry [18], a Monte Carlo simulation is defined as:

> A computerized mathematical technique that allows users to account for risk in quantitative analysis and decision making. Monte Carlo simulation

> furnishes the decision-maker with a range of possible outcomes and the probabilities they will occur for any choice of action. Monte Carlo simulation performs risk analysis by building models of possible results by substituting a range of values—a probability distribution—for any factor that has inherent uncertainty. It then calculates results over and over, each time using a different set of random values from the probability functions. Depending upon the number of uncertainties and the ranges specified for them, a Monte Carlo simulation could involve hundreds or thousands of recalculations before it is complete and capable of giving an accurate assessment. Monte Carlo simulation produces distributions of possible outcome values.

2. Why Monte Carlo is Beneficial for This Specific Problem

This type of simulation is advantageous for this particular topic because the hundreds of iterations of randomly created scenarios produce a high confidence value in the percentages calculated. Conversely, with only a few iterations, a program such as this would not collect enough data to provide a reasonable mean value; there would be far too much deviation between cases. For example, prior to utilizing the Monte Carlo simulation to repeatedly loop the code, one scenario was created and saved, but this required the user to manually input different parameters to observe if there were any drastic changes. Rerunning the same program and detecting changes was still possible, but due to the random nature, one random simulation could be very different from the next. This creates a large range of uncertainty rather than statistical accuracy. Those initial scenarios were still very useful, but once the concepts were folded into a Monte Carlo operation, a different random scenario could be automatically generated for as many times as specified, providing much more confidence in the average.

With the reliable percentages calculated through the Monte Carlo algorithm, organizations or individuals can base decisions on whether or not to expand their ground station network for their projected number of satellites for future years. Through this simulation, the latitude of the additional ground station(s) can be evaluated.

D. SUMMARY

Chapter III discussed the interface method used to communicate from MATLAB script files, to scenario generation in STK, back to data storage through MATLAB. The justification behind the selection of a Monte Carlo algorithm was also given in this chapter, as well as a description of the benefits it provides to this type of analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TEST SCENARIOS AND RESULTS

This chapter will discuss the reasoning for selecting scenarios to illustrate the interesting nature of the *many satellite, few ground station* problem. The chapter will also present the results, observed trends, and the analysis of how these findings relate to the reference satellite missions outlined in Chapter II.

A. SCENARIO FORMULATION

The selection of scenarios began through trial and error to observe how the access conflicts grew as the satellite numbers increased. Beginning with a single, randomly generated satellite, and increasing by twos, tens, and eventually several-hundred-satellite intervals, the scenarios produced enough information to shed light on the results. The percentage of conflicted time was the first calculation for each scenario. This calculation was a simple ratio, displayed in equation (5), of the time spent with *multiple* (more than one) satellites overhead, compared to the duration where *at least* one satellite was overhead the ground station.

$$\%\ \text{Time Conflicted} = \frac{\text{Total duration of multiple satellites in view}}{\text{Total duration of at least 1 satellite in view}} \tag{5}$$

Table 3 shows the twenty-satellite step-size used to investigate the growing percentages of time spent with multiple satellites in view of the ground station. It also shows the percentages of the total scenario time spent conflicted with two through six satellites simultaneously in view, during each of the "10" to "70 random-satellite" scenarios. Table 3 only considers the orbital geometry of having multiple satellites in view, and the analysis does not yet incorporate the threshold data download requirements.

Table 3. Increasing Conflicts Percentages for Various Satellite Numbers and Time Periods

Number of Satellites	Scenario Duration	Total Duration of At Least One Sat in View (days)	Total Accesses	Total Number of Conflicts	Total Accesses Conflicted (%)	Time with 2 Sats in View (%)	Time with 3 Sats in View (%)	Time with 4 Sats in View (%)	Time with 5 Sats in View (%)	Time with 6 Sats in View (%)
10	1 Day	0.13	28	4	14.29	5.30	0	0	0	0
10	1 Week	0.88	192	19	9.90	5.20	0.10	0	0	0
10	1 Month	3.96	872	93	10.67	5.40	0	0	0	0
10	6 Months	23.14	5129	619	12.07	5.80	0.20	0	0	0
10	1 Year	46.21	10152	1089	10.73	5.10	0.20	0	0	0
30	1 Day	0.23	99	47	47.47	18.50	3.80	0.30	0	0
30	1 Week	2.71	703	290	41.25	17.10	2.50	0.20	0	0
30	1 Month	11.98	3109	1295	41.65	17.90	2.20	0.20	0	0
30	6 Months	71.21	18442	7666	41.57	17.60	2.20	0.20	0	0
30	1 Year	141.3	36621	15217	41.55	17.50	2.20	0.20	0	0
50	1 Day	0.44	137	77	56.20	21.30	6.40	1.10	0	0
50	1 Week	3.21	955	546	57.17	22.30	5.20	0.70	0	0
50	1 Month	14.54	4222	2329	55.16	21.90	4.20	0.50	0	0
50	6 Months	100.6	29220	19358	66.25	21.90	4.20	0.60	0	0
50	1 Year	171	49628	27765	55.95	21.90	4.20	0.50	0	0
70	1 Day	0.61	201	151	75.12	22.70	10.00	2.60	0	0
70	1 Week	4.20	1400	1149	82.07	28.10	8.90	2.20	0.40	0
70	1 Month	18.79	6205	5148	82.97	29.00	8.50	1.90	0.30	0
70	6 Months	130.7	42885	35762	83.39	27.80	8.50	2.00	0.30	0.10
70	1 Year	222.3	72790	60596	83.25	27.90	8.50	2.00	0.30	0

Table 4 illustrates a side-by-side comparison of unweighted and weighted contact time percentages as broken out by number of satellites simultaneously in view and satellite population quantities for one month (31 days). The average values for each number of satellites in view were calculated from fifteen iterations of random-satellite scenarios. The mean percent time of N satellites-in-view was then multiplied by N to determine the *weighted overhead time* for the entire scenario. The sum of these numbers resulted in a *total weighted overhead time* percentage, greater than the feasible one hundred percent that the system can support. For example, with a *total weighted overhead time* of 200%, any given satellite could expect half of the time overhead to be available for communications, and the average number of satellites in view for the duration of the scenario would be two. The analysis determines whether this is an adequate amount of contact time based on mission needs. If not, the ground station has reached saturation.

Table 4. Total Unweighted and Weighted Percent of Satellites Simultaneously in View

	15 Iterations, 1 Month Scenario, Random Satellites											
	Unweighted						Weighted					
	Satellite Scenario Number						Satellite Scenario Number					
Number of Satellites in View	1	10	100	250	500	1000	1	10	100	250	500	1000
0	98.65	86.04	24.95	2.91	0.09	0.00	98.65	86.04	24.95	2.91	0.09	0.00
1	1.35	12.98	35.41	10.44	0.63	0.00	1.35	12.98	35.41	10.44	0.63	0.00
2		0.94	24.01	18.59	2.18	0.01		1.87	48.01	37.18	4.36	0.02
3		0.04	10.79	21.86	5.15	0.04		0.12	32.38	65.59	15.46	0.11
4		0.00	3.63	18.99	9.12	0.13		0.00	14.54	75.94	36.48	0.52
5			0.95	13.24	12.81	0.37			4.74	66.19	64.06	1.86
6			0.21	7.70	15.00	0.85			1.26	46.19	90.02	5.11
7			0.04	3.78	15.07	1.71			0.29	26.43	105.5	11.97
8			0.01	1.61	13.18	2.98			0.05	12.85	105.5	23.88
9			0.00	0.61	10.20	4.72			0.01	5.47	91.84	42.45
10				0.20	7.04	6.59				1.97	70.43	65.94
11				0.06	4.42	8.47				0.62	48.57	93.18
12				0.02	2.58	9.96				0.21	30.92	119.5
13				0.00	1.34	10.77				0.06	17.41	140.0
14				0.00	0.66	10.74				0.01	9.25	150.3
15				0.00	0.31	10.00				0.00	4.63	150.0
16				0.00	0.13	8.72				0.00	2.04	139.5
17					0.05	7.14					0.91	121.4
18					0.02	5.51					0.39	99.11
19					0.01	4.02					0.16	76.44
20					0.00	2.80					0.03	56.05
21					0.00	1.84					0.01	38.56
22					0.00	1.14					0.00	25.05
23					0.00	0.68					0.00	15.57
24						0.38						9.22
25						0.21						5.18
26						0.11						2.84
27						0.06						1.55
28						0.03						0.80
29						0.01						0.41
30						0.01						0.22
31						0.004						0.12
32						0.002						0.05
33						0.0007						0.02
34						0.0002						0.01
35						0.0001						0.005
36						0.0001						0.004
37						0.00001						0.0003
Total Percent	100	100	100	100	100	100	100	101	162	352	699	1397
	Average Number of Satellites in View						1.00	1.01	1.62	3.52	6.99	13.97

The left-hand portion of Table 4 displays the *unweighted overhead time* percentages. For smaller satellite populations, the distribution trends towards fewer satellites in view on average. For all multiple-satellite cases, there are rare instances where a substantially larger than average quantity of satellites is immediately in view of the station. Even in the "1000-satellite" case, there are approximately 14 satellites in view at any one point in time but occasionally up to 37 satellites can be demanding the ground station's resources. Not too surprisingly, when the satellite population grows large enough, the station experiences virtually no inactivity.

In contrast, the right side of Table 4 provides the *weighted* contact time percentages. The total weighted overhead time percentage shows how over-subscribed the ground station is, but this over-subscription does not mean that the ground station is saturated. Saturation comes from whether or not the satellites can accomplish their threshold mission in the time available (the RTT), as calculated in Chapter II, equation (1). The total weighted overhead time percentage is a measure of how much capacity is demanded of the ground station when multiple satellites are in view.

Table 4 also shows the average number of satellites overhead throughout the 31-day scenario duration. For example, the scenario with "100 satellites" averages between one and two satellites overhead, for the amount of time that there are any satellites in view. Of the fifteen Monte Carlo iterations, the calculated average converged on 1.6 satellites always in view of the station. One interpretation of this result is that with all other factors kept equal, each satellite could expect their total overhead time available for communications contact to be divided by 1.6. If this amount of contact time were less than the RTT, then saturation would have occurred. Figure 13 shows the average number of satellites in view for the different Monte Carlo simulations.

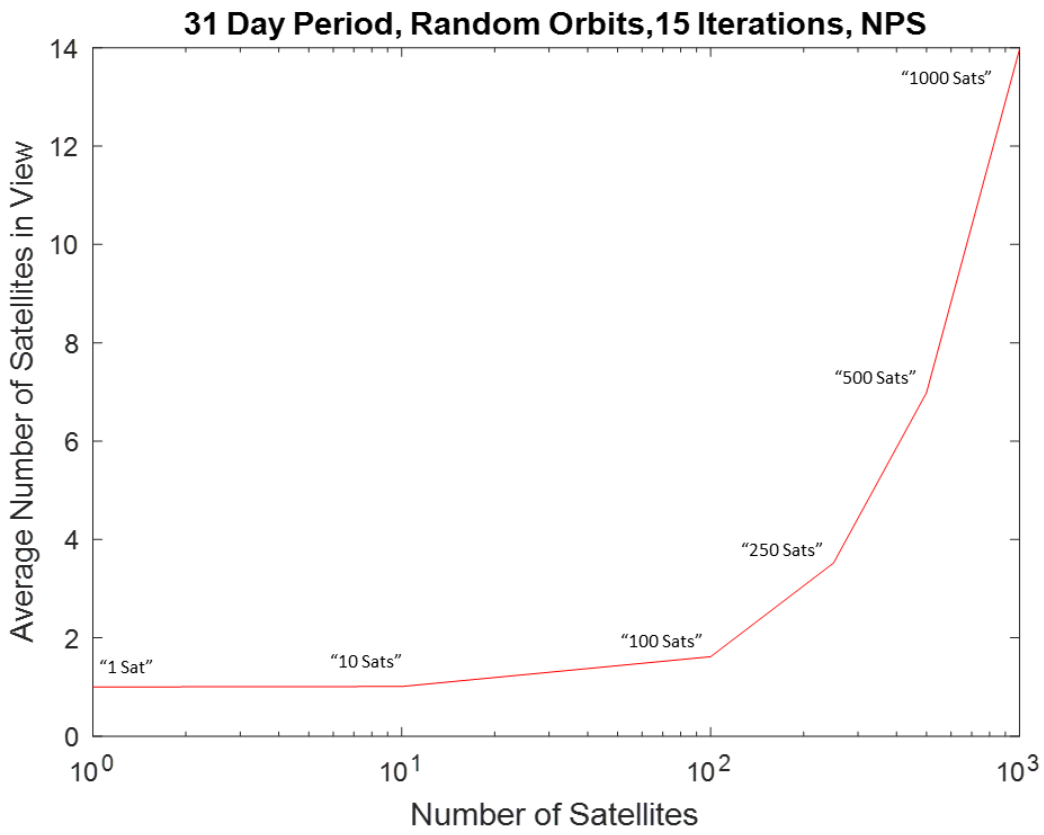

Figure 13. Average Satellites in View

To better visualize the rise in conflicting passes, Figure 14 displays conflict quantities in the form of a *box plot* with scenarios containing "10, 30, 50, 70, 90, 110, 130, and 150 satellites." It displays the increase in pass conflicts and the range of deviation when evaluating 20 iterations.

Box plots provide a visualization of summary statistics for data samples [19], allowing the study of a single group of measurements. The horizontal line across the width of the box indicates the median value from the data set. The bottom and top horizontal lines depict the 25th and 75th percentiles of the data, respectively. A variation of the box plot, the *notched box plot*, used in Figure 14, accentuates the varying locations of the medians, for the purpose of comparison.

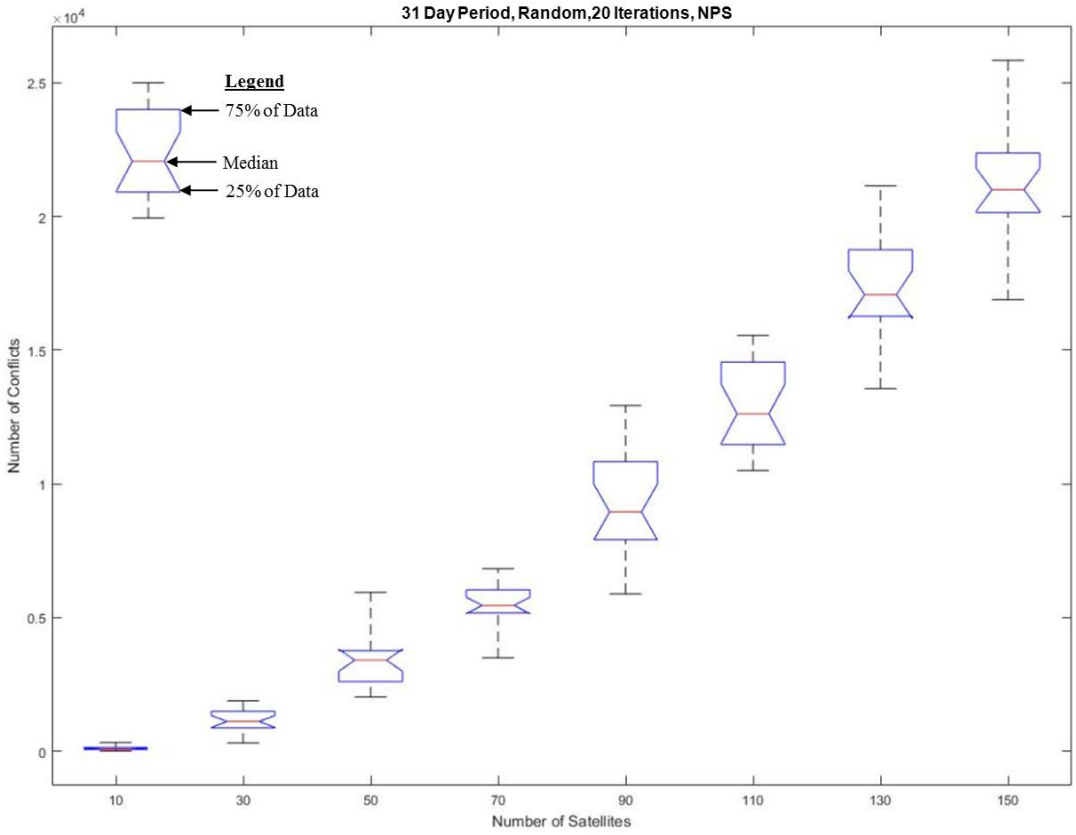

Figure 14. Box Plot of 20 Iterations for One-Month Duration Scenarios

All of the values contained within the box are defined as the *inter-quartile range*, since they fall between 25% and 75%, which shows the spread of the observation cases [20]. The brackets that extend vertically beyond the box account for the extreme cases, or outliers. This plotting method shows that since the medians in Figure 14 do not fall reliably at the center of each box, scenarios with many more iterations could be run to increase confidence in the results.

B. ANALYSIS AND OBSERVED TRENDS

The aspects of saturation identified in Chapter II are revisited in this section and come together to provide the definition of saturation. One characteristic of saturation comes from including current small satellite mission data requirements and weighing them against the *required threshold time* to complete a full data downlink versus the *total*

overhead time a satellite is physically in view of a station. If data requirements are minimal, multiple overlapping satellites may not pose serious resource conflicts for the station. However, data-intensive missions may more readily lead to saturation of the ground resources, compounded if there are additional overhead spacecraft. The low, medium, and high complexity mission parameters defined the minimum time requirement for each CubeSat, providing a useful parameter for mission planning.

It should be noted that contact windows used to generate the analysis in this section are missing a fraction of the satellite population in the scenario as mentioned in Chapter II. A percentage of the randomly generated satellites were placed into orbits that never had access to the ground station due to their orbital inclinations and the latitude of the station. This number seemed less significant for the smaller scenarios, but after creating "1000 satellite" scenarios it was observed that an average of 23%, or 230 satellites, would never see the ground station. The case of "1000 satellites," in reality, becomes approximately 770 satellites that were able to contact the ground site based on line-of-sight visibility. Although labeled as "N satellite" scenarios in the subsequent figures, only 77% of satellites in each scenario can access the ground station from above ten degrees elevation angle. And moving the ground station to a different latitude would produce different results. Of course, mission planners must seriously consider both the location of their station(s) and expected orbital geometries when seeking to maximize the network's performance.

To analyze the different mission sets outlined in Chapter II, the same Monte Carlo runs were applied to varying mission complexities. To reduce the number of variables in this initial analysis, it was assumed that the entirety of the satellite population corresponded to the same performance parameters presented in Table 2. Figure 15 displays the total data that can be collected at the ground station from the low complexity missions over one month. The *accumulated data* (red) represents the total possible data that the ground station can download for a particular scenario and is shown in equation (6). It was then compared to threshold (green) and objective (blue) totals for the 31-day duration, which were calculated using equations (7) and (8). These equations incorporate

the previously discussed *.77N* to account for the satellites that do not make any contact with the station.

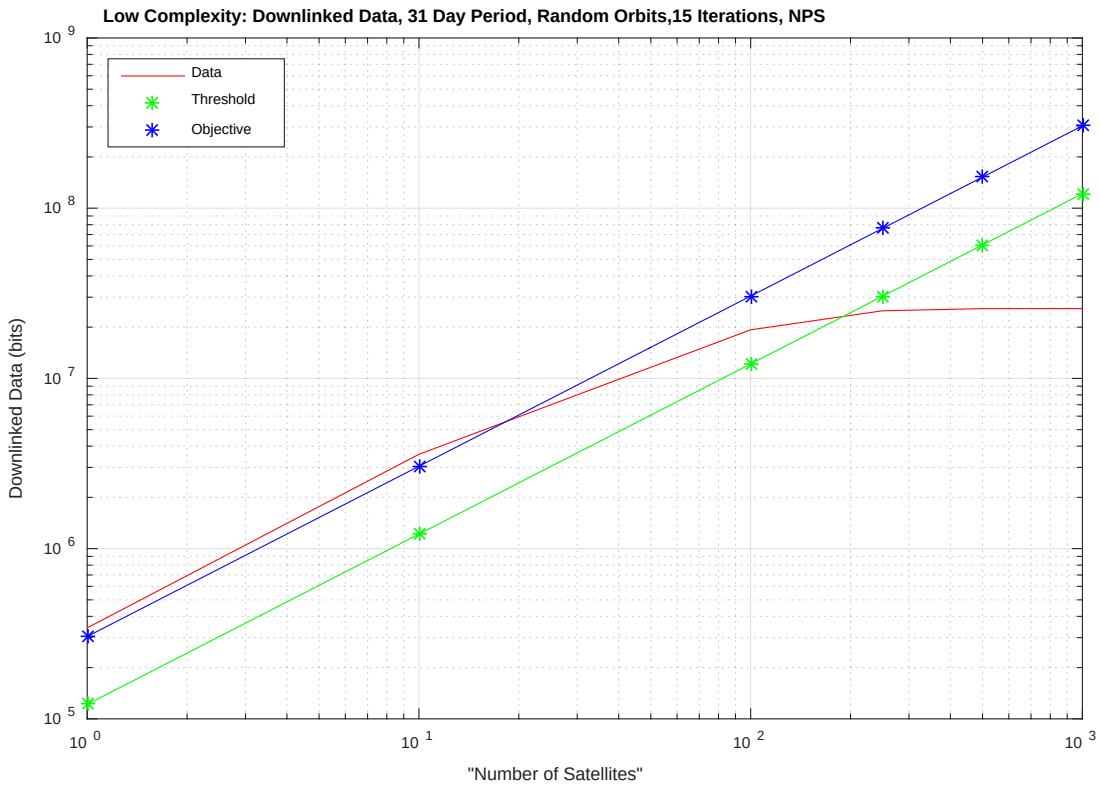

Figure 15. Low Complexity Mission Downloaded Data

$$Accumulated\ Data = (Unweighted\ Contact\ Time\ in\ Seconds)\ * (DR) * (EF) \quad (6)$$

$$Required\ Threshold\ Data = (.77) * (\#Satellites\ in\ Scenario)\ * (RTT) * \left(\frac{Average\ \#\ Passes}{Day}\right) * (31\ Days) \quad (7)$$

$$Required\ Objective\ Data = (.77) * (\#Satellites\ in\ Scenario)\ * (ROT) * \left(\frac{Average\ \#\ Passes}{Day}\right) * (31\ Days) \quad (8)$$

The threshold and objective data requirements for these low complexity missions are limited and it is possible to meet RTT at the NPS ground station until approximately 143 satellites that have contact with the ground station. The objective data requirements become unattainable with approximately 11 satellites, indicating either a need for additional ground stations or a reduction of required data. The red line depicting the

accumulated data in Figure 15 steadily increases until reaching an upper limit. Total overhead time initially increases thus accruing more downlinked data from the satellites. Eventually, there is always at least one satellite over the station, thus placing an upper boundary on *accumulated data* regardless of satellite population, hence the behavior of the total possible data to be downlinked, when multiplied by the data rate and efficiency. Also worth noting, the available portion of overhead time per satellite begins decreasing due to the larger number of satellites competing for downlink slots, and causing the overhead time to be split amongst them. Table 5 displays the values calculated by linear interpolation to understand the location at which the above scenario of low complexity satellite missions reaches threshold and objective saturation. The total and adjusted values correlate to the number of satellites; the total is the value including all cases in the scenario, the adjusted value takes into consideration the 23% of satellites that do not make contact with the ground station. The downlinked data in the right hand column corresponds with the number of satellites at which saturation occurs.

Table 5. Interpolated Saturation Values for Low Complexity Missions

Low Complexity Saturation Point			
		# of Satellites	Downlinked Data
Threshold	"Total"	185	~23 Mb
	Adjusted by .77	~143	
Objective	"Total"	14	~4 Mb
	Adjusted by .77	~11	

Figure 16 displays data collected from the medium complexity mission requirements. All other values pertaining to the scenario generation remain constant, aside from the data rate and download requirements. The improved downlink rate provides more total data from the satellites and the corresponding threshold and objective values remain attainable for a larger population than that of the low complexity missions. Table 6 shows the corresponding saturation points for this set of mission parameters.

The high complexity missions downlink the greatest quantities of data, due to their increased capabilities (higher data rates and better efficiency). However, as shown

in Figure 17, the objective requirement is never met and a threshold saturation point is reached at a reasonably large satellite population greater than “1000 satellites.” Table 7 parallels these values by showing the unattainable objective requirement, while displaying their impressive capacity to sustain the threshold requirement.

Mission planners should evaluate ground station performance as shown in Figures 15, 16, and 17 to determine a balance between the number of satellites, mission requirements, and the location (particularly the latitude) of the ground stations.

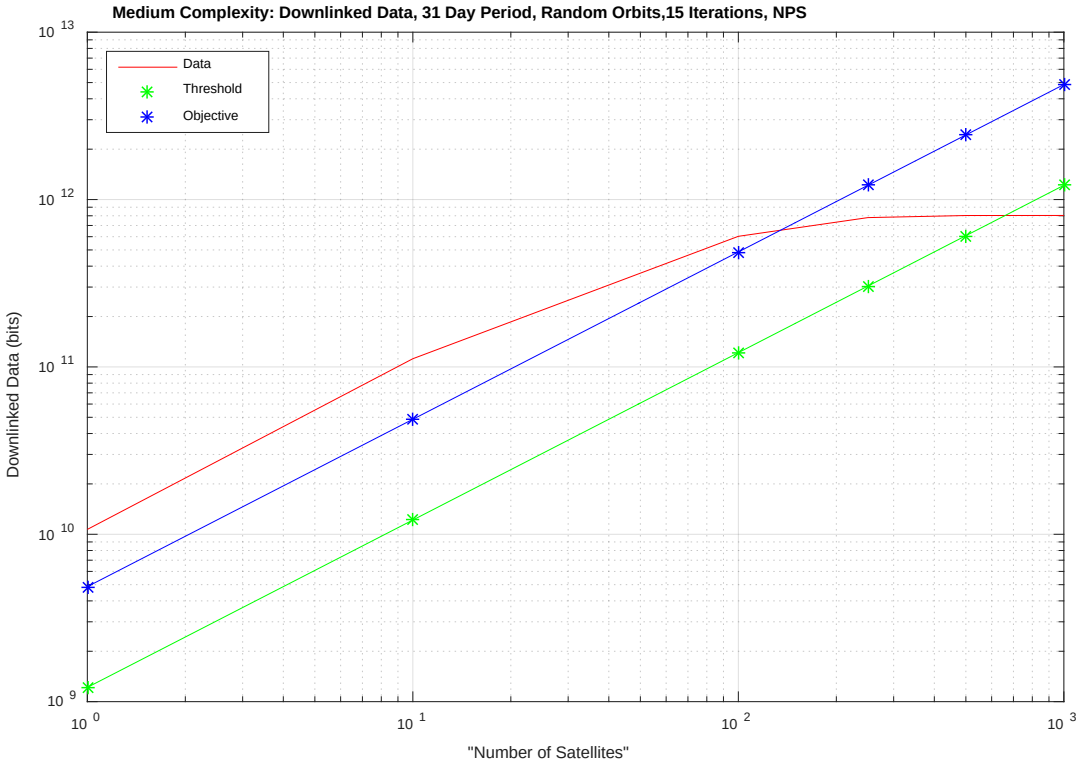

Figure 16. Medium Complexity Mission Downloaded Data

Table 6. Interpolated Saturation Values for Medium Complexity Missions

Medium Complexity Saturation Point			
		# of Satellites	Downlinked Data
Threshold	"Total"	660	~803 Gb
	Adjusted by .77	~508	
Objective	"Total"	132	~641 Gb
	Adjusted by .77	~102	

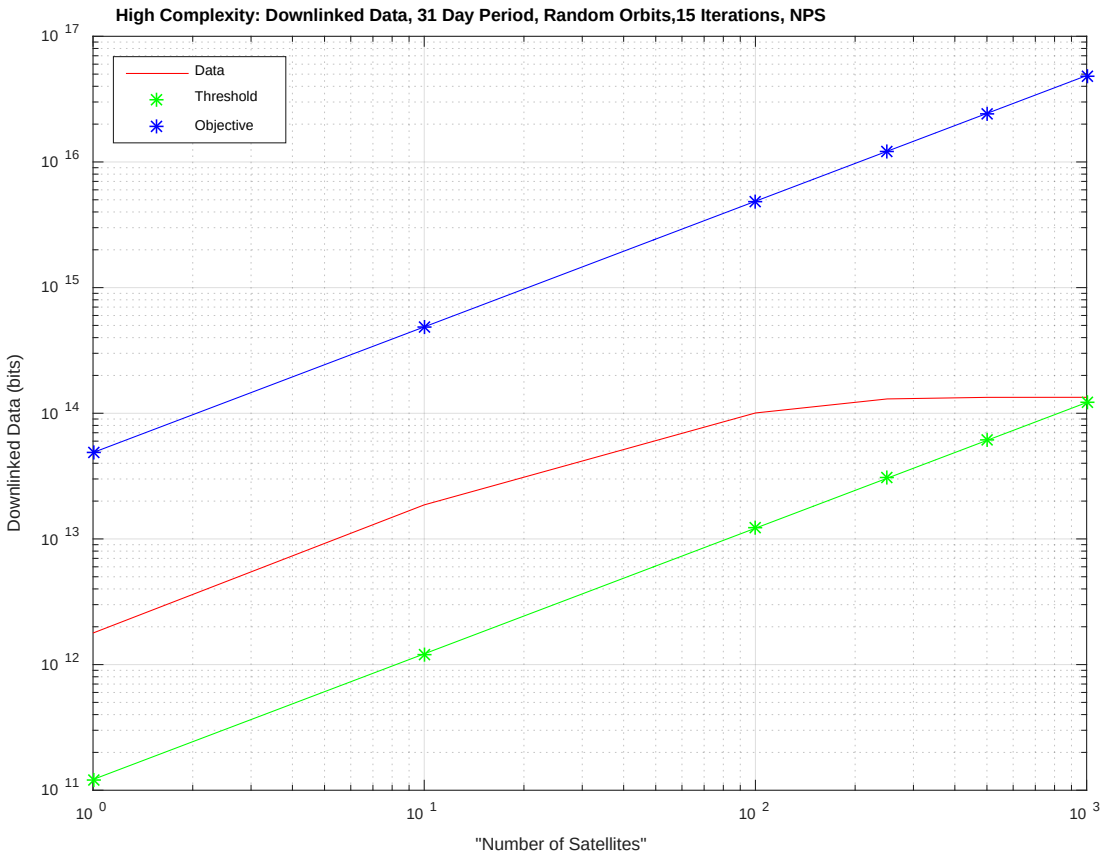

Figure 17. High Complexity Mission Downlinked Data

Table 7. Interpolated Saturation Values for High Complexity Missions

High Complexity Saturation Point			
		# of Satellites	Downlinked Data
Threshold	"Total"	>1000	>1.2e5 Gb
	Adjusted by .77		
Objective	"Total"	N/A	N/A
	Adjusted by .77	N/A	

A real-world consideration using the data presented in this section is that ground stations will likely service satellites of varying complexity. The analysis performed in this thesis assumed one type of mission at a time, thus simplifying the problem, but removing some of the practical aspects. In reality, a mix of low, medium, and high complexity spacecraft will be leveraging a ground station of potentially modest efficiency. This assumption would result in a shifted *point of saturation*. Knowing the details of their application, mission planners can use the tool developed in this research to scale the complexity level of each satellite, and take into consideration the differing time needed to achieve the minimum requirement for data download. Future work to analyze scenarios with a random mixture of low, medium, and high complexity missions in a Monte Carlo simulation would be able to add even more accuracy. Incorporating this realism into the scenarios provides a more precise assessment of when a ground station handling a multitude of satellites, with differing capabilities, reaches capacity.

Assessing ground station capacity is possible by incorporating the *saturation factor*, discussed in Chapter II. Saturation of a single ground station is determined in equation (9) using the following logic; if the average-number-of-satellites-in-view from Table 4, multiplied by the threshold saturation factor (TSF) from Table 2, results in a value greater than 1, then that ground station is saturated. This term is referred to as the *saturation score*. This method can be used for each mission complexity to assess the range between the threshold and objective data requirements by using the TSF and objective saturation factor (OSF). This determines whether those requirements are feasible with the satellite's data rate and ground station efficiency. The saturation score

provides the link between the geometry of the *overhead time*, and the data needs of the *required threshold time*, to give a value that grades the capacity of the ground station.

$$Saturation\ Score = (.77) * (Average\ \#\ of\ Satellites\ in\ View) * (Saturation\ Factor) \quad (9)$$

Examples of the *saturation score* for the NPS ground station are shown in Figures 18, 19 and 20. Low, medium, and high complexity missions are illustrated in individual plots, displaying each satellite population scenario. The RTT and ROT are the bounds for the downlinked data required for each of the scenarios, displayed in red. The vertical blue line depicts the point at which the ground station is considered saturated.

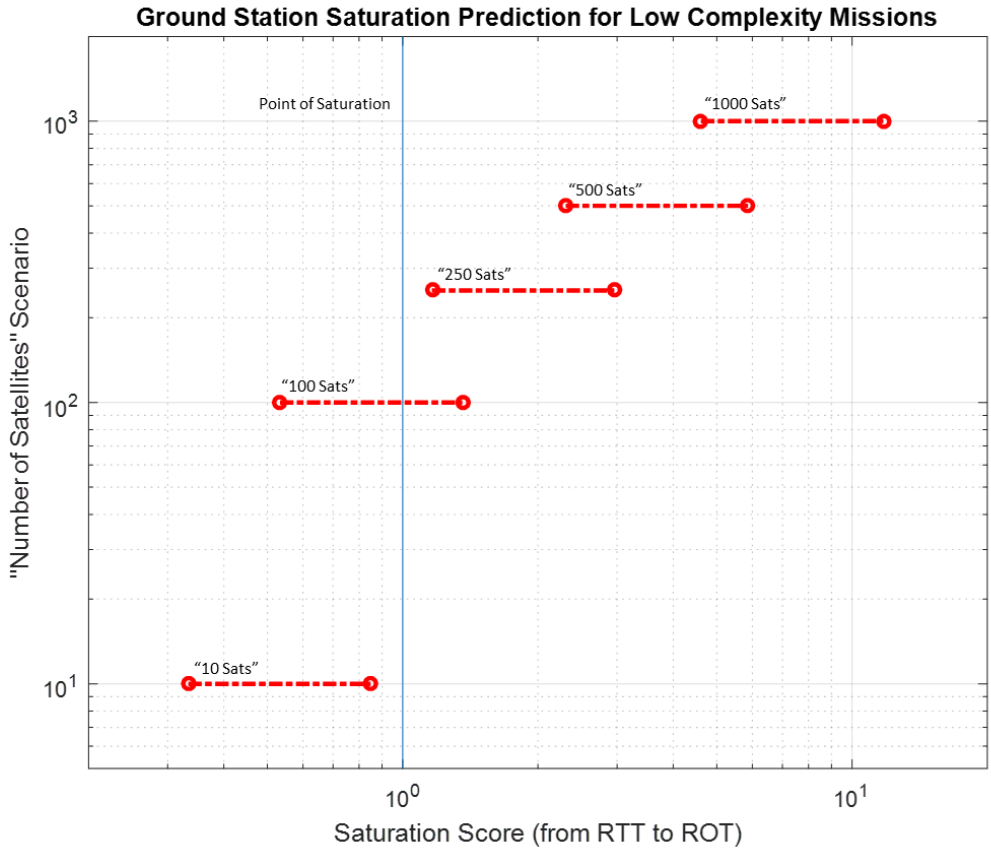

Figure 18. Saturation Scores for Low Complexity Mission Scenarios

By looking closely it is possible to tell where the low, medium, and high missions may meet their threshold and objective requirements. For example, the low complexity

missions in Figure 18 do not reach saturation for *threshold* requirements for the "10" and "100" cases, but reaches saturation only for the *objective* at "100 satellites," and completely exceeds both levels starting at the "250-satellite" mark.

The medium complexity missions shown in Figure 19 remain below the threshold saturation up through "100 satellites." The "250" and "500 satellite" cases meet the threshold but not the objective requirements. Finally, the "1000 satellite" case does not meet any.

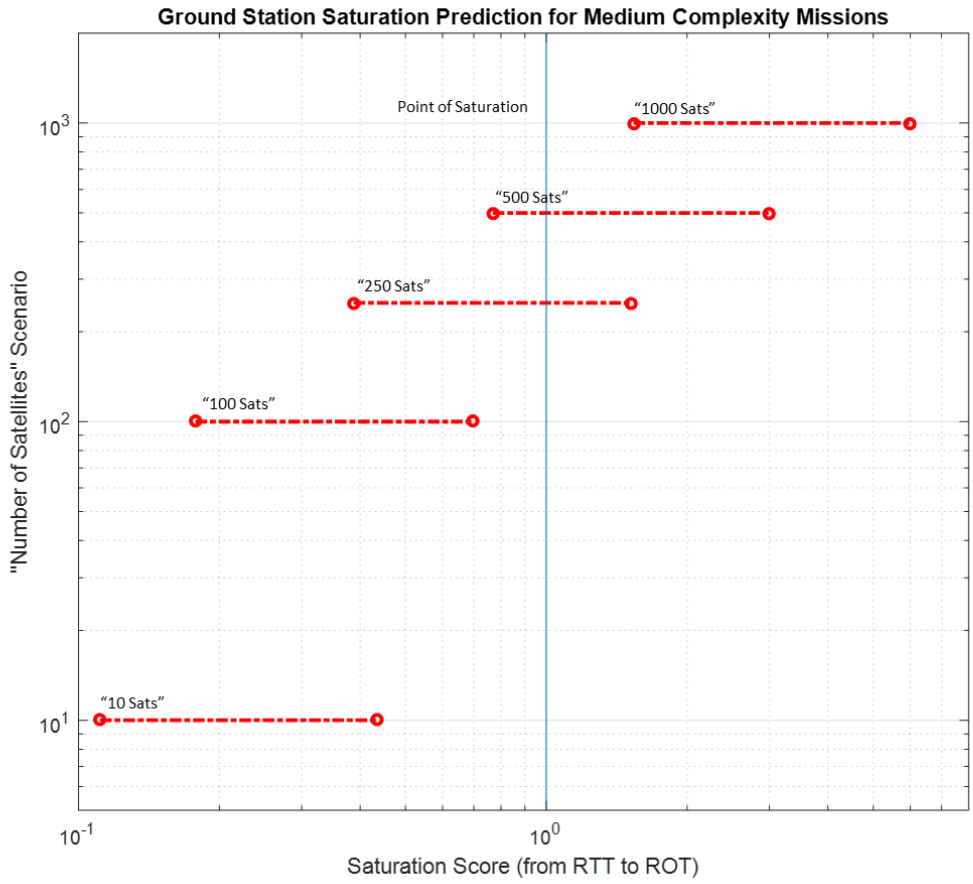

Figure 19. Saturation Scores for Medium Complexity Mission Scenarios

The high complexity missions displayed in Figure 20 only meet the threshold requirement for the "10-satellite" case. All other simulations fail to meet threshold and

objective requirements for this single ground station. Clearly, a high complexity, multi-satellite mission with sizable data requirements will need more than one station to satisfy its objectives.

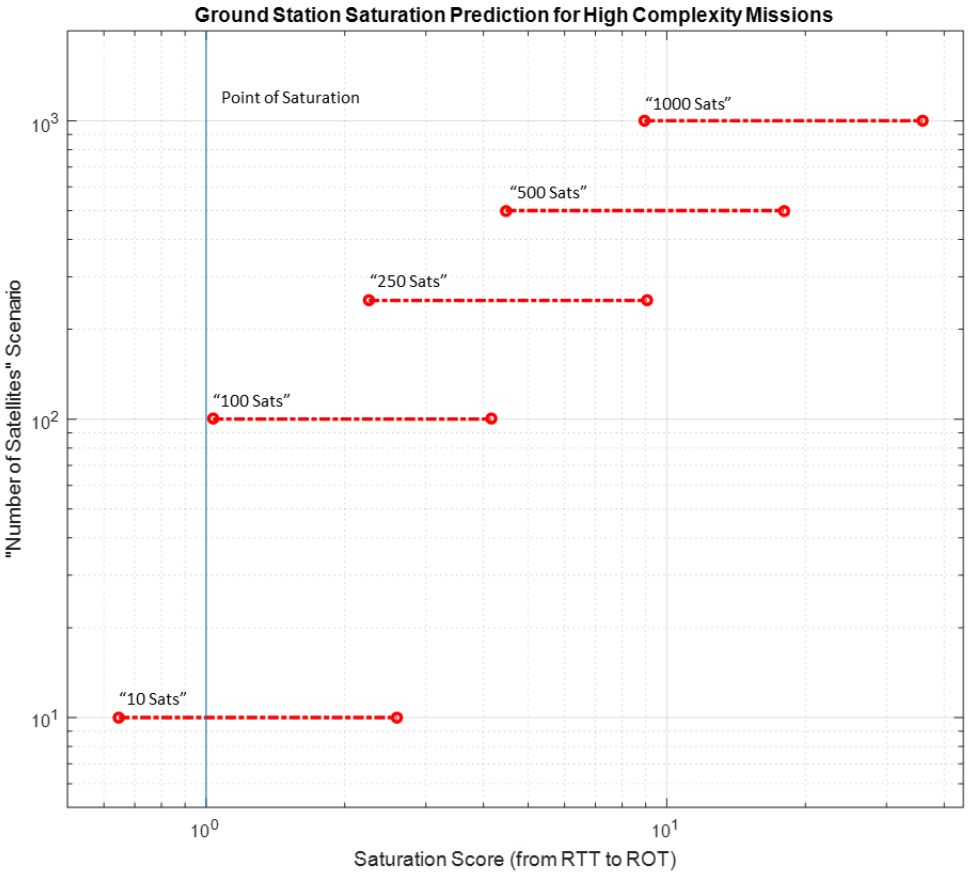

Figure 20. Saturation Scores for High Complexity Mission Scenarios

Depending on the mission data requirements, the orbits simulated, and the ground station latitude, the exact values in such a plot will differ. However, the saturation score is an objective way of grading the ability of the ground station to support these missions. Due to the generalization inherent to such an analysis, approaching a saturation score of one should be treated as an indicator of potential saturation and mission planners should allow adequate margin in their mission design.

C. SUMMARY

Chapter IV discussed the process behind the scenario selection and the data created from the Monte Carlo simulation. Plots were used to analyze trends in the data. Finally, the understanding gleaned from the simulation was applied to the reference satellite missions to provide context and validity to the conclusions drawn about when a ground station's capacity becomes saturated.

V. CONCLUSION

Chapter IV discussed the data and analysis supporting the findings to the initial question of: *at what point does a ground station experience saturation*. This final chapter will cover the implications of this research for varied solutions based on the Monte Carlo simulation. Such an analysis can help optimize mission planning for ground station operators and satellite owners. Finally, suggested areas of research are discussed for carrying these pursuits forward.

A. APPLICATIONS

Discussed at the beginning of this thesis, as the population of LEO CubeSats increases, the communication capacity of the terrestrial stations supporting them becomes strained. Ground station operations become considerably more challenging as higher numbers of satellites compete for finite resources. This thesis has explored how those systems are affected as the number of satellites proliferates.

Saturation was defined as the point at which satellite missions are, on average, not meeting their minimum data download requirements. These data requirements were determined from current, known CubeSat mission types. Figures 18 through 20, in Chapter IV, are the visual representation of the saturation data. With the known average number of satellites overhead and imposed data requirements, the point of saturation can be evaluated for any ground station.

The result of this thesis is a tool that could be developed further to help understand when there will be a significant decrease to the ground station's ability to effectively communicate with its satellites. At this threshold, an organization will be able to visualize the trends as the numbers of satellites and ground stations are changed. This provides planners with more information on when to consider adding more ground stations to their network, reallocating assets, or reducing data requirements by flying fewer satellites or collecting less data. The broad nature of the code should be useful for applications of interest to ground station program offices.

B. RECOMMENDATIONS FOR FOLLOW-ON RESEARCH AND POTENTIAL IMPLICATIONS

This thesis has covered the general issues associated with this challenging puzzle. Modifications could be made to the MATLAB code to consider different constructs. The next logical progression in analysis would be to consider scenarios such as "*one ground station, one constellation*," "*multiple ground stations, many random satellites*," and "*multiple ground stations, multiple constellations*." Data from varying constructs such as these would provide simple comparisons to the results collected in this thesis. Other factors such as latitude could be varied to assess the benefits to ground station contacts.

For an added level of realism, one could incorporate the orbital parameters more commonly used for CubeSats today, for example the many small satellites deployed from International Space Station, to provide a weighted value that produces more satellites with those parameters within the *randomly* selected orbital elements. These considerations for continuing research all play a significant role of the larger-scale challenge of scheduling these small spacecraft. Knowing the saturation point for a particular ground station network can help determine the approach to utilize for maximizing its capacity.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX . MATLAB SCRIPTS

A. MONTE_CARLO_LOOP

```
clear all; close all; clc;

% Main loop that calls Random_Satellite_Generator function
T1 = now;
MAXSATS = 1;                    % Set number of randomly generated satellites
MAX_ITERATIONS = 5;             % Set numer of iterations for Monte Carlo Loop to run
SAT_STEP =1;                    % Set the step size to increment number of sats to change
altlow = 400000;                % Set minimum altitude in m
althigh = 700000;               % Set maximum altitude in m
max_incl = 98;                  % Set maximum inclination in degrees
start_time  = '19 Jul 2017 02:00:00';
end_time    = '20 Jul 2017 02:00:00';
GS = {'NPS'}];

filepath = '\\special\ssagcommon$\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3
Optimization Threshold)\Results\'; % add/change "Day" to "Week" or "Month" depending on
scenario time
folderpath = strcat(filepath,int2str(MAX_ITERATIONS), 'Iterations_',
char(datetime('now','TimeZone','local','Format','y-M-d_HH-mm-ss\')));
mkdir(folderpath);

disp('Starting STK10')
app = actxserver('STK10.application');
root = app.Personality2;
root.Children.Import('\\special\ssagcommon$\Projects\Student Theses\Thesis - Writt, Andrea
(2018 MC3 Optimization Threshold)\Scenario\Monte_Carlo_MC3.sc');
sc = root.CurrentScenario;
station = root.GetObjectFromPath('/Facility/NPS')];
sc.SetTimePeriod(start_time, end_time);
root.ExecuteCommand('Animate * Reset');

for NUMSATS = MAXSATS:SAT_STEP:MAXSATS
   fprintf('starting: NUMSATS = %d at %s\n',NUMSATS, char(datetime('now')))
   for iteration = 1:MAX_ITERATIONS
      fprintf('Starting iteration: %d at %s\n', iteration, char(datetime('now')));
      Random_Satellite_Generator(iteration, NUMSATS, altlow, althigh, max_incl, start_time,
end_time, GS, app, root, sc, station, MAX_ITERATIONS, folderpath);
   end
end

T2 = now;
Runtime = T2 - T1;                      % In days
Run_min  = (Runtime * 86400) / 60;      % In minutes
Run_hrs  = Run_min / 60;
fprintf('Program ran for %6.2f minutes, or %6.2f hours, or %6.2f days\n',Run_min, Run_hrs,
Run_hrs/24)
clear app                               % closes STK 10
```

B. RANDOM_SATELLITE_GENERATOR

```
function Random_Satellite_Generator(iteration, NUMSATS, altlow, althigh, max_incl, start_time,
end_time, GS, app, root, sc, station, MAX_ITERATIONS, folderpath)

earthrad = 6378000;                    % Earth radius in m
    for i = 1:NUMSATS
        radlow = altlow + (althigh - altlow).*rand();
        radhigh = altlow + (althigh - altlow).*rand();
        R1 = radlow + earthrad;
        R2 = radhigh + earthrad;
        Rp = min(R1 R2]);
        Ra = max(R1 R2]);
        ecc = (Ra - Rp)/(Ra + Rp);

        satname = ‘‘,int2str(i)];
        sma = int2str((Ra+Rp)/2);
        rand_incl = int2str((max_incl)*rand());
        raan = int2str((360)*rand());
        mean_anomaly = int2str((360)*rand());
        arg_perigee = int2str((360)*rand());
        eccentricity = num2str(ecc);
        sat_create = sc.Children.New(‘eSatellite’,satname);
        cmd = ‘SetState */Satellite/’,satname ‘ Classical HPOP “‘, sc.StartTime,’ “ “‘, sc.StopTime,
‘“ 60 ICRF “‘, sc.StartTime, ‘“ ‘, sma, ‘ ‘, eccentricity, ‘ ‘, rand_incl,’ ‘, arg_perigee,’ ‘, raan, ‘ ‘,
mean_anomaly,’ “‘];
        root.ExecuteCommand(cmd);
        sat(i) = root.GetObjectFromPath(‘/Satellite/’, satname]);
        sat(i).Propagator.Propagate;
        satellite(i) = {satname};
    end

   accesses = 0;
   % For each satellite
   aos = ];
   los = ];
   gs = ];
   satnum = ];
   num_of_misses = 0;

   for i = 1:length(satellite)
      for j = 1:length(station)
        access = station(j).GetAccessToObject(sat(i));
        dp =];
        AOS = ];
        LOS = ];
        duration = ];
        access.ComputeAccess;

        try
          dp = access.DataProviders.Item(‘Access Data’).Exec(sc.StartTime, sc.StopTime);
          AOS = dp.DataSets.GetDataSetByName(‘Start Time’).GetValues;
          LOS = dp.DataSets.GetDataSetByName(‘Stop Time’).GetValues;
```

```
            duration = dp.DataSets.GetDataSetByName('Duration').GetValues;
        catch
            AOS = {start_time};
            LOS = {start_time};
            duration = {0};
            num_of_misses = num_of_misses +1;
            disp(strcat('AOS/LOS lists not generated_', satellite(i)))
            continue
        end

        % Datenum units are in days
        tempaos = datenum(AOS, 'dd mmm yyyy HH:MM:SS');
        lengthaos, y] = size(tempaos);  % lengthaos(1,1) = length of aos
        aos = cat(1, aos, tempaos);
        templos = datenum(LOS, 'dd mmm yyyy HH:MM:SS');
        los = cat(1, los, templos);

        for k = 1:lengthaos
            gs = cat(1, gs, j);
            satnum = cat(1, satnum, i);
        end
        accesses = accesses + lengthaos;

    end
  end
  sorted, durations, num_passes, num_slices, num_conflicts] = Conflict_Compiler( aos, los, gs,
satnum );

  filepath = '\\special\ssagcommon$\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3
Optimization Threshold)\Results\'; % change "Day" to "Week" or "Month" depending on scenario
time
  filename_mat = strcat(folderpath, int2str(NUMSATS), '_Rand_Sats_', int2str(iteration),
'Iteration', '.mat');

  total_duration = sum(durations) - durations(1);
  save(filename_mat, 'sorted', 'durations', 'total_duration', 'num_passes', 'num_slices',
'num_conflicts', 'NUMSATS', 'num_of_misses');
  fprintf('num_of_misses: %d\n', num_of_misses)
  fprintf('total duration of slices: %10.3f \n\n', total_duration)
  num_durations = length(durations);
  for i = 1 : num_durations
      fprintf('duration of %d sats in view: %10.3f \n',i-1,durations(i))
      if i > 1
          fprintf('percentage of pass time with %d sats:   %10.3f\n', i-1,
durations(i)/total_duration*100)
      end
  end

  for i = 1:NUMSATS
      sc.Children.Unload('eSatellite',cell2mat(satellite(i)));
  end
end
```

C. CONFLICT_COMPILER

```
%% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
function sorted, durations, num_passes, num_slices, num_conflicts] = Conflict_Compiler(aos, los,
gs, satnum)
%%
%   This function computes the conflicts in a series of passes given:
%   aos - list of aos's
%   los - list of los's
%   gs  - list of gs number
%   satnum - list of the satnum for each access
%
%   The function makes a list of all the aos's and los's in time order
%   with the attribute of aoslos_index of 0 for AOS and 1 for LOS.
%
%   The smusher flattens the aos and los and sorts and then walks through
%   the list, compiling a conflict_index for every bin, where a bin is the
%   space between every aos and los time. The bin total starts at 0 and
%   every AOS adds 1 and every LOS subtracts 1. So the number in the
%   conflict_index list is the number of sats currently accessing that gs.
%   E.g. A simple overlap would look like:
%   aoslos:          aos1   aos2   los1   los2
%   conflict_index:    1      2      1      0
%
%   0 is a gap with no access.  2 signifies 2 sats able to access the gs
%   at the same time.

% returns an array of # of aos, # of dim]
num_passes = length(aos) ;
ones    = zeros(num_passes, 1);
ones(:) = 1;
minus_ones    = zeros(num_passes, 1);
minus_ones(:) = -1;

% do the actual smushing and sorting
list1 = cat(1, aos, los);
list2 = cat(1, ones, minus_ones);
aosloslist = cat (2, list1, list2);
sorted = sortrows(aosloslist);
num_lines = length(sorted);

% now adds up the aos's and los's in turn to get the # of sats in view
% for each slice of time, make that the third column
summ = 0;
for i = 1 : num_lines;
   summ = summ + sorted(i,2);
   sorted(i,3) = summ;
end

if ~isempty(sorted)
   % find the max # of sats in view for adding up the durations in each slice
   % 0 sats in view in the first column, then 1 sat, 2 sats,
   max_sats = max(sorted,],1);
   num_durations = max_sats(3) + 1;
```

```
else
    max_sat = 0;
    num_durations = 1;
    sorted = 0 0 0];
end

% now add up the durations in each slice and put in the durations list
durations = zeros(1, num_durations);
for ii = 1 : num_lines - 1;
    jj = sorted(ii, 3);
    slice_duration = sorted(ii+1,1) - sorted(ii,1);
    durations(jj+1) = durations(jj+1) + slice_duration;
end

num_slices = sum(sorted(:,3)==0);
num_conflicts = sum(sorted(:,3)>1);
total_duration = sum(durations) - durations(1);

end
```

D. PLOTTING SCRIPTS

1. Boxplot

```
clear all

MAX_ITERATIONS = 10;
filepath = 'Z:\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization
Threshold)\Results\';
% filefolders = dir(filepath)
filenames = dir(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', '*.mat'));
num_mats = length(filenames);

figure
hold on
index = 1;
col_index = 1;
for i = 1:num_mats
    load(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', filenames(i).name));
    conflicts(index, col_index) = num_conflicts;
    if ~mod(i,MAX_ITERATIONS)
        num_sat_vec(col_index) = NUMSATS;
        index = 1;
        col_index = col_index + 1;
    else
         index = index + 1;
    end
end

boxplot(conflicts, num_sat_vec, 'Notch' , 'on' , 'Whisker' , 3)
xlabel('Number of Satellites')
ylabel('Number of Conflicts')
```

```
title(strcat('31 Day Period, Random, ', int2str(MAX_ITERATIONS),' Iterations, NPS'))
hold off
```

2. Downlinked Data

```
clear all;

% % high complexity rates
data_rate = 100000000;                % bps
efficiency = .5;                      % fluff factor
min_data_req = 1270000000;            % minimum data required in bits, per satellite, per pass
max_data_req = 507900000000;          % maximum data required in bits, per satellite, per pass
%
% % medium complexity rates
% data_rate = 1000000;                % bps
% efficiency = .3;                    % fluff factor
% min_data_req = 12700000;            % minimum data required in bits, per satellite, per pass
% max_data_req = 50790000;            % maximum data required in bits, per satellite, per pass

% % low complexity rates
% data_rate = 9600;                   % bps
% efficiency = .001;                  % fluff factor
% min_data_req = 1270;                % minimum data required in bits, per satellite, per pass
% max_data_req = 3180;                % maximum data required in bits, per satellite, per pass

MAX_ITERATIONS = 15;
filepath = 'Z:\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization
Threshold)\Results\';
filenames = dir(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', '*.mat'));      % read in
each .mat
num_mats = length(filenames);

index = 1;
iteration_index = 1;
for i = 1:num_mats
    load(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', filenames(i).name)); % load in all
data from .mat

    % Keep track of iteration number
    iteration = iteration_index;
    iteration_index = iteration_index + 1;
    if iteration_index > MAX_ITERATIONS
        iteration_index = 1;
    end

    num_durations = length(durations);
    num_lines, num_cols] = size(sorted);

    % now add up the durations in each slice and put in the durations list
    weighted_durations = zeros(1, num_durations);
    accumulated_data = zeros(1,num_durations);
```

```
    dropped_data = zeros(1,num_durations);

    for ii = 1 : num_lines - 1;
        jj = sorted(ii, 3);                     % number of sats in view at a time
        slice_duration = sorted(ii+1,1) - sorted(ii,1);
        weighted_durations(jj+1) = weighted_durations(jj+1) + jj*slice_duration;
        accumulated_data(jj+1) = accumulated_data(jj+1) +
slice_duration*3600*24*data_rate*efficiency;
        dropped_data(jj+1) = dropped_data(jj+1) + (jj-
1)*slice_duration*3600*24*data_rate*efficiency;
    end

    for j = 1 : num_durations
        if j > 1
            weighted_conflict_time_days(index,1:7) = weighted_durations(j), j-1,NUMSATS, iteration,
total_duration, accumulated_data(j), dropped_data(j)];      % sorts from first to third columns by
NUMSATS
            index = index + 1;
        end
    end
end

sorted_conflict_time_days= sortrows(weighted_conflict_time_days, 3, 2, 4]);     % sorts three
times, first column 3 (number of iterations), then column 2 (sats in view)
sat_scenarios = sorted_conflict_time_days(1,3);                                  % Seeded with
initial value
iteration_values = sorted_conflict_time_days(1,4);
max_conflicts = max(sorted_conflict_time_days(:,2));
sat_scenarios = unique(sorted_conflict_time_days(:,3));                          % Find number of
satellites for all cases
iteration_values = unique(sorted_conflict_time_days(:,4));                       % Find all iteration
numbers used

legend_vec = ];
plotting_array_index = 1;
for i = 1:length(sat_scenarios)
    for j = 1:length(iteration_values)
        x = find(sorted_conflict_time_days(:,3)==sat_scenarios(i));              % identifies where
3rd column of sorted_conflict_time_days moves from one scenario to next
        y = find(sorted_conflict_time_days(:,4) == iteration_values(j));         % identifies
where the 2nd column equals the number of conflicts
        z = intersect(x,y);                                                      % find common
indicies where conflicts and sat scenario number intersect
        sum_weighted_conflict_time_percent(j) =
sum(sorted_conflict_time_days(z,1)./sorted_conflict_time_days(z,5));      % sum of durations that
have the same NUMSATS and same iteration number
        sum_accumulated_data(j) = sum(sorted_conflict_time_days(z,6));
        sum_dropped_data(j) =  sum(sorted_conflict_time_days(z,7));

    end
    avg_weighted_conflict_time_percent = mean(sum_weighted_conflict_time_percent);       %
takes average of a single NUMSAT case across all available iterations
    avg_accumulated_data = mean(sum_accumulated_data);
    avg_dropped_data = mean(sum_dropped_data);
```

```
    sat_scenarios(i);
    plotting_array_desired_time(plotting_array_index, 1:2) = sat_scenarios(i),
avg_weighted_conflict_time_percent];
    plotting_array_accumulated_data(plotting_array_index, 1:2) = sat_scenarios(i),
avg_accumulated_data];
    plotting_array_dropped_data(plotting_array_index, 1:2) = sat_scenarios(i), avg_dropped_data];
    plotting_array_min_data(plotting_array_index, 1:2) = sat_scenarios(i),
sat_scenarios(i)*min_data_req*4.5*31];            % in bits
    plotting_array_max_data(plotting_array_index, 1:2) = sat_scenarios(i),
sat_scenarios(i)*max_data_req*4.5*31];          % in bits

    plotting_array_index = plotting_array_index + 1;
end

% figure
% semilogx(plotting_array_desired_time(:,1), plotting_array_desired_time(:,2), 'r-')
% hold on
% xlabel('Number of Satellites')
% ylabel('Average Number of Satellites in View')
% title(strcat('31 Day Period, Random Orbits,  ', int2str(MAX_ITERATIONS),' Iterations, NPS'))
% hold off

figure
loglog(plotting_array_accumulated_data(:,1), plotting_array_accumulated_data(:,2), 'r-')
hold on
loglog(plotting_array_desired_time(:,1), plotting_array_min_data(:,2), 'g*')
loglog(plotting_array_desired_time(:,1), plotting_array_max_data(:,2), 'b*')
xlabel('Number of Satellites')
ylabel('Downlinked Data (bits)')
title(strcat('High Complexity: Downlinked Data, 31 Day Period, Random Orbits, ',
int2str(MAX_ITERATIONS),' Iterations, NPS'))
legend('Data', 'Threshold', 'Objective', 'Location', 'northwest')
hold off

% figure
% semilogx(plotting_array_dropped_data(:,1), plotting_array_dropped_data(:,2)./(8*1000), 'r-')
% hold on
% xlabel('Number of Satellites')
% ylabel('Dropped Data (bits)')
% title(strcat('Low Complexity: Dropped Data, 31 Day Period, Random, ',
int2str(MAX_ITERATIONS),' Iterations, NPS'))
% hold off
```

3. Saturation Score

```
clear all;
data_rate = 100000000;                  % bps

% PICK complexity level here, low, med, or high
low_TSF = 0.33; med_TSF = 0.11;  high_TSF = 0.64;
low_OSF = 0.84; med_OSF = 0.43; high_OSF = 2.57;
```

```
TSF = high_TSF;
OSF = high_OSF;

plotting_param = 'r-.o';
figure

MAX_ITERATIONS = 15;
filepath = 'Z:\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization Threshold)\Results\';
filenames = dir(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', '*.mat'));      % read in each .mat
num_mats = length(filenames);

index = 1;
iteration_index = 1;
for i = 1:num_mats
   load(strcat(filepath, int2str(MAX_ITERATIONS), '_Iterations\', filenames(i).name)); % load in all data from .mat

   % Keep track of iteration number
   iteration = iteration_index;
   iteration_index = iteration_index + 1;
   if iteration_index > MAX_ITERATIONS
      iteration_index = 1;
   end

   num_durations = length(durations);
   num_lines, num_cols] = size(sorted);

   % now add up the durations in each slice and put in the durations list
   weighted_durations = zeros(1, num_durations);
   unweighted_durations = zeros(1, num_durations);
   accumulated_data = zeros(1,num_durations);
   dropped_data = zeros(1,num_durations);

   for ii = 1 : num_lines - 1;
      jj = sorted(ii, 3);                    % number of sats in view at a time, including 0, 1, 2...
      slice_duration = sorted(ii+1,1) - sorted(ii,1);
      unweighted_durations(jj+1) = unweighted_durations(jj+1) + slice_duration;
      dropped_data(jj+1) = dropped_data(jj+1) + (jj-1)*slice_duration*3600*24*data_rate*efficiency;

      % if no sats in view, don't accumulate data, otherwise...accumulate!
      if jj ~= 0
         accumulated_data(jj+1) = accumulated_data(jj+1) + slice_duration*3600*24*data_rate*efficiency;
         weighted_durations(jj+1) = weighted_durations(jj+1) + jj*slice_duration;    % indexed as jj+1 because sometimes there are 0 sats in view and you can't have a 0 index
      else
         accumulated_data(jj+1) = 0;
         weighted_durations(jj+1) = weighted_durations(jj+1) + slice_duration;      % count accumulated time for zero sats without actually multiplying by zero
```

```
        end
    end

    total_duration_recalc = sum(unweighted_durations);
    total_duration = total_duration_recalc;

    for j = 1 : num_durations
%       if j > 1                % if you don't want to consider 0 satellite cases
        weighted_conflict_time_days(index,1:8) = weighted_durations(j), j-1, NUMSATS, iteration,
total_duration, accumulated_data(j), dropped_data(j), unweighted_durations(j)];      % sorts from
first to third columns by NUMSATS
        index = index + 1;
%       end
    end
end

sorted_conflict_time_days= sortrows(weighted_conflict_time_days, 3, 2, 4]);     % sorts three
times, first column 3 (number of iterations), then column 2 (sats in view)
sat_scenarios = sorted_conflict_time_days(1,3);                                 % Seeded with
initial value
iteration_values = sorted_conflict_time_days(1,4);
max_conflicts = max(sorted_conflict_time_days(:,2));
sat_scenarios = unique(sorted_conflict_time_days(:,3));                         % Find number of
satellites for all cases
iteration_values = unique(sorted_conflict_time_days(:,4));                      % Find all iteration
numbers used
sats_in_view_times = ];

legend_vec = ];
plotting_array_index = 1;
for i = 1:length(sat_scenarios)
    for j = 1:length(iteration_values)
        x = find(sorted_conflict_time_days(:,3) == sat_scenarios(i));           % identifies where
3rd column of sorted_conflict_time_days moves from one scenario to next
        y = find(sorted_conflict_time_days(:,4) == iteration_values(j));        % identifies
where the 2nd column equals the number of conflicts
        z = intersect(x,y);                                                     % find common indicies where
conflicts and sat scenario number intersect
        sum_weighted_conflict_time_percent(j)   =
sum(100.*sorted_conflict_time_days(z,1)./sorted_conflict_time_days(z,5));      % sum of
durations that have the same NUMSATS and same iteration number
        sum_weighted_conflict_time(j)          =
sum(sorted_conflict_time_days(z,1)./sorted_conflict_time_days(z,5));      % sum of durations that
have the same NUMSATS and same iteration number

        sats_in_view_times(1:length(z), 1:3, j) =
100.*sorted_conflict_time_days(z,1)./sorted_conflict_time_days(z,5),
sorted_conflict_time_days(z,2),
100.*sorted_conflict_time_days(z,8)./sorted_conflict_time_days(z,5)];

        sum_accumulated_data(j)             = sum(sorted_conflict_time_days(z,6));
        sum_dropped_data(j)                 = sum(sorted_conflict_time_days(z,7));
    end
```

```
    % Compensate for divide by 0 NaN (happens when no satellites in view of
    % GS so total duration is 0)
    sats_in_view_times(isnan(sats_in_view_times)) = 100;
    sum_weighted_conflict_time(isnan(sum_weighted_conflict_time)) = 0;

    avg_sats_in_view_times_temp = mean(sats_in_view_times,3, 'omitnan');

    max_sats_in_view = max(max(sats_in_view_times(:,2,:)));
    sat_scenarios(i);
    avg_sats_in_view_times = avg_sats_in_view_times_temp(:,1), 0:1:max_sats_in_view]',
avg_sats_in_view_times_temp(:,3)];
    adj_avg_sats_in_view = sum(avg_sats_in_view_times_temp(:,1))/100;

    filepath = 'Z:\Projects\Student Theses\Thesis - Writt, Andrea (2018 MC3 Optimization
Threshold)\Results\';

    avg_weighted_conflict_time_percent = mean(sum_weighted_conflict_time_percent);        %
takes average of a single NUMSAT case across all available iterations
    avg_sats_in_view = mean(sum_weighted_conflict_time);
    avg_accumulated_data = mean(sum_accumulated_data);
    avg_dropped_data = mean(sum_dropped_data);

    plotting_array_desired_time(plotting_array_index, 1:2) = sat_scenarios(i),
avg_weighted_conflict_time_percent];
    plotting_array_accumulated_data(plotting_array_index, 1:2) = sat_scenarios(i),
avg_accumulated_data];
    plotting_array_dropped_data(plotting_array_index, 1:2) = sat_scenarios(i), avg_dropped_data];
    plotting_array_min_data(plotting_array_index, 1:2) = sat_scenarios(i),
sat_scenarios(i)*min_data_req*3.09*31];                % in bits
    plotting_array_max_data(plotting_array_index, 1:2) = sat_scenarios(i),
sat_scenarios(i)*max_data_req*3.09*31];              % in bits
    plotting_array_avg_sats_in_view(plotting_array_index, 1:2) = sat_scenarios(i),
avg_sats_in_view];
    plotting_array_adj_avg_sats_in_view(plotting_array_index, 1:2) = sat_scenarios(i),
adj_avg_sats_in_view];

    plotting_array_saturation_score(plotting_array_index, 1:3) =
avg_weighted_conflict_time_percent*TSF, avg_weighted_conflict_time_percent*OSF,
sat_scenarios(i)];
    plotting_array_index = plotting_array_index + 1;
end

for i = 2:length(sat_scenarios)
    xaxis = plotting_array_saturation_score(i,1:2)./100;
    yaxis = plotting_array_saturation_score(i,3), plotting_array_saturation_score(i,3)];
    loglog(xaxis, yaxis,plotting_param, 'LineWidth',2,'MarkerSize',5)
    hold on
end

grid on
loglog( 1 1],5 2000])
```

```
xlim(.5 44])
ylim(5 2000])
xlabel('Saturation Score (from RTT to ROT)')
ylabel('"Number of Satellites" Scenario')
title('Ground Station Saturation Prediction for High Complexity Missions')
```

LIST OF REFERENCES

[1] S. Loff, “CubeSats overview,” NASA, February 14, 2018. Online]. Available: https://www.nasa.gov/mission_pages/cubesats/overview

[2] E. Mabrouk, “What are SmallSats and CubeSats?” NASA, August 6, 2017. Online]. Available: https://www.nasa.gov/content/what-are-smallsats-and-cubesats

[3] C. Kief, N. Buonaiuto, M. Louie, J. Aarestad, B. Zufelt, R. Mital, R. Monical, R. Sivilli, and A. Bhopale, “Emergent Trends for CubeSat Ground Systems – A University View,” in *Conference on Small Satellites, SmallSat 2017*. Online] Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3756&context=smallsat

[4] S. C. Spangelo, D. Kaslow, C. Delp, B. Cole, L. Anderson, E. Fosse, B. S. Gilbert, L. Hartman, T. Kahn, and J. Cutler, “Applying Model Based Systems Engineering (MBSE) to a Standard CubeSat,” in *Aeropsace Conference, 2012 IEEE*. Online]. doi: 10.1109/AERO.2012.6187339

[5] S. C. Spangelo, “Modeling and Optimizing Space Networks for Improved Communication Capacity,” Ph.D. dissertation, Aero. Engr., Univ. of Michigan, Ann Arbor, MI, USA, 2013.

[6] G. Minelli, M. Karpenko, M. Ross, and J. H. Newman, “Autonomous Operations of Large-Scale Satellite Constellations and Ground Station Networks,” presented at the AAS/AIAA Astrodynamics Specialist Conf. 2017, Columbia River Gorge, Stevenson, WA, USA, Aug. 24, 2017.

[7] J. J. Leone, “CubeSat Pass Quality Analysis and Predictive Model,” M.S. thesis, Space Systems Academic Group, NPS, Monterey, CA, USA, 2018.

[8] “Remote Sensing Glossary,” Reference Information for Virtual Nebraska, University of Nebraska-Lincoln, 2005. Online]. Available: http://www.casde.unl.edu/glossary/k.php

[9] M. Swartwout, “CubeSat Database,” AENG 3150 (Astrodynamics), Saint Louis University, April 26, 2018. Online]. Available: https://sites.google.com/a/slu.edu/swartwout/home/cubesat-database

[10] Woensdag, “World Population Distribution By Latitude and Longitude – 2015,” DATAGRAVER, September 21, 2016. Online]. Available: http://www.datagraver.com/case/world-population-distribution-by-latitude-and-longitude-2015

[11] “Monterey, CA, USA Geographic Information,” LatLong. Accessed April 29, 2018. Online]. Available: https://www.latlong.net/place/monterey-ca-usa-2833.html

[12] H. J. Kramer, “DICE (Dynamic Ionosphere CubeSat Experiment), DICE-1 and DICE-2,” EO Portal, Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/d/dice

[13] H. J. Kramer, “AeroCube 7-OCSD (Optical Communication and Sensor Demonstration,” EO Portal, Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/a/aerocube-ocsd

[14] “Flock 1 Imaging Constellation, Planet – Flock Imaging Constellation,” eoPortal Directory, Accessed May 21, 2018. Online]. Available: https://directory.eoportal.org/web/eoportal/satellite-missions/content/-/article/flock-1-imaging-constellation#ground

[15] HawkEye 360, Inc., Pathfinder Cluster, “Exhibit 2 – FCC Form 442 (Technical Information).” Online]. Available: https://apps.fcc.gov/els/GetAtt.html?id=186546&x

[16] D. CaJacob, N. McCarthy, T. O’Shea, and R. McGwier, “Geolocation of RF emitters with a formation-flying cluster of three microsatellites,” HawkEye 360, Inc., Herndon, VA, USA, Rep. SSC16-VI-5, 2016. Online]. Available: https://digitalcommons.usu.edu/smallsat/2016/TS6NextOnPad/5/

[17] “MATLAB Interface, Integrating with STK,” STK Help. Accessed May 5, 2018. Online]. Available: http://help.agi.com/stk/index.htm#matlab/matlab.htm

[18] “Monte Carlo Simulation,” Palisade. Accessed April 20, 2018. Online]. Available: http://www.palisade.com/risk/monte_carlo_simulation.asp

[19] “Box plots,” MathWorks Documentation. Accessed May 21, 2018. Online]. Available: https://www.mathworks.com/help/stats/box-plots.html

[20] “Chapter 152: Box plots,” NCSS Statistical Software. Accessed May 21, 2018. Online]. Available: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/.../Box_Plots.pdf

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California