IMPLEMENTATION OF A CONFIGURABLE SMALL SATELLITE DESIGN AND MISSION PLANNING TOOL

BY

MICHAEL HARRIGAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Adviser:

Clinical Associate Professor Michael Lembeck

ABSTRACT

This thesis introduces an implementation of a configurable small satellite mission design tool based on a. i. Solutions' FreeFlyer™ platform. Designing and operating small satellite missions poses significant challenges due to their size and mass constraints. Small satellites have limited resources and constrained capabilities, such as attitude control, power, and communication. Optimizing the performance of these subsystems is crucial for ensuring mission success and maximizing scientific return.

To address these challenges, a comprehensive, integrated tool for small satellite mission design and analysis is needed. FreeFlyer™ provides a highly customizable, robust, and reliable framework for orbital performance simulations. A vital feature of the tool is its ability to script adaptive mission plans. The tool supports a wide range of orbit types, propagators, maneuvers, models, and output options for modeling and simulating complex space scenarios. The tool allows users to define and adjust satellite system parameters (e.g., satellite pointing configurations, attitude control devices, solar array and battery sizing, and communications link budgets) in a unified analysis and simulation environment to evaluate the possible trade-offs in performance against resource costs for various design choices.

A survey of existing programs and their shortcomings is explored. A description of an improved tool is provided, and examples of its applications demonstrate its effectiveness in optimizing and validating a previously designed small satellite mission. These case studies illustrate the tool's usefulness in sizing critical mission components and showcase its potential to expedite the small satellite mission design process.

To the family, friends, and everyone else who has supported me along my chosen path

ACKNOWLEDGMENTS

I know this road is not the one I expected to take, but now I stand at this milestone, and I am left grateful to have crossed paths with so many wonderful and talented individuals throughout my personal and academic journey before and during the construction of this thesis.

I am privileged to have had the opportunity to learn from my advisor, Dr. Michael Lembeck. Without his guidance over the past six years, I wouldn't be half the engineer I am today. He provided me with immense opportunities to develop my skills and provided the professional support that allowed me to succeed in whatever I set my mind to accomplish. I would also like to thank Dr. Jason Merret for the instruction in the classes I took with him, as they were a wonderful source of inspiration for me as an aerospace systems engineer.

I am thankful to my fellow lab members who worked in the LASSI research group. Specifically, I would like to acknowledge Eric Alpine, Rick Eason, James Helmich, Qi Lim, Chris Young, and Hongrui Zhao. My life is for the better because you are all a part of it. I would also like to thank the lab's manager, Murphy Stratton, for ensuring I always had the tools necessary to complete this research. All these listed members of LASSI and more have assisted me in some way, sort, or form in creating this thesis.

I would also like to tell my parents how much I appreciate them; without them being a core pillar of support throughout my entire life, I wouldn't be where I am today. Thank you. I love you.

TABLE OF CONTENTS

CHAPTER 1: SMALL SATELLITE MISSION DESIGN INTRODUCTION

The CubeSat standards [1] established a quarter-century ago through the collaborative efforts of Dr. Jordi Puig-Suari of California Polytechnic State University and Dr. Bob Twiggs of Stanford University [2], have been instrumental in democratizing space exploration. These specifications have enabled diverse equipment to function together by creating uniform configurations and interfaces. This standardization has been a catalyst for competitive innovation, significantly reducing the financial barriers to launching small-scale payloads into space [3], thus accelerating the pace of discovery and technological advancement using small satellites [4]. Consequently, a wide range of applications, such as biological sciences, climate change monitoring, orbital debris identification, astronomy, heliophysics investigations, communications, and new technology demonstrations, are increasingly using CubeSats [5].

While smaller in size than previous generations of satellites, CubeSat mission design remains a multidimensional problem requiring an iterative approach to configure available components into a package that satisfies stakeholder objectives subject to a set of constraints (e.g., mass, volume, power, data). Systems engineers decompose the objectives into a set of functional and performance requirements allocated to the physical components, configured to address environmental, pointing, and configuration issues, that are verified and validated prior to flight operations. Initial designs are used for feasibility studies and cost estimates, and it is, therefore, imperative that these designs are representative of a mature solution.

The design process allocates a constrained set of resources (e.g., mass, volume, and power) to the typical satellite subsystems (e.g., Structure, Thermal, Power, Communications (Comms), Attitude Control (ACS), Command and Data Handling (CDH)) to satisfy mission objectives. Users

define the desired outcome and make iterative design changes to reallocate resources, evolving the design into a viable solution. Many of these design decisions are informed by the outputs from programs used to perform trade study analysis and simulations. Many of these programs are non-interconnected, necessitating that the user reformat data from one program's output as an input for another. This time-consuming and laborious process is prone to errors at the interfaces.

This thesis focuses on the development of a configurable, integrated mission design tool that alleviates many of these challenges. The developed tool is built on top of a.i. Solutions' FreeFlyer [6]. FreeFlyer is a simulation and visualization tool with a scripting capability that can be used to evaluate candidate satellite configurations against mission objectives iteratively. For example, attitude control specifications, solar array and battery sizing, and communications link budget analysis, can be automated and rapidly evaluated in a FreeFlyer orbital mechanics simulation. The resulting design tool has proven its utility in the evaluation of Laboratory for Advanced Space Systems at Illinois (LASSI) missions, including the UIUC / Virginia Tech LAICE-F CubeSat [7].

CHAPTER 2: SURVEY OF MISSION DESIGN TOOLS

A number of small satellite design programs have been developed to assist users in developing systems that meet a set of functional and performance requirements. What follows is a survey of the existing programs and their capabilities.

The Small Satellite Concurrent Engineering Model (SmallSatCEM)

The Small Satellite Concurrent Engineering Model (SmallSatCEM) [8] is a single-user, Microsoft Excel-based [9] program designed by the Aerospace Corporation in 2001 to aid in the conceptual design phase of small satellites less than 25 kg in mass. Excel was selected as the operating platform for this program due to its customizability and ease of use [8]. The program allows the user to rapidly design and analyze potential solutions by selecting components from a database of parts and comparing their performance against the requirements identified. SmallSatCEM offers a simplified astrodynamics model sufficient for the early conceptual design phase, implemented in the Visual Basic scripting language. The inclination and eccentricity of any small satellite can be customized for use around either Earth or Mars.

This program uses defined requirements to assist in designing a configuration state for the satellite. This configuration state is then analyzed and evaluated against the original requirements. If the design fails to meet the requirements, the process is iterated (Figure 1). This analysis can be completed at both system and subsystem levels.

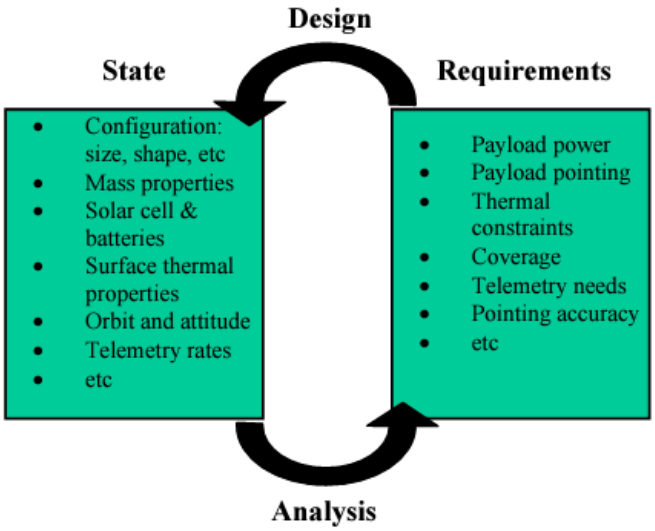

Figure 1: The iterative process of defining requirements, designing a solution, realizing the output performance state, and analyzing the results that the SmallSatCEM program utilizes. [8]

This program is organized into several worksheets: Payload and Mission, Configuration, Propulsion, Command and Data Handling, Communications, Attitude Determination and Control, Power, Thermal, Mass Properties and Distribution, Model State, Database, and Cost. The Payload and Mission worksheet is where the user can capture the mission and orbit requirements. The subsystem worksheets are where the user can compare the requirements to a drop-down menu of parts provided by the database or input the specifications of a custom component (Figure 2).

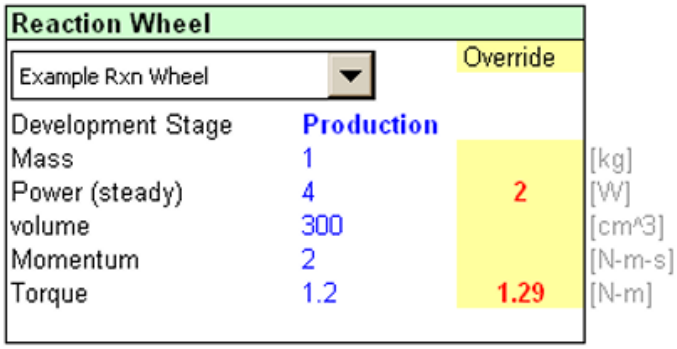

Figure 2: An example of SmallSatCEM's graphical user interface for component selection. [8]

The selected subsystem designs are compiled to form the model state worksheet. The design can then be evaluated in each of the individual subsystems' worksheets based on the satellite's simulated performance (Figure 3). This evaluation provides answers to questions such as the satellite's energy production or downlink availability over a weeklong period.

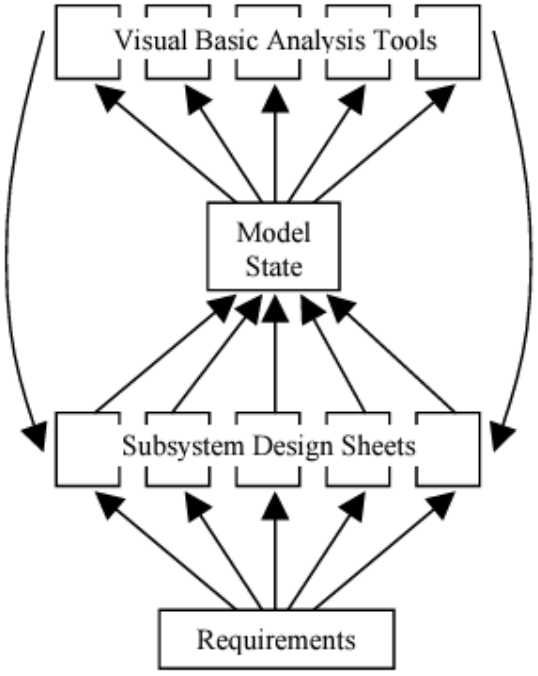

Figure 3: A flow diagram of the SmallSatCEM program depicting data flow from the requirements to the subsystem design sheets, to the model state, to the analysis programs, and then back to the subsystem design sheets. [8]

Two issues are evident with the design of the SmallSatCEM. The first is the component selection methodology. The design is based on a database of parts and maintaining an up-to-date database with available component offerings is challenging. The second issue arises from the limitations introduced by basing the program on Excel. A full astrodynamics model was never intended to be programmed into the software. This program cannot model high levels of fidelity and complex mission profiles without significant performance costs or code rewrites.

The System Engineering Design Tool (SEDT)

The System Engineering Design Tool (SEDT) [10] is a single-user statistical analysis design program designed for the initial design phases of small satellite designs. Subsystems can be rapidly selected and evaluated by referencing an internal database of components, ensuring the selections represent a viable solution based on predefined requirements. Over 200 satellite and design element references makeup the database. This program calculates the mass and volume of the designed system, the power consumption, and the cost.

The program is organized into five principal blocks: the Mission Design Block (MDB), the Systems Design Block (SDB), the Performance Verification Block (PVB), the Cost Block (CB), and the Visualization Block (VB) (Figure 4). The SDB is further broken down into the individual subsystems: Payload, Electrical Power (EP), Structure, Attitude Determination and Control (ADC), Telemetry, Tracking, and Communications (TT&C), Thermal Control (TC), and Command and Data Handling (C&DH).

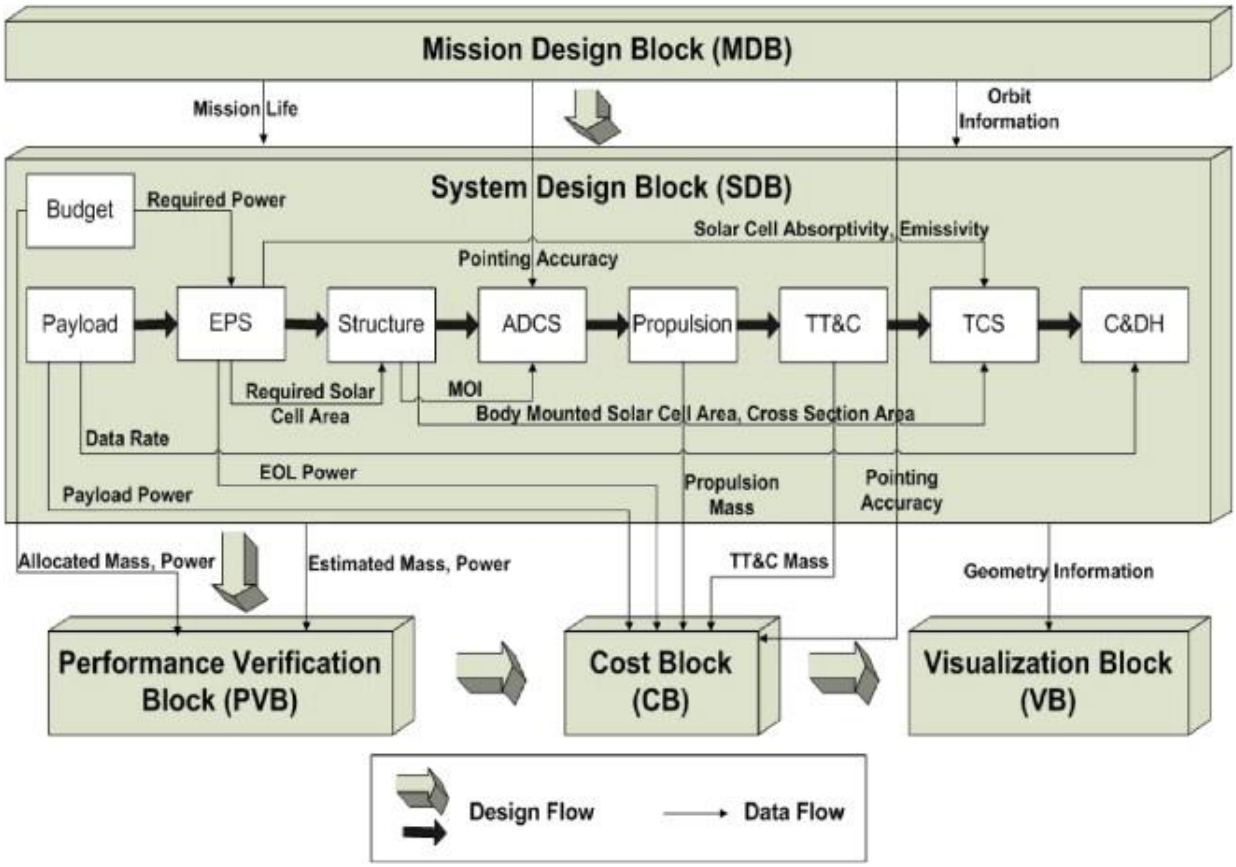

Figure 4: A block diagram showing the organization of the SEDT and the flow of information within the program. [10]

The Mission Design Block captures the mission's requirements and the predetermined orbital parameters. The System Design Block is where the component selection takes place from a database of over 200 systems. This database calculates the performance (e.g., mass, volume, power, and cost of the subsystems). The design is evaluated in the order of Payload, EP, Structure, ADC, Propulsion, TT&C, TC, and C&DH. Design decisions made in earlier subsystem blocks constrain the design of later design blocks. The PVB compares the mass and power budgets to the requirements for the small satellite. The mass of the individual components is compared to the maximum allowable mass per subsystem. The power analysis involves conducting a simple energy balance analysis. The Cost Block predicts the total cost of the satellite down to a subsystem level.

A predictive rendering of the satellite is generated based on the structure selected in the Visualization Block.

Since the functional and performance requirements dictate satellite designs, beginning with the Mission Design Block is natural. In this block, the profile of the mission is selected (e.g., remote sensing or communications), the user requirements are input (e.g., ground station locations and revisit time), the mission requirements are defined (e.g., mission lifetime, inclination, and orbit eccentricity), and the satellite capabilities are chosen (e.g., tilt angle). The mission and orbit analysis sub-block receives all this data and uses it to determine the satellite's orbital dynamics and its properties. This includes information such as the daylight vs. eclipse times, the data downlink times, the satellite's velocity, and the orbit period (Figure 5).

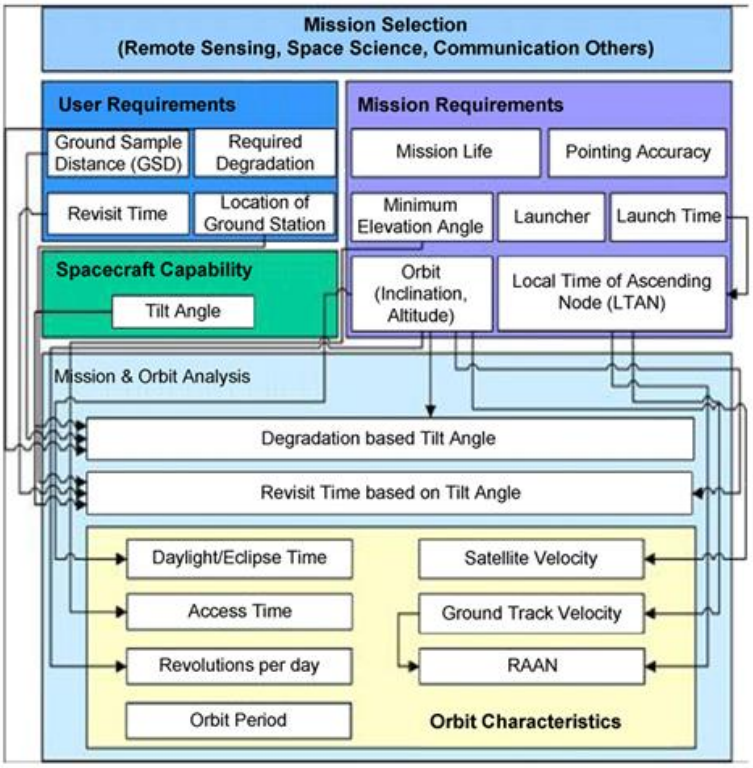

Figure 5: The flow chart provides a detailed view of the design characteristics available for customization within the SEDT's mission design block. Major components include mission selection, mission requirements, user requirements, and satellite capability. These components feed into the mission and orbit analysis block. [10]

For the System Design Block, the SEDT program utilizes a top-down design of the subsystems. By designing the EPS first, the rest of the subsystems are constrained to meet the energy generation and storage capabilities of the components selected. The second subsystem designed is the structure, which constrains the surface area, the mass (based on a given maximum density), and the volume. Following these subsystems, the ADCS, Propulsion, TT&C, TCS, and C&DH are designed (in that order), and components are selected to meet each subsystem's requirements.

Within the Performance Verification Block, the design is analyzed to ensure compliance with the constraints placed on the system's mass, volume, and power usage. Based on the results, the user may need to iterate on the components selected for the system or even revisit the design factors in the Mission Design Block. This iterative process continuously refines the system until the established requirements are met. To verify the mass and volume constraints, the ratio of the estimated value (from a parts database) is compared to the subsystems' allocations. An Energy Balance Analysis (EBA) is conducted for power verification to determine if the satellite is generating sufficient power for the amount of power that is being consumed.

The Cost Block implements the Small Satellite Cost Model (SSCM) [11] that the Aerospace Corporation created to help allocate portions of the overall hardware budget to individual subsystems. Cost analysis is another place where tradeoffs can be analyzed based on the subsystem technologies selected for more cost-sensitive missions.

The Visual Block [10] offers a flexible and user-friendly interface for early visualization of the physical configuration of the satellite hardware (Figure 6). Users can easily manipulate and reposition subsystems based on their specific requirements without the need for an independent

Computer Aided Design (CAD) software program. This flexibility empowers users to make informed design decisions, enhancing the overall design process.

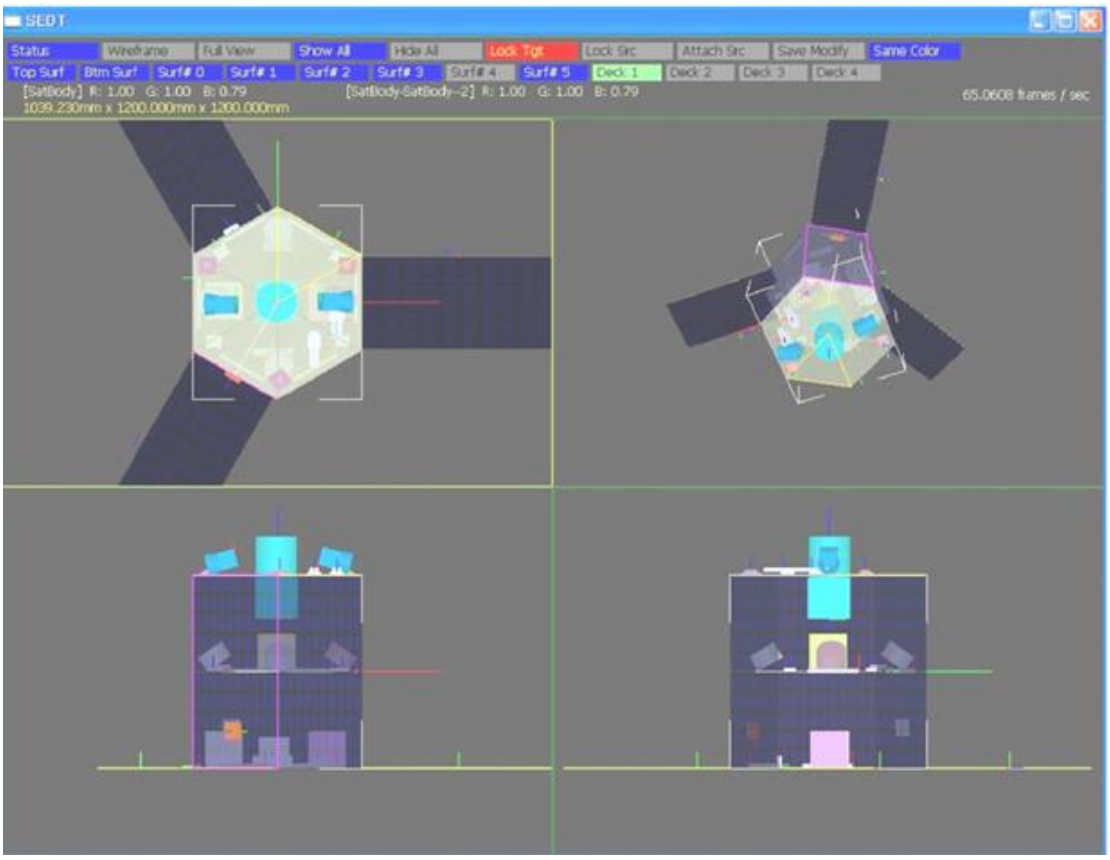

Figure 6: An example of the SEDT's rendering of a satellite's layout. The render is a multi-view 3D rendering with multiple cross-sections to allow the user to identify major subsystems and components. [10]

A Graphical User Interface (GUI) (Figure 7) makes it easy to determine where information needs to be input and where the analysis results may be found. While the database and linear performance predictive models (Figure 8) give the program a significant advantage in assessing designs, they unfortunately introduce a considerable limitation. With small satellite technologies constantly evolving (e.g., miniaturization and cost reductions), predictive models and databases quickly become outdated, resulting in future designs being based on significantly out-of-date expected performance values.

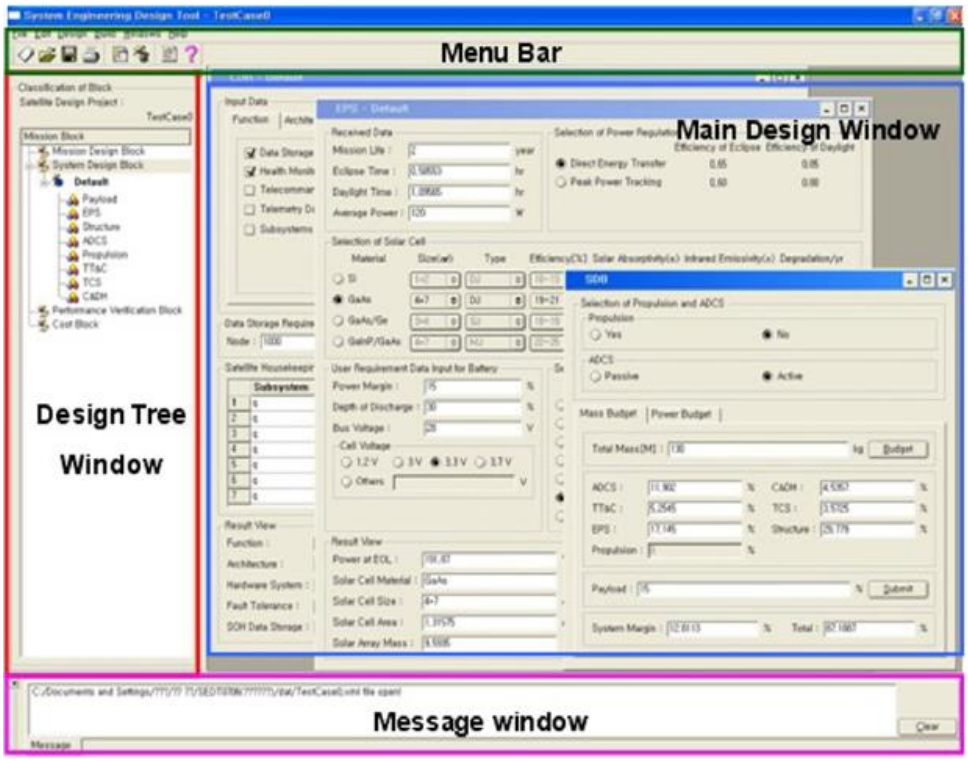

Figure 7: A screenshot of the graphical user interface of the SEDT. Major components, such as the menu bar, design tree window, main design window, and message window, are labeled. [10]

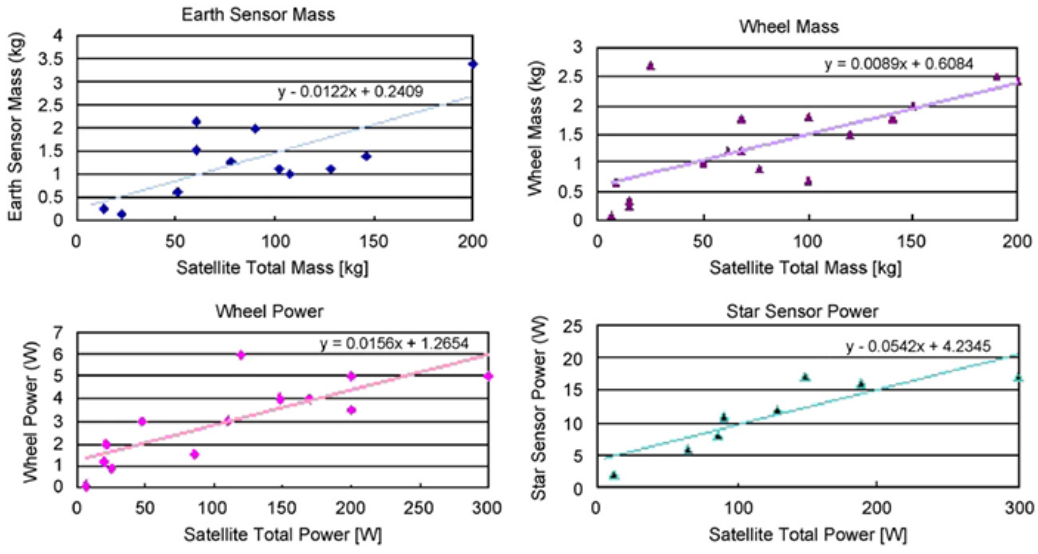

Figure 8: Four examples of the linear performance predictive models the SEDT uses to size different systems onboard the satellite. Pictured from top left to bottom right are the Earth sensor, wheel mass, wheel power, and star sensor power. [10]

While this program is very well designed and provides many nice-to-have features, such as the satellite visualizer, it does not take advantage of modern simulation programs that can provide a higher level of design analysis fidelity and more complex mission profiles. Its ability to choose only from a list of preset mission profiles limits this program.

The Spacecraft Portal for Integrated Design in Real-Time Project (SPIDR)

The Spacecraft Portal for Integrated Design in Real-time (SPIDR) project [12] is a satellite design program developed in 2009 at the USC Information Sciences Institute. USC identified three functions that are critical to providing an integrated solution for an early-stage satellite design:

- The design equations and models used to calculate the necessary performance of components need to be incorporated.
- An up-to-date database of procurable components needs to be integrated for users to reference.
- Design optimization capabilities through different configuration iterations.

The SPIDR program uses AGI's Satellite Tool Kit [13] to implement the orbital dynamics equations and models. The program [12] also has a parts database that the users can reference and a standardized set of outputs that come from Hardware-in-the-Loop (HITL) testing and simulation. This allows for iterative optimization of solutions based on user-defined Measures of Effectiveness (MOEs).

The design optimization engine of SPIDR takes two different kinds of inputs (Figure 9). The database of components and the optimization rules serve as the first input. The mission requirements and the MOEs (e.g., cost, mass, power consumption, or volume) are the second input required. From these inputs, the number of degrees of freedom is determined by the optimization

engine. These degrees of freedom include technical values, such as mass, and non-technical values, such as integration difficulty, based on a user-generated value weighting system.

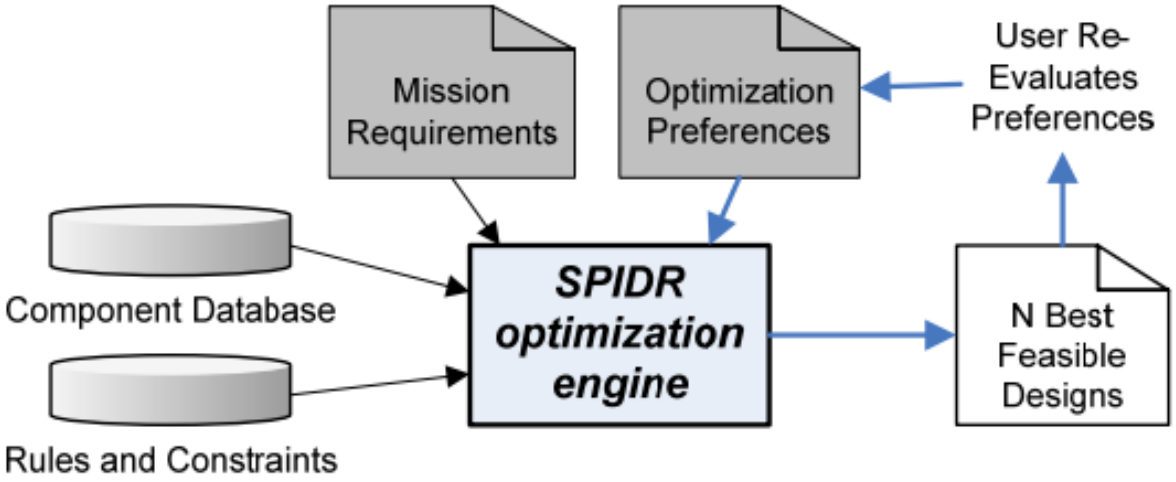

Figure 9: Flow chart depicting the optimization process of the SPIDR program. The user defines the mission requirements and optimization preferences, combined with the component database and the rules and constraints in the optimization engine, N feasible designs are created to be evaluated by the users. [12]

The SPIDR optimization engine uses constraint reasoning techniques to solve for multiple "optimal" solutions based on the user inputs. After each iteration, the user can redefine their preferences or start over by changing the mission requirements or updating the component database and rules. The development of rules and constraints on a mission results in an appropriate design, but over or under-development of these rules could lead to an over-constrained or under constrained system instead, preventing the optimization engine from developing a useful solution.

One benefit of this type of design is that the subsystems can be designed in any order based on the individual mission profile. The optimization engine can optimize around mission profiles that require a robust power subsystem or attitude control subsystem and prioritize their design first so as not to constrain them any more than they already are. Another benefit of this program is the

user-friendly GUI (Figure 10) that provides users with an easily readable and development-friendly environment. Rules and requirements can be inputted using a sliding bar, which allows for quick iterative designs and testing.

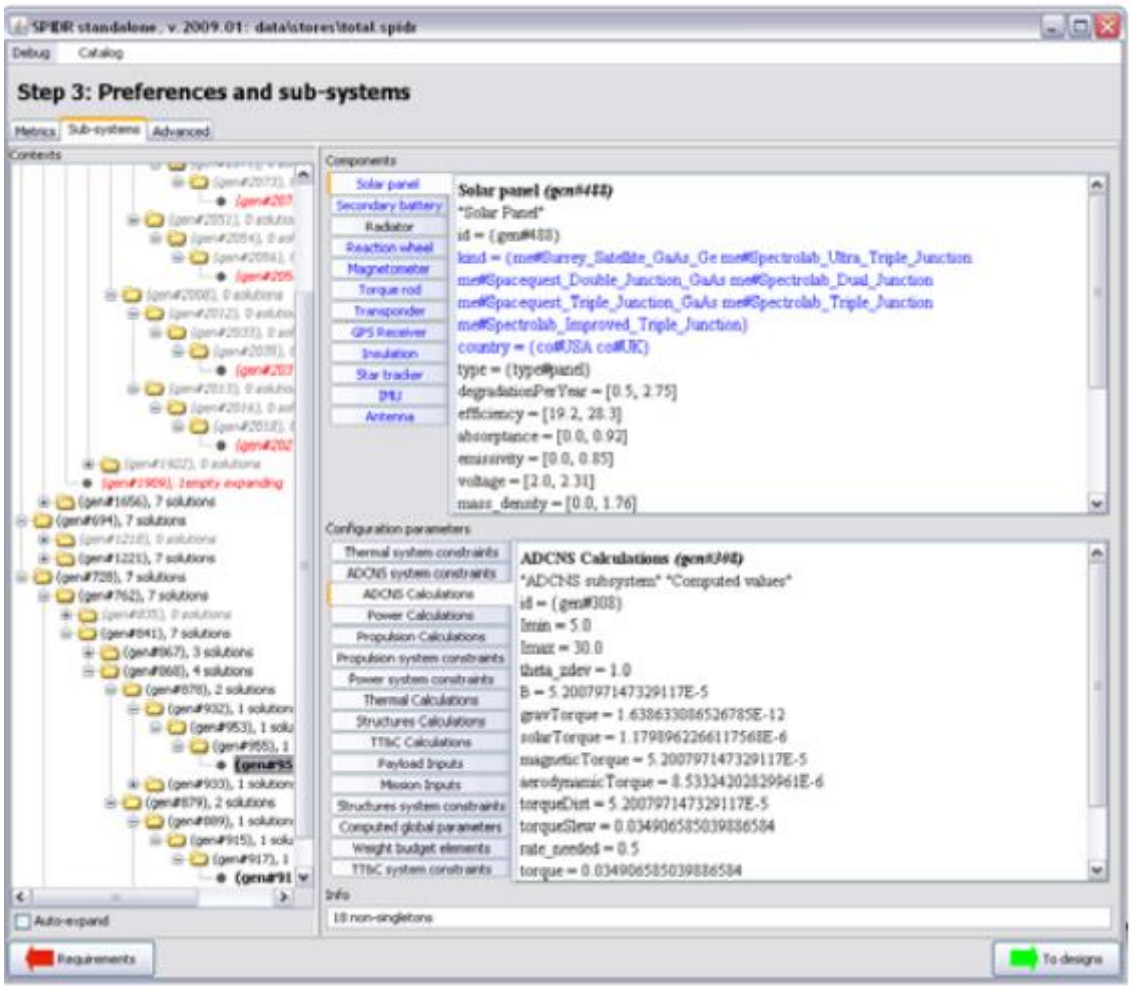

Figure 10: A screenshot of SPIDR's GUI. The design tree windows are on the left side, the components are on the upper right, and the configuration parameters are on the bottom right. Buttons to advance to the produced designs or back to the requirements are present on the bottom left and bottom right, respectively. [12]

Once the analysis has been completed, the user can explore the subsystem tabs to see which components are compatible with the design and the rationale for why these parts were selected. A single "most optimal" design will also be displayed based on the MOE selected.

One of the disadvantages of this design program is that it requires STK [13] as secondary software. While the use of orbital dynamics software will benefit the accuracy of the analysis, the configuration of the two programs to connect with one another is an inconvenience. Another downside to a program like SPIDR [12] is database maintenance. Without updating, the database will become out of date. If satellites are being designed with components that are no longer on the market or no longer optimal for the design, then the program becomes significantly less useful to the end user. Another disadvantage is that the program does not allow for custom modifications to the models imported from STK.

System Engineering Module (SEM)

The purpose of the System Engineering Module (SEM) [14] is to reduce the number of industry professionals, state-of-the-art equipment, and software programs required to complete the conceptual design (Phase A) of a new satellite. The program is designed to be user-friendly, accessible, and inexpensive to operate to allow both industry professionals and academics to take advantage of the solution developed. Developed in C++, the program utilizes the preexisting Space Trajectory Analysis (STA) program [15] offered by the European Space Agency (ESA) to handle the program's astrodynamics, orbit propagation, and simulation needs. The properties of defined satellites within STA are passed to the SEM program to allow for assessment of the design (Figure 11) [14].

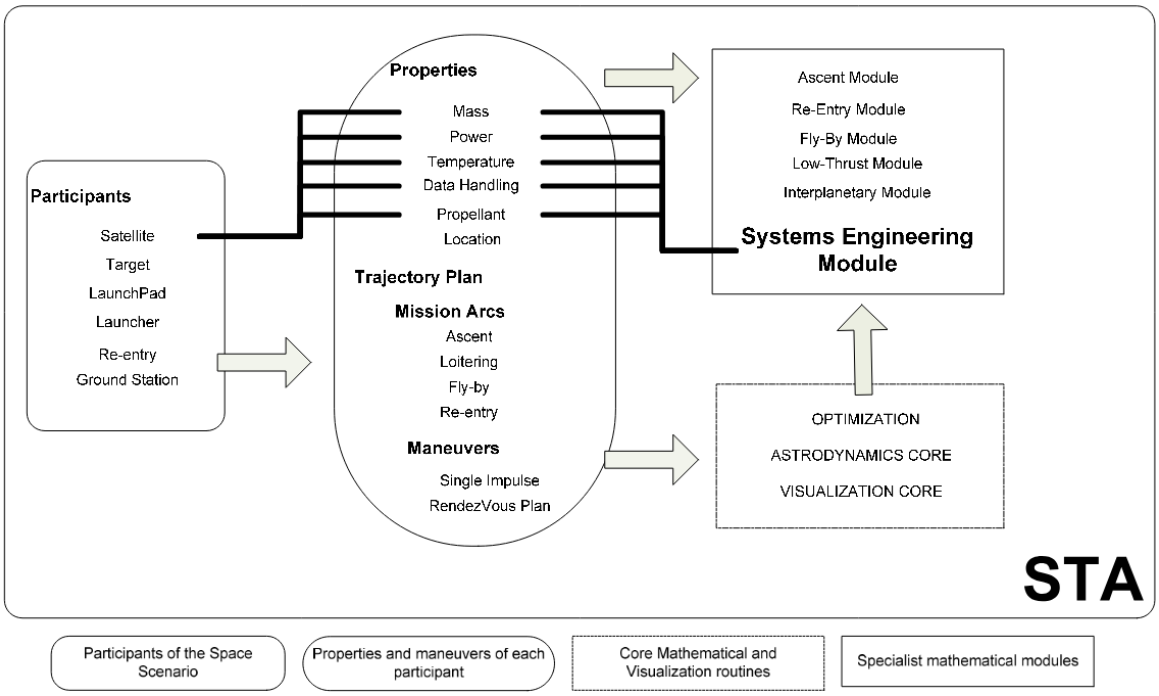

Figure 11: Program integration diagram between SEM and STA. The diagram depicts how the properties of the satellite are passed to the SEM from STA. [14]

The SEM models the power, attitude determination and control systems, communication, propulsion, thermal, structure, and data handling subsystems. Performance evaluations are based on the design's mass, power, thermal energy, data, communication, and propellent budgets. The program is meant to be used alongside additional forms of analysis, such as cost and risk identification. Once outputs are received from the initial satellite definition, the design can be iterated upon and updated to meet the mission requirements (Figure 12).

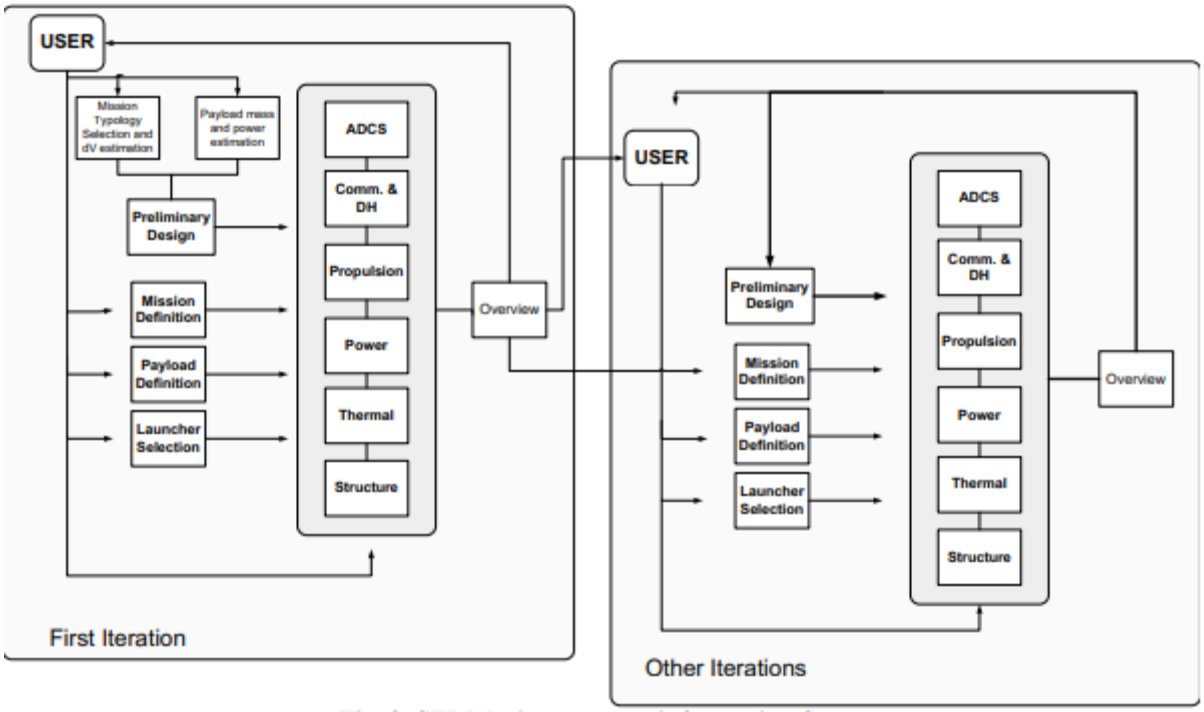

Figure 12: The iterative process of refining the preliminary design, mission definition, payload definition, and launcher selection that the user undertakes when using SEM. [14]

The user interacts with the SEM through a custom GUI (Figure 13), in which the user inputs the required data, selects the inputs, and views the results of the analysis. The user then defined the external constraints on the satellite, such as the space environment, the ground stations, and other satellites that will be interacted with. The payload and mission definition sections are responsible for differentiating missions from one another. The SEM database is populated with the parameters of components and can be modified to meet the user's needs (e.g., the efficiency of a solar cell). The SEM tables on the other hand, contain values used in the analysis calculations sourced from averaged analysis or trade studies from past missions (e.g., solar cell degradation rate) and are not editable. Values from calculations are stored in the Data Sheet, and the calculations and analysis are completed in the mathematical routines block. Afterwards, the outputs are presented to the user. Figure 14 displays the complete architecture.

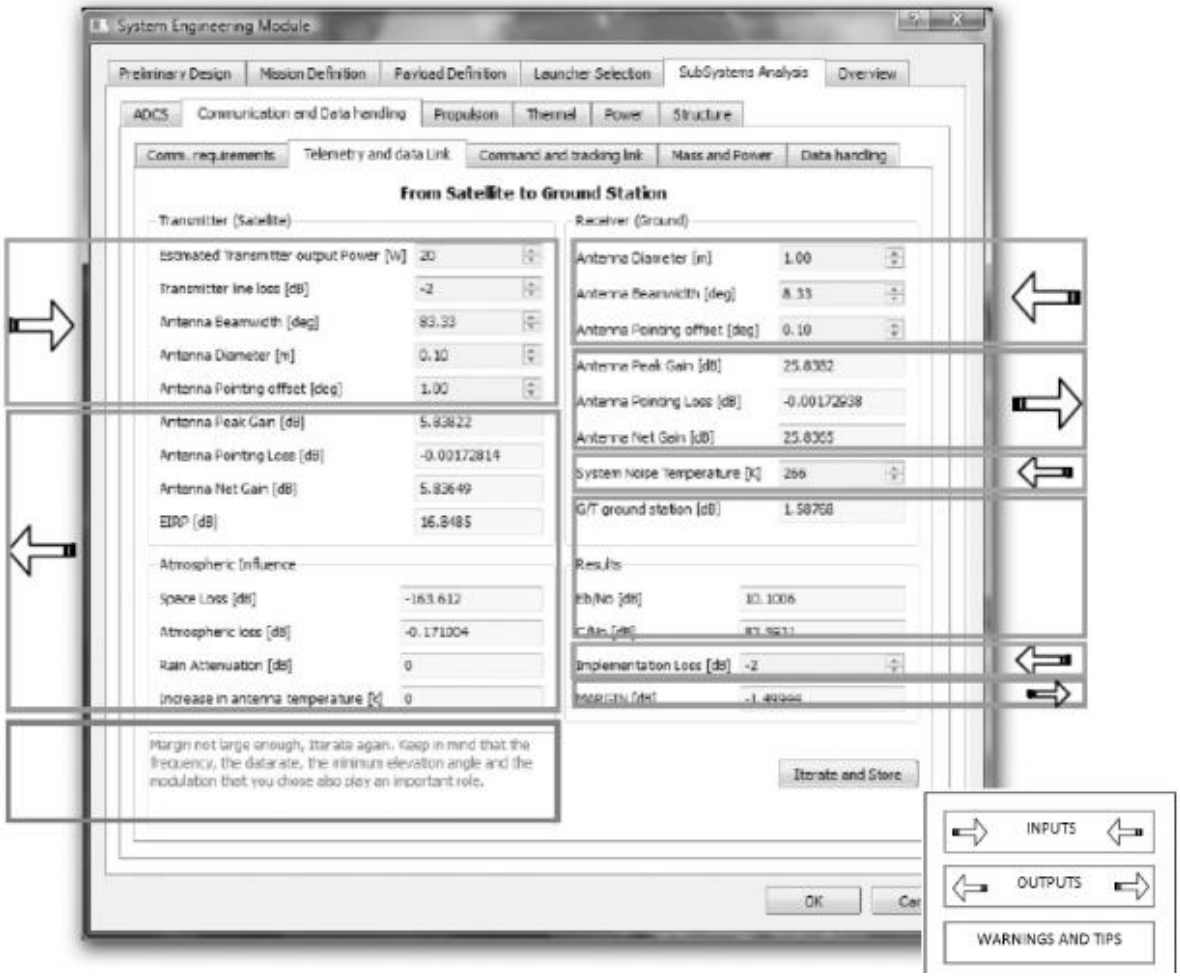

Figure 13: The graphical user interface for the SEM program. The program's inputs are at the top of the window, and the outputs are below. Each tab of the program is a different function, subsystem, or piece of the design and analysis. [14]

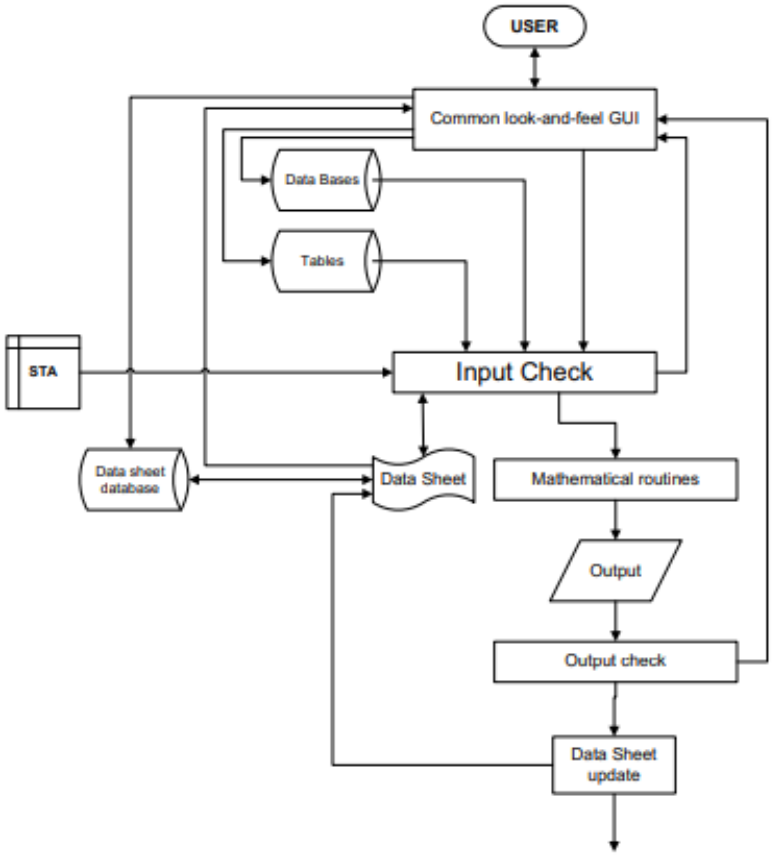

Figure 14: The SEM's programming architecture. The user who interacts with the GUI fills in the subsystem tables. Inputs from STA and all user inputs are checked for validity. These input are then passed through the mathematical routines and produce an output design based on the system's needs and the parts database's datasheets. [14]

The SEM is the most complete version of a preliminary design program among those surveyed. The program features a GUI, a parts database, a full suite of subsystems that are subject to analysis, and operates on open-source software. The parts database is still subject to becoming out of date, and having to connect the program to secondary software is an annoyance. Still, the main disadvantage of this program is its questionable accuracy. During the program's verification, an 18% under-prediction difference in predicted mass appeared when comparing identical input parameters between the SEM analysis and the CDF report for the Hyper mission [16]. A similar (19%) overestimate also appeared in the power subsystem design. Thus, while the program's feature set is exemplary, its accuracy is not.

Design Tool Summary

When reviewing the four conceptual design assistance programs, five characteristics were identified that can be used to compare them: the user interface, the use of databases, the use of separate orbital dynamics software, the subsystems modeled, and the accuracy of the programs.

The first feature that all these programs have in common is their use of an user interface. SmallSatCEM [8] uses Excel's Visual Basic for a customized interface. SEDT [10] uses a traditional Windows-like interface with a design tree to track decomposition. SPIDR [12] uses a Java-based interface with a similar design tree. SEM [14] uses a more modern tab-based design for the graphical interface. Of these different programs, it was clear that the most straightforward programs to navigate were the ones that separated the inputs and outputs into sections. The more modern designs of the SPIDR [12] and SEM [14] programs provide a better first impression and give the user a design they are likely already familiar with. The more mature designs of the SmallSatCEM [8] and SEDT [10] are less daunting to navigate after becoming familiar with the programs but provide a poor first impression because of their learning curves.

The second characteristic evaluated was the use of a parts database. Every evaluated program depends on a parts database to extract component performance parameters from. Maintaining these databases is expensive, and open-source collections of components exist from which users can pull parts' performance information (e.g., NASA's State-of-the-Art Report for Small Satellite Components [17]). Accordingly, maintaining a parts database for the conceptual design program's operation is viewed as a drawback.

The third characteristic is the orbital dynamics software that each program uses. SmallSatCEM [8] programs all of the orbital dynamics calculations in Excel [9]. SEDT [10] integrates its simplified orbital dynamics simulations into the standalone program. SPIDR [12]

utilizes STK [13] for its orbital analysis and simulation. SEM [14] uses STA [15] for the orbital analysis piece. Using an established orbital dynamics simulator significantly increases accuracy and the mission profile modeling capabilities compared to the mission design programs which do not take advantage of one.

A fourth characteristic is the satellite subsystems that are modeled in each program. The subsystems modeled by all of these programs include ACS, communications, power, structures, thermal, and propulsion. All programs except for SPIDR [12] modeled the data handling subsystem. The programs all feature a relatively complete analysis of the typical subsystems.

The fifth characteristic evaluated is accuracy. Any results produced by the design programs are expected to be of sufficient accuracy for a conceptual design. Unfortunately, the accuracy of the SEM program [14] is lower than that of the other programs, with 18% underpredictions for mass and 19% overestimates in the power requirement when compared against the Hyper mission's CDF report [16]. The other programs were sufficiently accurate for this level of design.

Desirable Features in a Design Tool

Analyzing these previous programs allows for the identification of features that a future conceptual design small satellite design tool should and should not include.

The first desirable feature is an intuitive user interface. Each of the surveyed programs implemented this to varying degrees of success. Some factors that impact the intuitiveness of the user interface are the consistency of the design, the clarity of buttons, menus, the use of familiar elements (e.g., a floppy disk icon to represent saving), and simple in-program navigation, which minimizes the training required to successfully operate the tool. Having an intuitive user interface will reduce the time needed to produce a result and increase the user's desire to use the tool.

The second desirable feature is to base the design tool around an astrodynamics simulator. This feature is necessary to ensure that the astrodynamics accuracy is sufficient for complex mission profiles. There are many simulation tools that provide this functionality, such as STK [13], GMAT [18], and Orekit [19]. A desirable feature of the astrodynamics simulator is a user-friendly scripting environment to develop the subsystem analysis models in. This allows all the analysis and simulation work to be conducted within a single (orbital dynamics) software package instead of setting up a connection between the orbital dynamics software and the subsystem analysis software, which can be unnecessarily challenging for users.

The third desirable feature of the conceptual satellite design tool is the performance modeling of the attitude control, power, data handling, communications, and structure subsystems. These are the most important subsystems for the conceptual design in determining the mission's feasibility. This tool does not need to complete accurate thermal modeling or CAD renderings, as they are reliant on the determination of the interior layout, as well as the specification of materials and mounting mechanisms, and therefore do not need to be completed within this tool.

The fourth feature considered important is output generation. Several different types of outputs need to be generated for the user to have an accurate understanding of the performance of the satellite and verify the proper execution of the mission plan. A modifiable 2D and 3D in-situ rendering of the satellite helps in verifying the mission plan's execution. Customizable tables and plots are needed to output the performance of the satellite so the user can use them to analyze and validate the design.

The fifth feature of a useful tool is accuracy. Comparisons of the systems sized using this tool will be compared to results previously calculated for existing designs.

CHAPTER 3: MISSION DESIGN METHODS

Many technological, logistical, and regulatory constraints challenge small satellite mission design. The systems engineering process (Figure 15) helps to satisfy some of these challenges through well-established principles and practices. The process of iteratively taking a mission objective, deriving a set of functional requirements, assigning the functions to physical elements, and evaluating the resulting system performance results in a baseline solution from which higher-fidelity solutions can be iteratively evolved as the design matures.

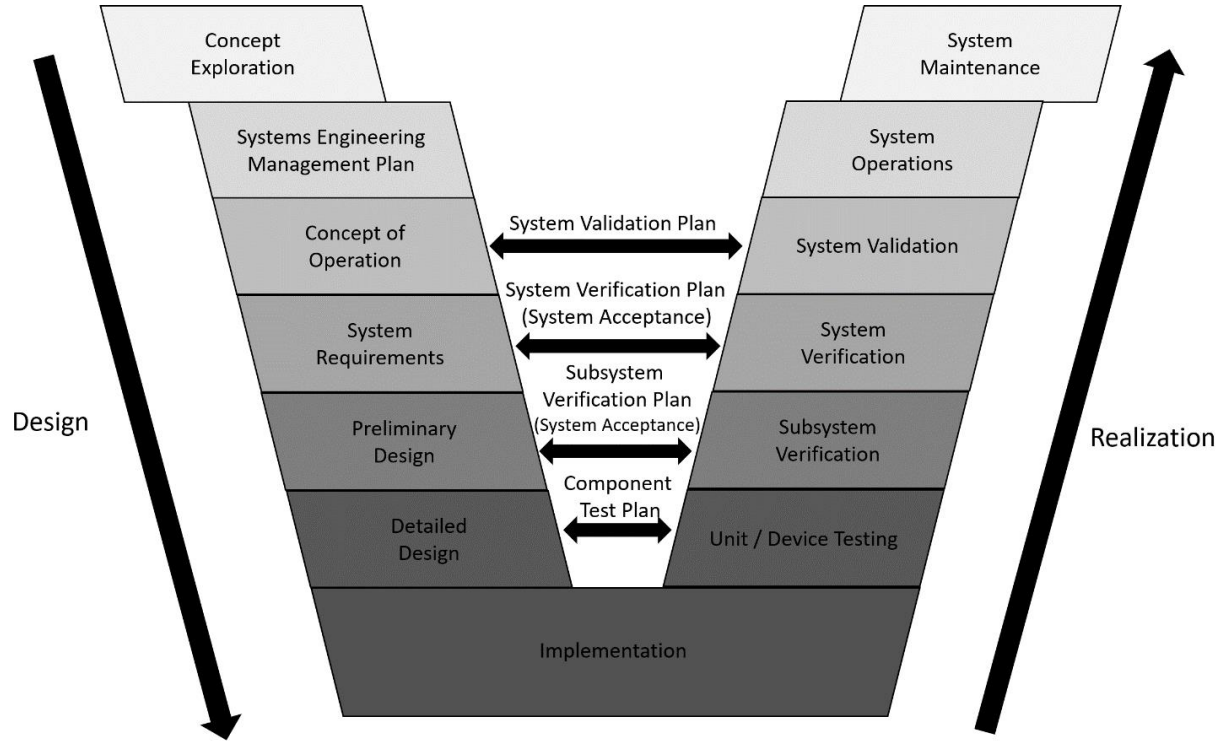

Figure 15: The systems engineering "V" depicts the process from the beginning of design to the completion of realization.

Satellite Systems Engineering

The systems engineering process starts with the formulation of functional and performance requirements to satisfy a set of mission objectives. Trade studies are then performed to evaluate the various design options for implementing a given function. Cost constraints are also used to make selections for the baseline solution. The process produces multiple supporting documents to document the design and its evolution, including a Systems Requirements Document (SRD), a Systems Engineering Management Plan (SEMP), a Statement of Work (SoW), and a Flight Assurance Plan (FAP). Figure 16 depicts the journey from the initial mission objective to the final solution.

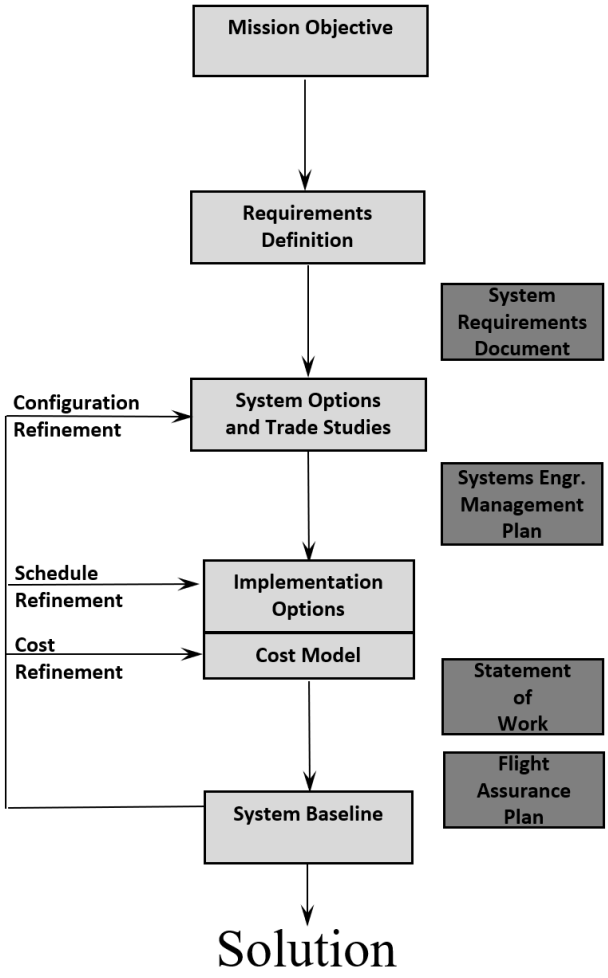

Figure 16: A flow chart of the iterative process of taking the mission objectives and achieving a baseline solution. The right-most side of the figure provides additional information on the supporting documentation required at each stage of the process.

An orbital and attitude simulation tool can expedite the development of a satellite concept into a mature design. The availability of a scripting language for implementing ordered processes allows multi-dimensional sizing calculations to be carried out for the most crucial satellite subsystems and components. These calculations allow the user to evaluate the performance of a particular design against the requirements. If a performance shortfall is revealed, the configuration is refined to address the shortfall, and sizing calculations are updated across all subsystems. This process is repeated until the solution converges.

The starting point for such an exercise is to define three budget allocations for the satellite: physical properties (mass and volume), power, and data. Mass is initially allocated to the subsystems to provide guidance in the selection of components that provide the required functionality to meet the stated mission objectives. These numbers can directly impact the launch vehicle's performance in placing the satellite at a particular operational altitude in orbit. Altitude, in turn, is tightly coupled to other performance parameters, such as mission duration, daylight and eclipse times, communication pass duration, and payload data collection.

These initial equipment selections also have associated power requirements that are added together to create a power budget. Consideration is given to the satellite's various modes of operation throughout its orbit, including during orbital eclipses, leading to a specification for power-generating solar arrays and energy-storage batteries. Energy consumed from the batteries during eclipse must be replaced by power generated for battery charging while on the daylight side of the orbit.

Attitude control and maneuvering requirements must also be addressed. The satellite's attitude must be controlled to point the payload and the solar panels at specific targets. The satellite must also counteract any external environmental forces acting on it. Determining the magnitude of these forces enables orbital lifetime computations and provides information for later attitude control system component sizing.

Payload data and satellite telemetry are stored onboard the satellite until it is downlinked to the ground over the available communications path(s). A data budget identifies the data sources, collection rates, and the size of each collection. Operational calculations determine the rate at which onboard memory is filled and depleted by each communication pass with the ground.

The rate at which data is transmitted to the ground is found through a link budget calculation. The communications channel (e.g., radio, laser) bandwidth and other pertinent parameters are used to compute how fast the data may be transmitted, which is tied to how much memory is used and then freed up after the communications pass is completed.

From this discussion, it is apparent that many of these parameters are interrelated. Having a simulation tool that can take these multi-dimensional relationships into account allows the designer to quickly pick a set of parameters to evaluate at each step in the iterative design process until the design converges. With this process in mind, the requirements for a productive simulation tool can now be enumerated.

Design Tool Requirements

With the objective of creating a tool to assist in the conceptual design of small satellites and incorporating the functions identified previously, the following requirements were identified for a conceptual design tool:

1. The tool shall have an intuitive user interface. This requirement allows the tool to be used more efficiently by the end-user. An intuitive interface reduces the likelihood of user-generated errors. This requirement can be achieved by clearly separating input and outputs during the design process and ensuring all menus, buttons, and icons are apparent to the user. In addition, providing documentation on how to operate the software is necessary.
2. The tool shall integrate with an established orbital dynamics software. This requirement ensures sufficient accuracy in the resulting analysis (at least three

orders of magnitude more accurate than two-body basic analysis) and allows for the evaluation of complicated mission profiles.

3. The tool shall support a custom scripting environment. This requirement allows users to modify any pre-designed models and generate custom mission profiles. To meet this requirement, orbital dynamics software is selected that allows for custom scripts to be processed.
4. The tool shall model ACS, Communications, Data Handling, Power, and Structures. This requirement establishes the subsystems to be modeled in this tool. By implementing the models and equations that govern the sizing and design of these subsystems in the custom scripting environment, these subsystems can have their performance analyzed based on their state in the orbital dynamics software.
5. The tool shall generate a variety of outputs that define the performance of the satellite. This requirement is necessary so that the user can iterate on their design based on the previous iteration's performance results. A few different types of outputs are needed, including 3D modeling capabilities, plotting capabilities, table generation, and terminal output.

One available tool that satisfies these essential design tool requirements is a.i. Solutions' FreeFlyer.

FreeFlyer-Based Design Tool

a. i. Solutions' FreeFlyer tool [6] performs detailed mission analysis, satellite trajectory analysis, and operations planning. As a simulation, scripting, and visualization software, FreeFlyer

can support all phases of a mission's development, from conceptual design to operations. FreeFlyer's simulation capabilities have been independently verified and validated [20]. FreeFlyer [6] is used for analysis and operations by NASA, commercial satellite providers, and industry professionals for ground, Low Earth Orbit (LEO), Medium Earth Orbit (MEO), Geosynchronous Earth Orbit (GEO), and beyond.

Mission plans capture a set of scripted instructions that a user generates to define the operations of a simulated satellite. These actions can be anything from maneuvering to assessing power use to a communications session. FreeFlyer's most powerful feature is this freeform scripting capability, as it allows the user to use predefined functions to undertake complex analysis and post-processing on any generated data. The scripting language is based on Microsoft's Visual Basic language constructs. Data can be output and saved using a variety of methods and visualizations (e.g., plots, tables, etc.). The simulation can also be visualized in 2D or 3D renderings (Figure 17).

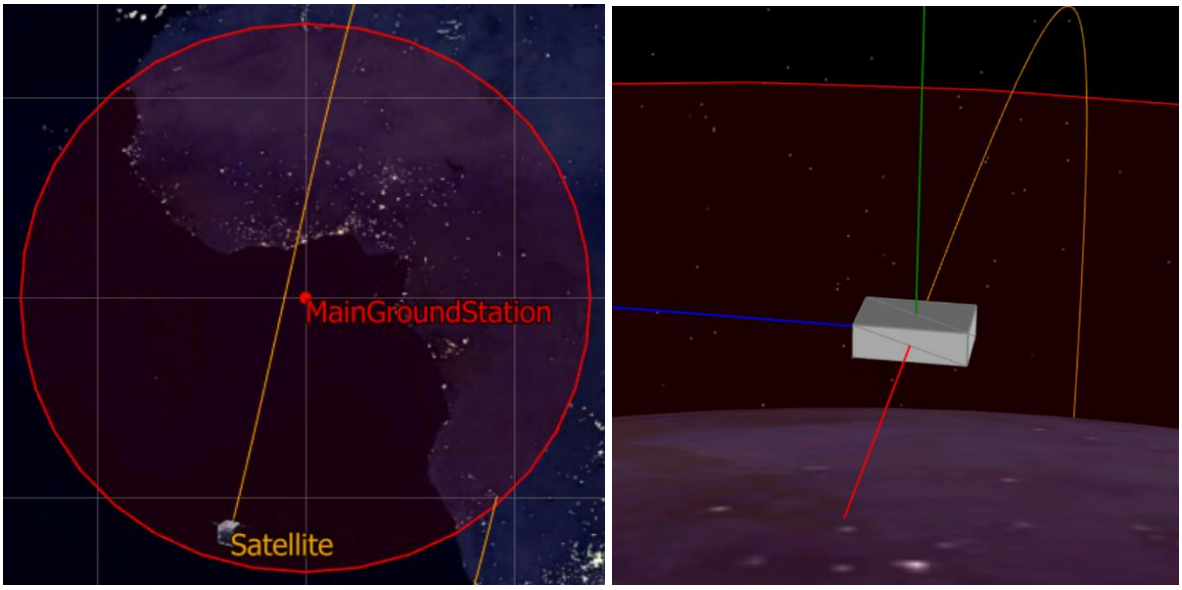

Figure 17: A 2D (left) and a 3D (right) example of the visualization FreeFlyer provides.

FreeFlyer performs the orbital dynamics and satellite attitude calculations and, through its scripting language, the iterative sizing calculations for the power, data, and communications subsystems. The GUI is intuitive and allows for users to easily navigate through the initialization and the results.

To set up and run a satellite simulation in FreeFlyer, the user first defines the mission profile. This includes all operation modes, as well as the conditions under which these modes are active. The user also needs to define any specific targets that the satellite needs to point to during the mission. In this step, the orbital configurations for the mission (e.g., orbiting body, altitude, inclination, RAAN, etc.) are set.

Next, a comprehensive model of the satellite's geometric platform is created. This is used by FreeFlyer to accurately model the external forces on the satellite.

The third step is to model the power subsystem. The incidence angle of light from celestial bodies impacting the solar arrays can be modeled using vector math by considering the satellite's attitude and orbital position. Power usage is tracked by assessing the power state of the various components. The battery's energy storage level can also be modeled by subtracting the integrated power usage and adding energy back into it while charging in daylight.

The fourth step is to model the data handling and communications subsystems. Satellite telemetry and payload data generation rates are integrated in the satellite's various modes of operation and subtracted from the total amount of available non-volatile memory. A satellite-to-ground-station contact analysis is used to determine when a downlink opportunity is available. Downlink bandwidth calculations determine if there is sufficient margin to establish a link and at what rate data may be downlinked. Onboard memory storage increases during a downlink session as the ground station receives the data.

The final step is to capture the calculated metrics and make them available in a user-digestible format. In Figure 18, the top and bottom left windows illustrate FreeFlyer's plotting capabilities. The top right window shows a 3D visualization of the satellite's orbit around the Earth. The bottom right window reports the communication parameters in a table.

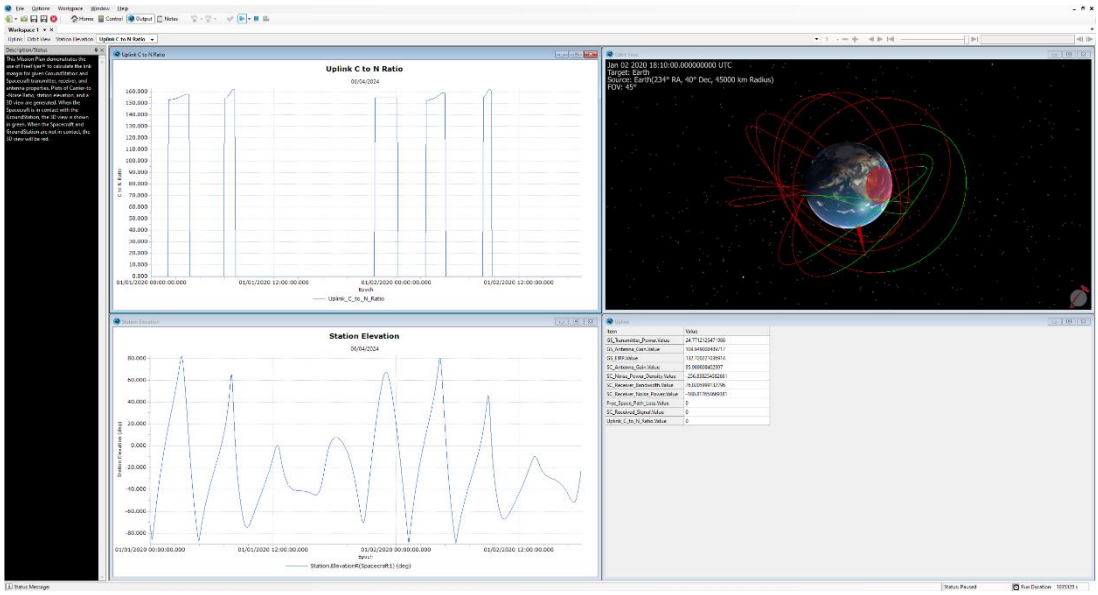

Figure 18: An example output from a satellite's analysis in FreeFlyer.

Besides making the subsystem performance outputs available for viewing, a 3D model of the satellite can be constructed out of 2D plate objects. Figure 19 depicts a 6U satellite with deployable solar arrays that utilizes this functionality.

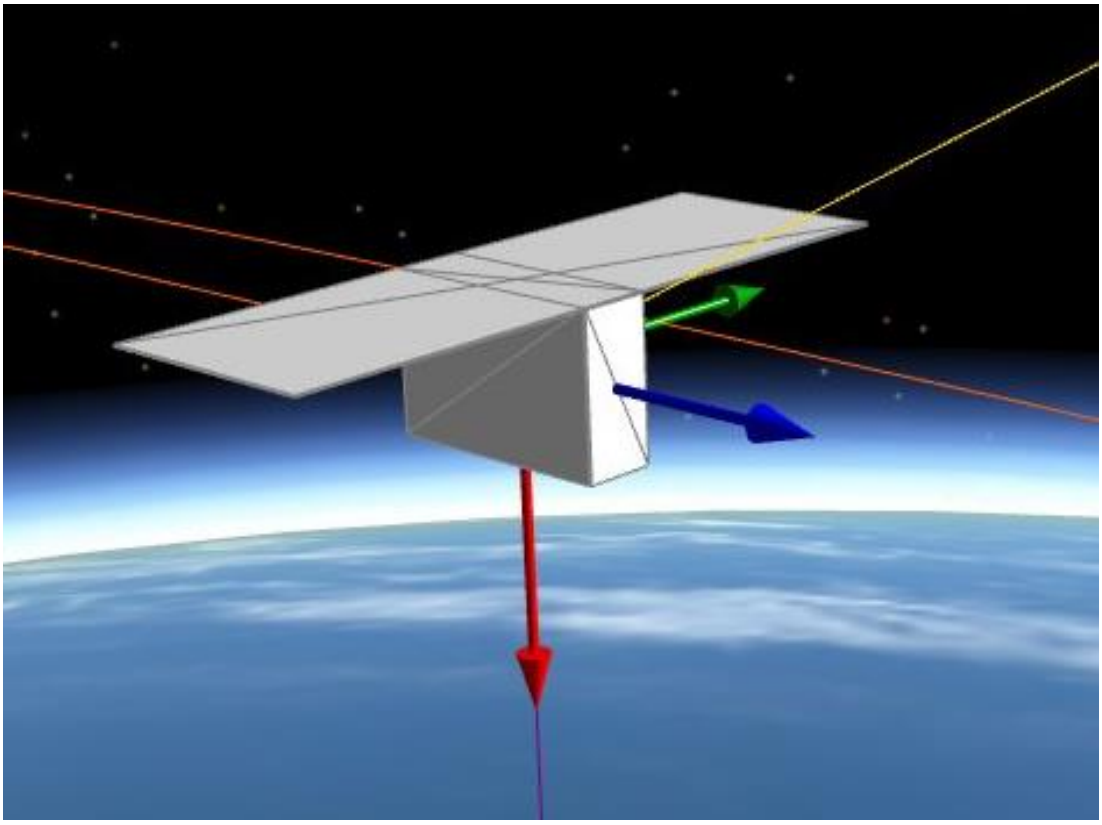

Figure 19: A screenshot of the visualization for the 3D satellite planform view. The colored arrows identify the three principal axes of the configuration.

The few examples presented touch on FreeFlyer's versatility in modeling and evaluating satellite designs in a quick and efficient manner. In the next chapter, the internal mechanics of the available modeling components for satellite design within FreeFlyer are examined.

CHAPTER 4: FREEFLYER CAPABILITIES

FreeFlyer's capabilities for modeling orbit propagation, as well as custom scripting functionality, satisfy the previously established requirements for a small satellite design tool. This custom scripting capability allows for the modeling of the operational orbit, power generation and energy storage, data handling, and communications with the ground. The sections that follow provide a description of the functionality and implementation of both the preexisting and custom functions utilized.

Orbit Models and Propagator

In FreeFlyer [6], orbit models and propagators are essential for accurately simulating the satellite's orbital dynamics. Accurate and timely evaluation requirements must be balanced to obtain reasonable results in a reasonable amount of time. The Runge Kutta4(5) propagator satisfies this requirement. The actual time it takes to move the simulation forward depends on the propagator used. Choosing a step size to evaluate the simulation is also a balancing act. Selecting a large step size decreases the simulation's accuracy, but choosing a smaller time step results in a simulation that takes longer to run. After some experimentation, a step size of six seconds was chosen to allow for sufficient data logging frequency and mode-switching ability. This step size is used throughout the presented analysis.

Equation of Motion

FreeFlyer propagates the satellite's dynamics by integrating the rigid body equations of motion. Knowing the environmental and control torques being applied to a satellite, the equations of motion may be integrated to determine the resulting rotational accelerations that result.

Expanding Euler's equations for a rotating rigid body reveals the components along the three principal axes of the body-fixed coordinate system, commonly referred to as the x, y, and z axes (Figure 20). These axes correspond to the body's principal moments of inertia, I_x, I_y, and I_z, respectively [21].

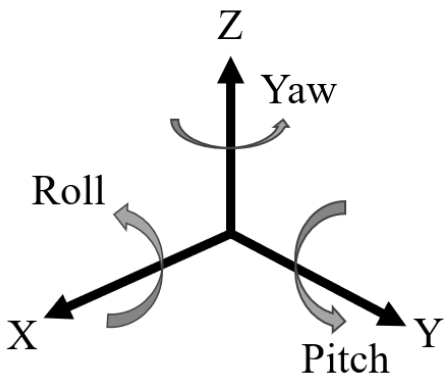

Figure 20: The three principal axes and the names of the rotations around these axes.

The expanded forms of Euler's equations are (Equation 1-3):

$$M_x = I_x\dot{\omega}_x + \left(\omega_y\omega_z\right)\cdot\left(I_z - I_y\right) \quad (1)$$

$$M_y = I_y\dot{\omega}_y + \left(\omega_x\omega_z\right)\cdot\left(I_x - I_z\right) \quad (2)$$

$$M_z = I_z\dot{\omega}_z + \left(\omega_x\omega_y\right)\cdot\left(I_y - I_x\right) \quad (3)$$

where M_x, M_y, and M_z are the components of the external and applied disturbance torques about the x, y, and z axes, $\dot{\omega}_x$, $\dot{\omega}_y$, and $\dot{\omega}_z$ are the components of the angular acceleration vector about the x, y, and z axes, ω_x, ω_y, and ω_z are the components of the angular velocity vector about the x, y, and z axes, and I_{xx}, I_{yy}, and I_{zz} are the moments of inertia about the x, y, and z axes. These equations account for the coupling between the rotational motions about the three axes due to the gyroscopic effects, represented by the cross-product terms (e.g., $\omega_y\omega_z$) and the differences in the moments of inertia.

To numerically integrate these equations, FreeFlyer [6] discretizes the simulation time into increments and then applies the Runge-Kutta 4(5) method numerical integration scheme to update the body's angular velocity and orientation at each time step.

A quaternion [22] representation of attitude is used. The angular velocity vector obtained at each time step is then integrated as follows (Equation 4-5):

$$\dot{q} = \frac{1}{2} q\Omega \tag{4}$$

$$\Omega = 0 + \frac{\omega_x}{2} i + \frac{\omega_y}{2} j + \frac{\omega_z}{2} k \tag{5}$$

The angular velocities for each axis *n* are represented by ω_n. The quaternion derivative $\dot{q}$ can be calculated using the current quaternion q and the angular velocity quaternion Ω. Given their difficulty being readily human interpretable, the quaternions are converted to Euler angles in a 3-1-2 rotation set. The first step in converting from the quaternion to Euler angles is to convert the quaternion into a 3x3 rotation matrix. Matrix *R* in Equation 6 provides the equation for this.

$$R = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix} \tag{6}$$

From here, the rotation sequence will need to be mapped out using the corresponding Euler angles, with ψ representing rotation around the Z axis, θ representing rotation around the X axis, and ϕ representing rotation around the Y axis. Equation 7-8 represents the Direction Cosine Matrix (*DCM*).

$$DCM = R_2(\psi)R_1(\theta)R_3(\phi) = \begin{bmatrix} \cos(\psi) & 0 & -\sin(\psi) \\ 0 & 1 & 0 \\ \sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$R_2(\psi)R_1(\theta)R_3(\phi) =$

$$\begin{bmatrix} \cos(\psi)\cos(\phi) - \sin(\psi)\sin(\theta)\sin(\phi) & \cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\phi) & -\sin(\psi)\cos(\theta) \\ -\sin(\phi)\cos(\phi) & \cos(\theta)\cos(\phi) & \sin(\theta) \\ \sin(\psi)\cos(\phi) + \cos(\psi)\sin(\theta)\sin(\phi) & \sin(\psi)\cos(\phi) - \cos(\psi)\sin(\theta)\cos(\phi) & \cos(\psi)\cos(\theta) \end{bmatrix} \quad (8)$$

By comparing the elements of these two matrices, the Euler angles can then be calculated (Equation 9-11):

$$\psi = \operatorname{atan2}(2(xy + wz), w^2 - x^2 + y^2 - z^2) \quad (9)$$

$$\phi = asin\big(-2(xz - wy)\big) \quad (10)$$

$$\theta = atan2(2(yz + wx), w^2 - x^2 - y^2 + z^2) \quad (11)$$

ACS Modeling in FreeFlyer

When sizing the ACS subsystem, it is important to understand additional mission requirements, such as the satellite's maneuvering rates and how often the satellite needs to be able to perform maneuvers. This chapter presents the models implemented by FreeFlyer to assist the designer in defining ACS component specifications. All the external environmental forces (e.g., gravity, SRP, drag, and lift) must be accurately modeled within FreeFlyer to determine their impact

on orbital lifetime (Figure 21). Once all forces have been calculated, they are summed together and applied to the equations of motion for integration.

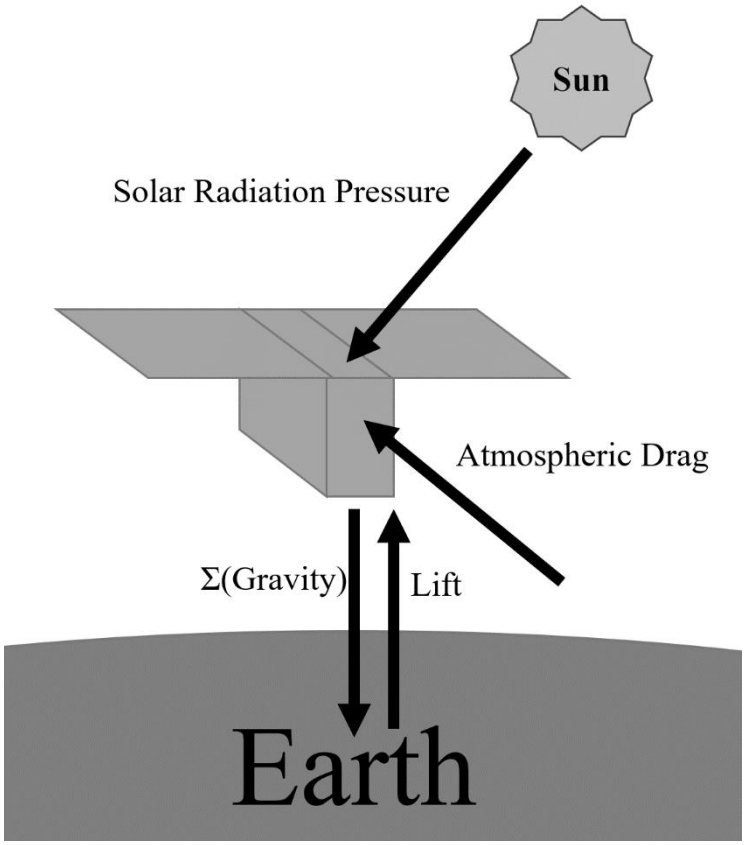

Figure 21: The environmental forces acting upon the satellite. Celestial bodies apply the force of gravity on the satellite. The satellite also experiences solar radiation pressure, atmospheric drag, and lift forces.

Gravity

Equation 12 calculates the magnitude of the gravity force between two bodies [23].

$$|F_g| = \frac{G \cdot m \cdot M}{r^2} \tag{12}$$

where $|F_g|$ is the magnitude of the force from gravity, *G* is the gravitational constant (6.673E-11 Nm^2kg^{-2}), m is the mass of the satellite, *M* is the mass of the secondary body, and r is the distance

between the satellite and the other body. This magnitude is then applied to a vector pointed from the satellite to each large body of influence and summed.

Solar Radiation Pressure

Solar radiation pressure (SRP) is the force applied to an object due to the photons emitted by the Sun exchanging momentum with the satellite. Equation 13 [24] provides a model for SRP.

$$|F_{SRP}| = \frac{S}{c} \cdot C_R \cdot A \tag{13}$$

where F_{SRP} is the SRP force, S is the solar constant, c is the speed of light, C_R is the coefficient of reflectivity, and A is the area of the surface. The solar constant can be determined by FreeFlyer using the solar flux function, which is based on the power of the Sun and the distance between the Sun and the satellite. The Reflectivity Coefficient [25] is a value between zero and two that describes the type of momentum exchange with the object. A value of zero represents an entirely transparent object (e.g., no momentum exchange); a value of one represents a black body (e.g., momentum transferred); and a value of two represents a fully reflective object (e.g., doubling of the momentum). The typical coefficient of reflectivity for a satellite ranges from 1.2 to 1.5 and tends to decrease due to the changes in the surface properties of the satellite over time while in orbit. An exact value can only be measured when the value of the satellite's surface treatments and exterior components is well known. The resulting force magnitude is applied along the Sun-to-satellite vector.

Drag

FreeFlyer [6] also allows for the modeling of atmospheric drag. It is the force caused by the friction of the atmosphere against the satellite, which reduces the satellite's orbital lifetime. Equation 14 is used to model the atmospheric drag [26].

$$|D| = C_D \cdot A_D \cdot \rho \cdot \frac{v^2}{2} \tag{14}$$

where $|D|$ is the magnitude of the drag force, C_D is the coefficient of drag, A_D is the drag area, ρ is the local density of the atmosphere, and v is the velocity of the satellite (relative to the atmosphere). The force is applied in the negative direction of the velocity vector. On average, the drag coefficient is 2.2 but can vary based on atmospheric conditions [27]. As the satellite's altitude the atmospheric density value increases. FreeFlyer [6] generates an analytic model of the atmospheric density values based on the altitude of the satellite.

Lift

Atmospheric lift is also modeled [28]. This force pushes the satellite away from the orbiting body. It is modeled using Equation 15.

$$|L| = C_L \cdot A_L \cdot \rho \cdot \frac{v^2}{2} \tag{15}$$

where $|L|$ is the magnitude of the lift vector, C_L is the coefficient of lift, A_L is the area that lift is applied, ρ is the local density of the atmosphere, and v is the velocity of the satellite (relative to the atmosphere). The lift force is applied along $\overrightarrow{U_L}$ (Equation 16).

$$\overrightarrow{U_L} = \vec{v} \times \left(\vec{R} \times \vec{v}\right) \tag{16}$$

with U_L representing the lift unit vector, $\vec{v}$ equaling the relative velocity of the satellite, and $\vec{R}$ being the satellite's position vector. The velocity with respect to Earth's atmosphere can be determined based on Equation 17.

$$\vec{v} = \overrightarrow{v_{satellite}} - \overrightarrow{\omega_E} \times \vec{R} \tag{17}$$

where $\vec{v}$ is the relative velocity vector, $\overrightarrow{v_{satellite}}$ is the velocity of the satellite, $\overrightarrow{\omega_E}$ is the vector of Earth's spin axis, and $\vec{R}$ is the satellite's position vector.

Orbital Updates

After all forces and torques acting on the satellite are calculated, they are summed up (Figure 22) and applied to the equations of motion by FreeFlyer. Any net negative change in orbital velocity results in orbital decay impacting the lifetime of the satellite [29].

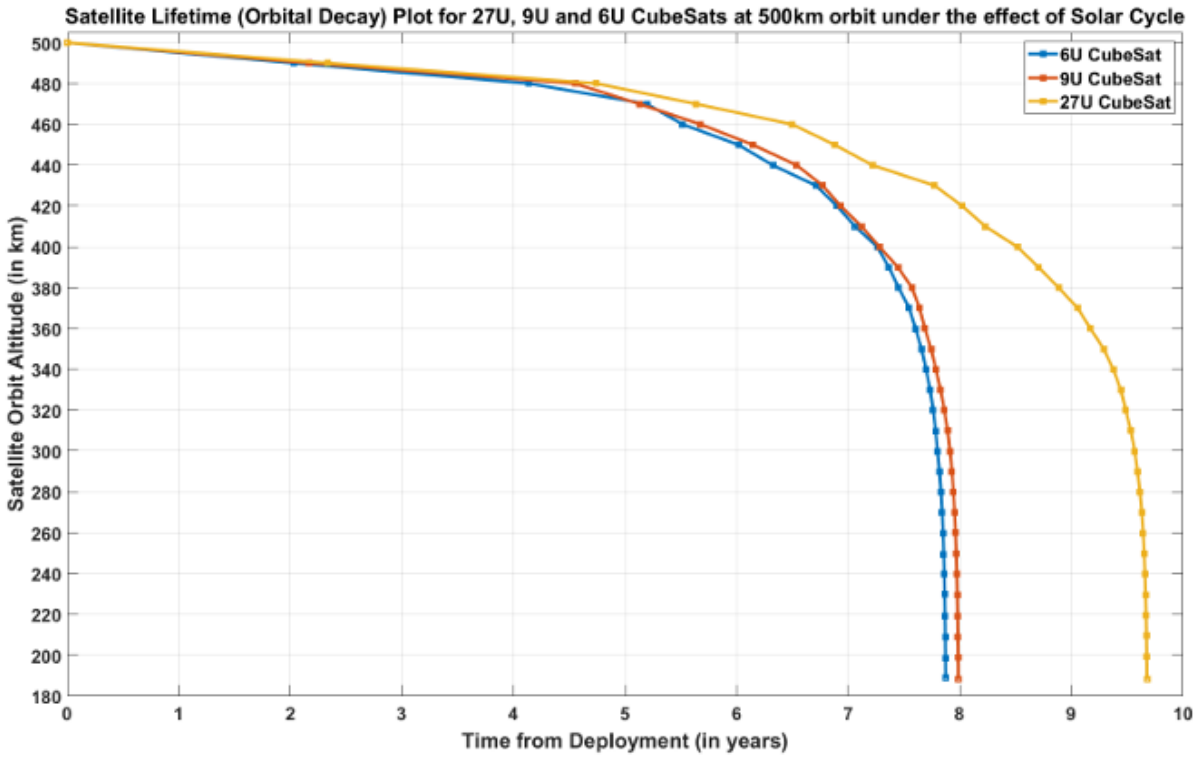

Figure 22: Integration of the environmental forces being applied to the satellite results in a change of orbital velocity, leading to altitude decay and eventual reentry. [29]

Power Modeling in FreeFlyer

When sizing the power subsystem, it is necessary to know how much power the satellite consumes and how much charging power will be needed to replace the energy that the batteries expend during the eclipse portion of the orbit. FreeFlyer is provided with the information necessary to model the satellite's power generation, power consumption, and battery storage systems. These three models interact with one another as shown in the diagram in Figure 23.

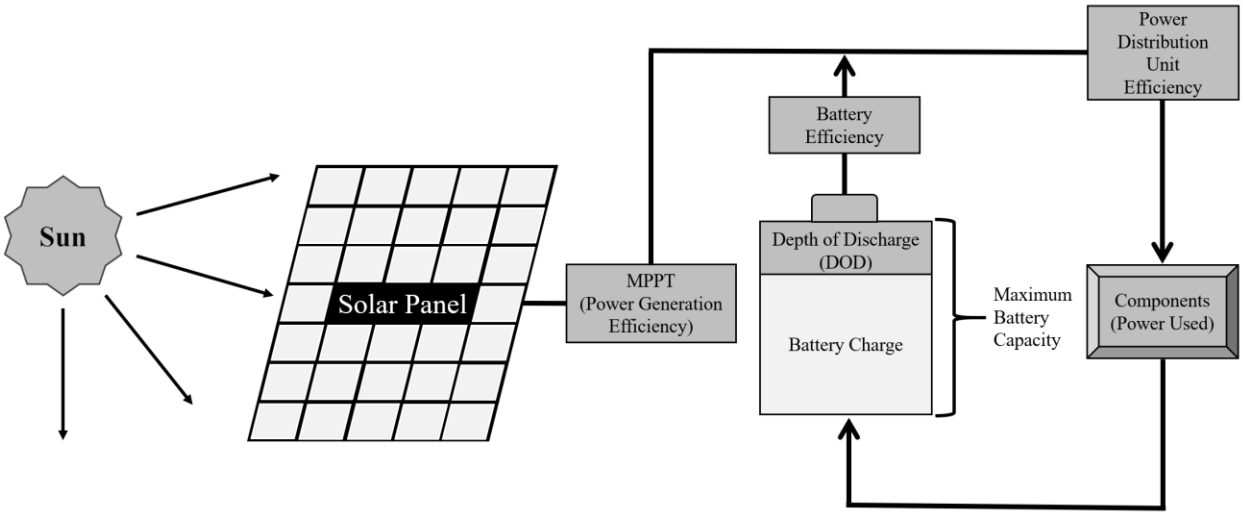

Figure 23: Power generation, consumption, and storage flow chart. The solar panel generates power from the Sun's radiation. Efficiency losses in the MPPT, PDU, and batteries reduce this power

Power Generation Model

Peak power is generated when the Sun directly illuminates the solar panels, and power generation diminishes as the Sun strikes off-angle [30]. The angle between the Sun and the satellite is labeled as θ (Figure 24).

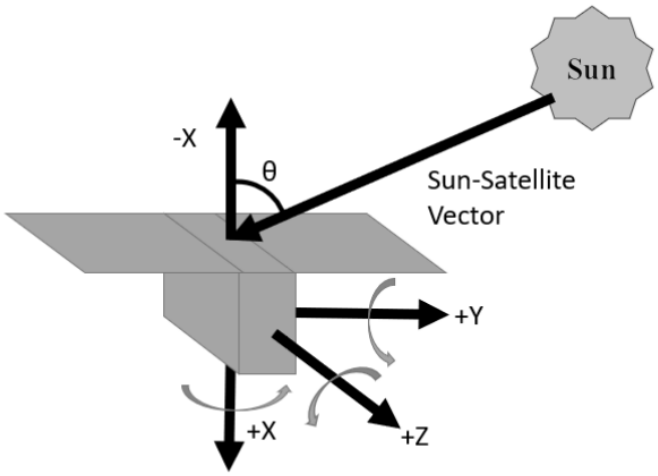

Figure 24: A visualization of the angle θ between the solar panel normal and the Sun-satellite vector.

The amount of energy generated by the arrays is also affected by the "solar constant," [31] which represents the solar illumination flux deposited on the solar arrays (usually given in W/m^2). The solar constant is dependent on the satellite's distance from the Sun and changes by up to 7% throughout the course of a year in Earth's orbit (Figure 25).

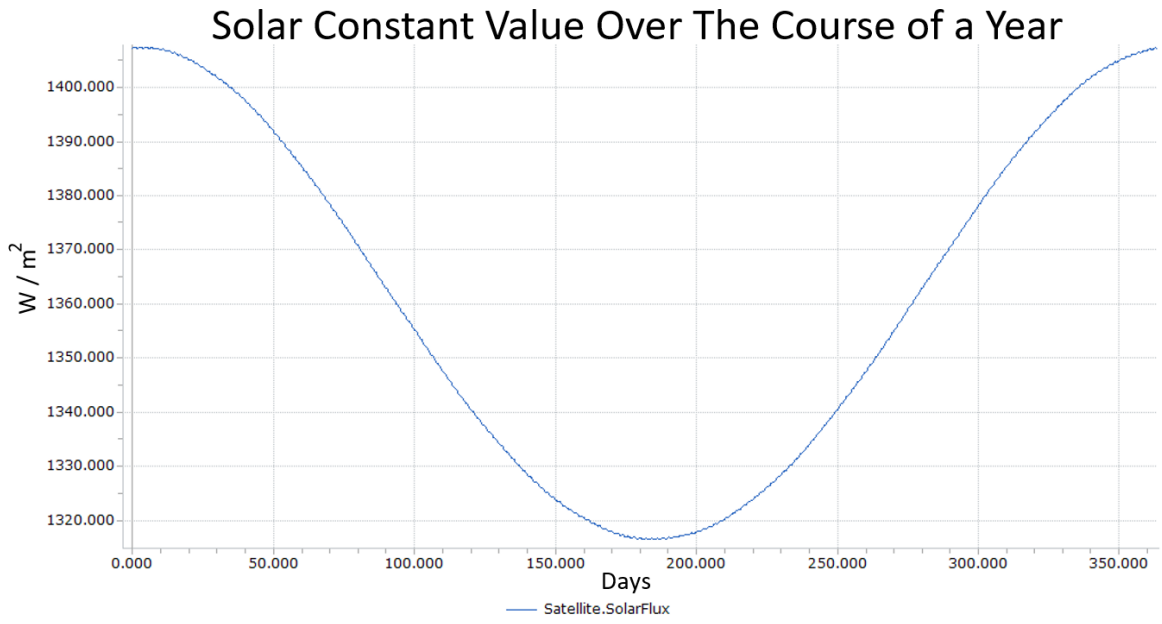

Figure 25: A plot of the dynamic solar constant throughout the year. The solar constant can change as much as 7% throughout the year.

To compute the amount of power that can be generated, the number of solar cells mounted to the array is multiplied by the area of an individual solar cell (Equation 18).

$$A_{SP} = A_{SC} \cdot n \tag{18}$$

where A_{SP} equals the area of the solar panel, A_{SC} equals the area of a solar cell, and *n* represents the number of solar cells on the solar panel. Solar cell efficiency is the last variable needed to determine the satellite's generated power. The solar cell efficiency commonly ranges from 20% to 40% [17] and can be found in the specification sheet of the selected solar cell. Equation 19 provides the equation for calculating the generated power [30].

$$P_G = \cos(\theta) \cdot S \cdot A_{SP} \cdot \eta \tag{19}$$

where θ is the solar incidence angle, P_G is the power generated, *S* is the solar constant, and η represents the solar cell efficiency.

Power can also be generated from the sunlight reflected off Earth's atmosphere [32]. To determine this value, three additional values are required. The first is the albedo factor (α), a coefficient ranging from zero to one, with more reflective surfaces having a higher albedo. This value constantly changes while orbiting around Earth, and it is affected by cloud cover and other surface features. The annual average value is 0.3. The following variable is the view factor of the Earth-facing array, which can be calculated as Equation 20.

$$vf = \cos(\varphi) \tag{20}$$

where *vf* represents the view factor variable coefficient, and φ represents the angles between the satellite's solar panel normal vector and the satellite-to-Earth normal vector (Figure 26).

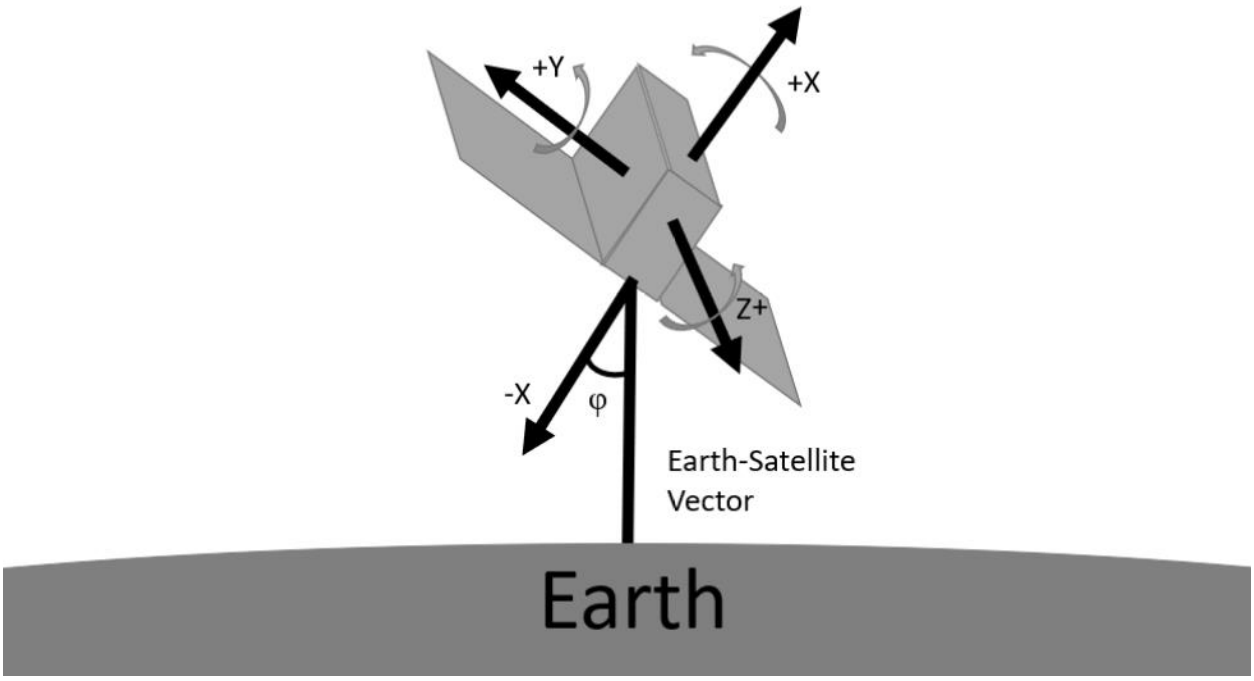

Figure 26: A visualization of the angle φ between the solar panel normal and the Earth-satellite vector.

The intensity of the sunlight decreases proportionately to the cosine of the angle between the Sun-to-Earth vector and the Earth-to-satellite vector (Equation 21).

$$if = S \cdot \cos(\beta) \tag{21}$$

where *if* represents the intensity factor, *S* is equal to the solar constant, and β represents the angle between the satellite-to-Earth vector and the Earth-to-Sun vector (Figure 27).

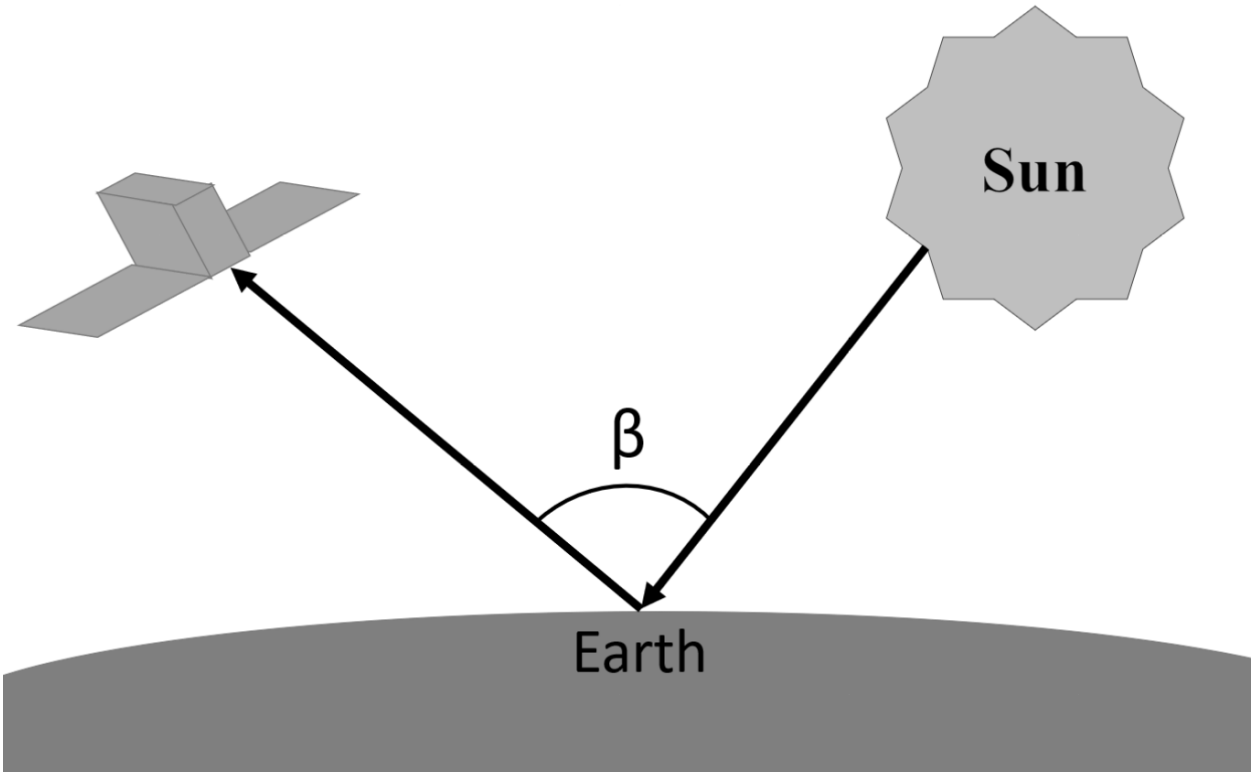

Figure 27: A visualization of the angle β between the Sun-Earth vector and the Earth-satellite vector.

In Equation 22, the Albedo Power Generation Equation is obtained by combining all of these factors.

$$P_G = A_{SP} \cdot \eta \cdot \alpha \cdot vf \cdot if \tag{22}$$

where P_G is the power generated, A_{SP} is the area of the solar panel, η is the solar cell efficiency, α is the albedo factor, *vf* is the view factor coefficient, and *if* is the intensity factor.

Another factor that needs to be considered when designing a power generation model is the potential shadowing of other solar panels if deployables are present. If the satellite has any deployables, they may partially or entirely obstruct the face of the satellite from the Sun or Earth. A script was developed to model the platform configuration information provided by the user to call FreeFlyer [6] provided functions to determine the view factor of the Sun and Earth for the solar panels as well as the illuminated percentage value for each panel.

Maximum Power Point Tracking (MPPT) [33] allows solar power systems to be optimized in terms of power production. This component maximizes power (which can be calculated using $P = I \cdot V$) by continuously adjusting the electrical operating point of the array to maximize the output power. The output conditions that maximize power output are called the Maximum Power Point (MPP) (Figure 28). The MPPT optimizes the power output by using a control algorithm to determine which direction to adjust the operating voltage to increase the power output within the parameters allowed by both the solar array and the battery bank to which the cells are connected.

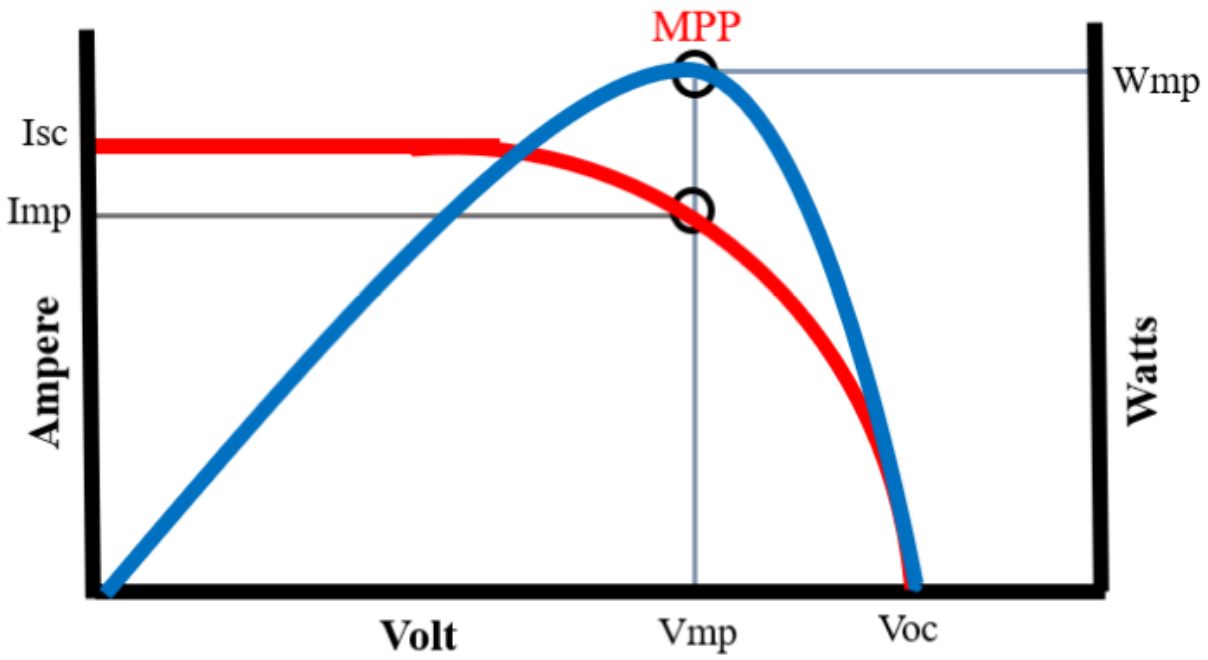

Figure 28: A plot displaying the current vs. voltage curves of a solar panel along with its power vs. voltage curve. The MPP is the maximum point on the watts vs. the voltage curve and the equivalent voltage on current vs. voltage curve.

MPPTs are not 100% efficient; a percentage of the maximum possible power generated is lost as heat in the process. Lower percentages of peak power generation result in less efficient operation of the MPPTs (Figure 29).

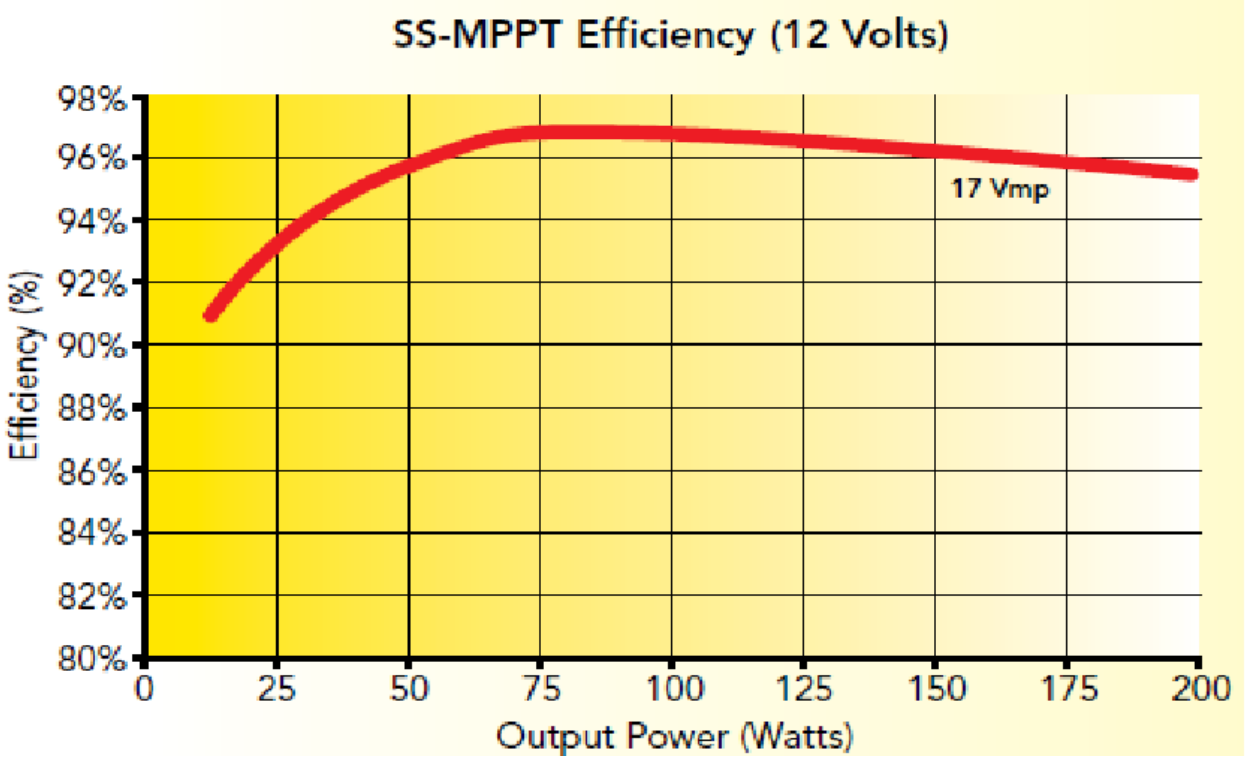

Figure 29: A plot of the efficiency of the MPPTs at various power outputs. The trend is that at lower power outputs, the MPPTs become less efficient. [33]

Solar panel degradation occurs because of material degradation, thermal cycling, outgassing, radiation, abrasion, and micrometer impacts. While the exact degradation rate vary from manufacturer to manufacturer and is also based on operating domain, commercial-grade solar cells typically have a degradation rate of 3% per year in LEO [34]. The solar panel degradation factor, *D*, can be modeled using the formula provided in Equation 23.

$$D = (1 - r)^{t/365} \tag{23}$$

where *r* is the degradation rate and *t* is the time in days.

The efficiency of the Power Distribution Unit (PDU) is also considered when determining the total power usage. As a result of voltage conversions and wiring resistances, a small portion of power is lost to heat, with typical ranges of efficiencies for a small satellite PDU ranging from 95% to 99% [17]. Equation 24 gives the total energy consumed by the satellite.

$$P_U = \frac{\sum(P_C)}{k} \tag{24}$$

where P_C represents the power consumed by the components, k represents the power distribution efficiency percentage, and P_U represents the total power used. The total power used is measured at each time step and may vary depending on the satellite's mode of operation.

Power Consumption Model

A satellite is comprised of several power-consuming components. Once the designer makes their initial component selections, a power budget is developed to identify each component's power consumption and duty cycle. For example, the flight computer must operate 100% of the time to control the satellite, while battery heaters may only be on intermittently to maintain temperature. Individual components can be commanded on and off in FreeFlyer at specific times, depending on the satellite's mode of operation. The power consumed over time is then summed to provide an estimate of the orbit average power required for satellite operations. An example of a power budget is shown in Table 1.

Table 1: Example Power Budget.

Component	Voltage (V)	Amperage (A)	Quantity	Duty Cycle (%)	Total Power
Flight Computer	3.3	0.1	1	100%	0.33
Payload Controller	3.3	0.075	1	100%	0.25
PDU	3.3	0.0455	1	100%	0.15
Heaters	3.3	0.1	4	20%	1.33
Magnetorquers	3.3	0.121	6	50%	1.2
Reaction Wheels (Steady State)	3.3	0.0455	4	100%	0.6
Sun Sensors	5.0	0.01	6	100%	0.3
IMU	3.3	0.3	1	100%	1.5
Radio	5.0	1.0	1	5%	5.0
Payload	20.0	1.0	1	30%	20.0

Battery Model

The satellite battery model [30] takes into account the maximum battery capacity and the desired maximum Depth of Discharge (DOD) allowed, as a function of the desired battery life. DOD is the percentage of the energy in the battery that can be used without significantly degrading the battery's lifetime, given the number of charge cycles the battery pack will experience. The maximum battery capacity can be determined simply by multiplying the number of battery cells by the cell's capacity (Equation 25). Most small satellite battery packs come in multiples of two or four cells.

$$C_T = C_B \cdot n \tag{25}$$

where C_T is the total capacity of the batteries, C_B is the capacity of a single battery cell, and n is the number of battery cells.

To determine the battery pack's maximum DOD [35], the number of charge and discharge cycles must be estimated as a function of the satellite lifetime and orbital period (Equation 26).

$$Q = \frac{T_L}{T_O} \tag{26}$$

where Q is the number of cycles that the batteries experience, T_L is the lifetime of the satellite, and T_O is the time of the satellite's orbit period. Battery manufacturers provide guidelines for selecting an appropriate DOD for their batteries. An example of the Battery Cycle vs. DOD plot for a NaS battery is provided in Figure 30.

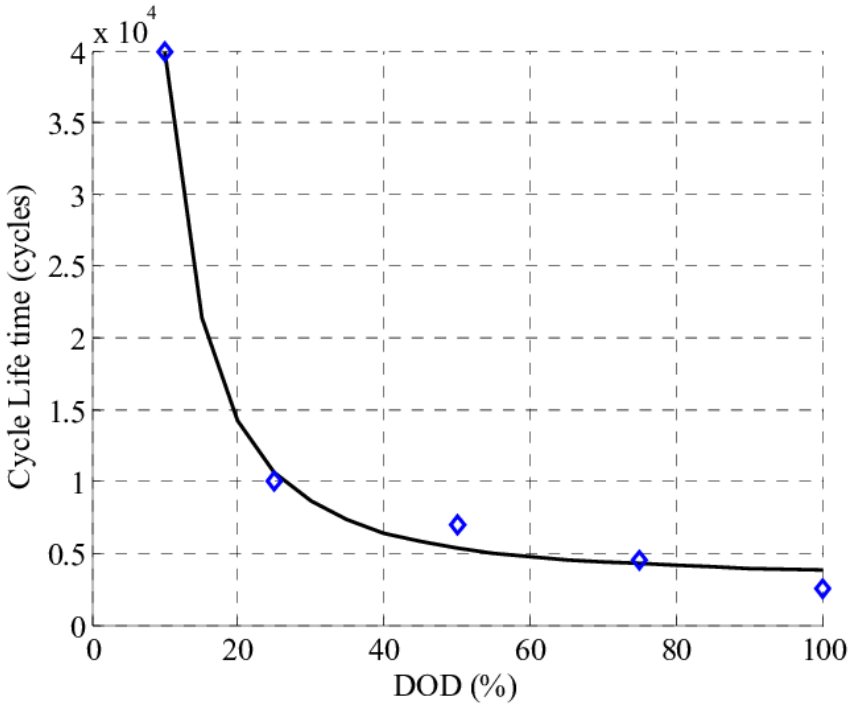

Figure 30: A NaS Cycle Count vs. Depth of Discharge graph. Based on the number of charge and discharge cycles planned within the mission lifetime, the usable capacity of the batteries changes. These plots look different for different battery chemistries. [35]

FreeFlyer computes a running total of the amount of energy stored in the batteries. After every time step, energy consumed by satellite operations is subtracted from this total and any surplus energy generated for charging is added to this value.

Data Budget Modeling in FreeFlyer

This section presents the models implemented in FreeFlyer that allow the user to determine the amount of data generated onboard the satellite by system telemetry and the payload to size the nonvolatile storage needed onboard the satellite.

The first step in sizing this storage component is estimating the data generation rates of the payload and other components within the satellite. This involves identifying the resolution (e.g., size) of the data packages, the frequency of reporting, and the type of data that needs to be stored. An example of this data budget can be seen in Table 2.

Table 2: Example Data Budget.

Component	Data Field	Unit	Data Format / Bytes	Quantity	Frequency (per second)
Payload	CMD State	Boolean	unsigned / 2	1	1
Payload	Power	Amps	double / 4	1	1
Payload	Heater	Amps	unsigned / 2	1	1
Reaction Wheel	Rotation Rate	Deg/S	double / 4	4	1
Gyroscope	Rotation Rate	Deg/S	double / 4	4	1
Battery	Voltage	V	double / 4	1	1
Battery	Current	A	double / 4	1	1
Battery	Temperature	C	double / 4	8	0.1

At each time step, the data generated since the last step is summed together and tabulated as the Data-Generated value (Equation 27).

$$D = \sum (F \cdot n \cdot f) \cdot t \tag{27}$$

where D is the number of bytes generated this time step, F is the number of bytes generated per data field, n represents the number of components generating that particular type of data, f represents the frequency of the data logging, and t represents the timestep in FreeFlyer. The resultant Data-Generated variable is continuously summed together after every time step. Depending on the mode of operation of the satellite, the data generated at each time step may be different from the previous value.

Data is collected from the payload and satellite subsystems and stored before downlink sessions with the ground station. Data generated at each timestep is added to the running storage total and decremented from the total when a downlink occurs at the rate determined by the link analysis.

Communications Modeling in FreeFlyer

FreeFlyer allows the user to define the characteristics of the ground station and onboard radio for the communications subsystem required to recover satellite data on the ground. FreeFlyer determines when the satellite has a line of sight with a ground station and then assesses whether a viable downlink can be established, given the performance parameters provided for the space and ground equipment implementing the link [6].

Ground Station Visibility

The FreeFlyer visibility calculator function determines if a satellite is within range of a ground station. Ground station locations are specified using latitude, longitude, and altitude (for location), along with the horizon mask definition, to determine the minimum angle of elevation required for successful communication (Figure 31). The mask represents the region in which communications between the satellite and the ground station can take place.

Figure 31: A visualization of the mask's elevation.

The options for the mask's geometry are between a cone (uniform) or a custom shape (which allows for a custom horizon polygon to be projected). For the cone-shaped mask, the user can specify the minimum usable elevation. For custom shape masks, the user can edit the number

of points on the polygon and change their location with respect to the center point of the ground station (Figure 32).

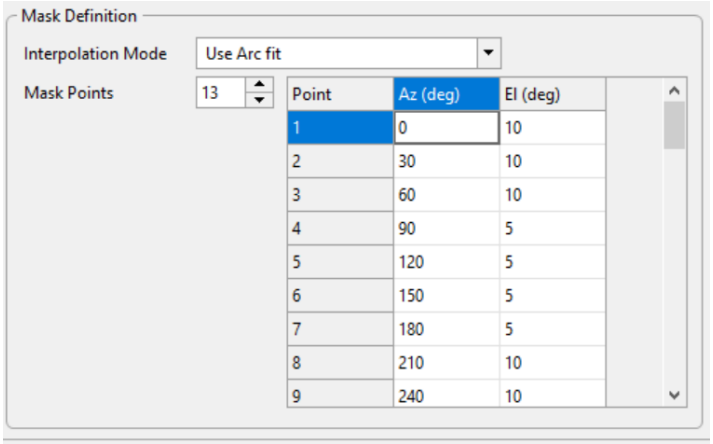

Figure 32: An example input with varying elevation for the custom-shaped mask. The figure displays a 13-point mask with varying elevations between 10 and 5 degrees.

In Figure 33, the ground station and its mask are red, and the visibility segment is yellow. In FreeFlyer, a visibility segment between the satellite and a ground station can be created. A visibility segment determines whether a set observer (e.g., the satellite) can see a target (e.g., the ground station). If any other object (e.g., a planet, satellite, or star) is between the satellite and the ground station, the visibility calculator will report that a link is unavailable.

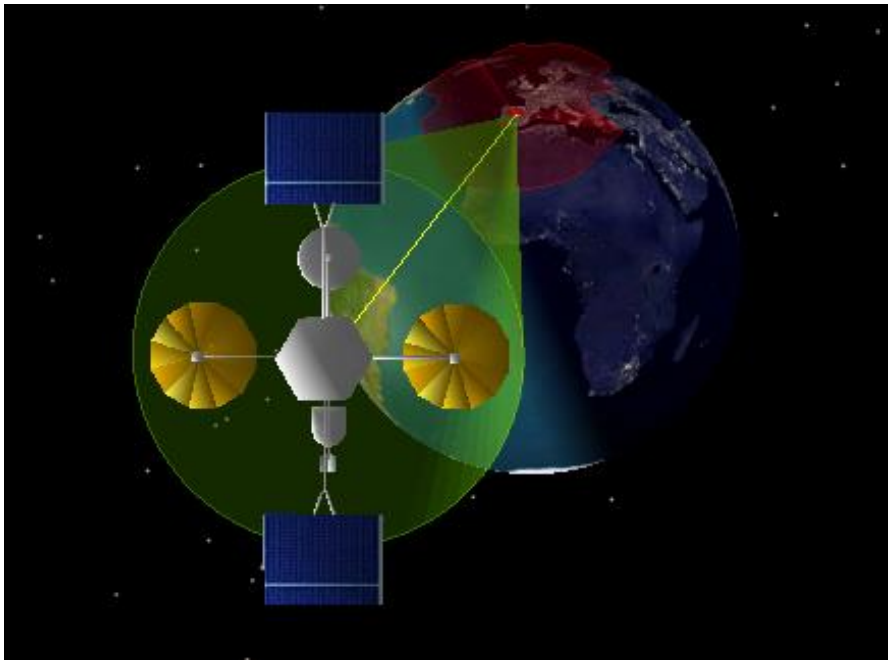

Figure 33: Spacecraft downlink 3D visualization in FreeFlyer. The yellow represents the visibility segment. The custom-shaped mask is represented in red. The red dot represents the ground station location.

Link Budget

Along with the ground station being visible from the satellite, a transmitted signal must be strong enough to achieve the desired bandwidth. The link budget analysis is based on the Friis transmission equation (Equation 28). The Friis transmission equation [36] is a basic equation that can be used to determine the margin of the link between the ground station and the onboard radio in dB.

$$M = -S_R + P_T + G_T + G_R + L \tag{28}$$

where M is the margin in the link, S_R is the ground station's sensitivity, P_T is the transmitter's power, G_T is the transmitter's gain, G_R is the receiver's gain, and L is the Free-Space Path Loss (FSPL) of the signal between the satellite and the ground station. All units are in dB.

Equation 29 represents the free-space path loss of the signal. The free-space path loss, L, is the energy loss resulting from transmitting through space and the atmosphere.

$$L = 20log_{10}\left(\frac{4\pi \cdot f \cdot d}{c}\right) \tag{29}$$

where L represents the total free-space path loss between a transmitter and a receiver, f represents the signal frequency, d is the distance between the transmitter and the receiver, and c is the speed of light. Figure 34 displays a visual representation plot of the Friis transmission equation.

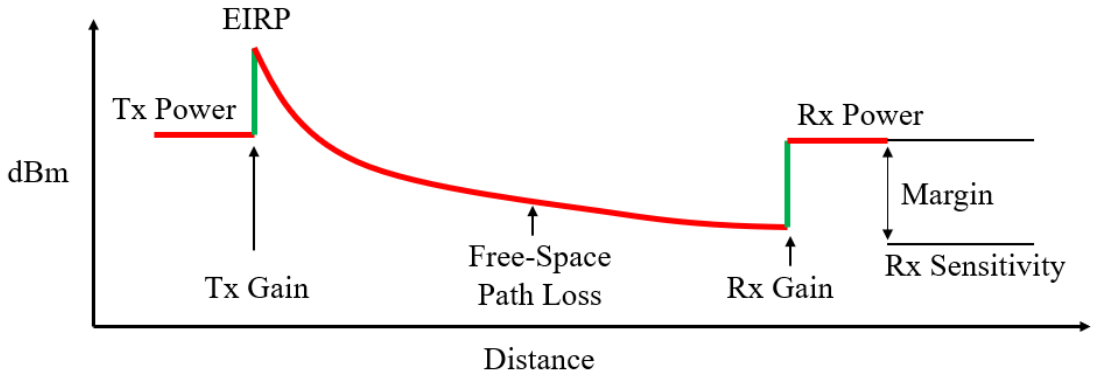

Figure 34: A visual representation of the Friis transmission equation with labels illustrating what components are contributing to the increase or decrease in the system's total dBm.

One factor not computed directly in this link analysis, but is frequently subtracted from the margin, is atmospheric attenuation. This factor is typically addressed by adding a margin of at least three dB to the link budget. There are several other minor factors that are accounted for in this margin, including losses from coaxial cable runs and thermal noise.

After determining that a link can be established, the rate of data that is capable of being sent to the receiver is calculated. To do this, the Shannon-Hartly theorem [37] (Equation 30) is used.

$$C = B \cdot \log_2\left(1 + \frac{S}{N}\right) \tag{30}$$

where *C* is the channel capacity in bits per second, *B* is the bandwidth in hertz, and the *S/N* function is the signal-to-noise ratio of the communication. The channel capacity (data rate) is based on the bandwidth available and the signal-to-noise ratio of the communications link. This data rate is multiplied by the time step to determine the amount of data that can travel from the satellite to the ground station per step while the communications link is established.

Downlink Modeling

To model the downlink budget (Figure 35) in FreeFlyer, the satellite line of sight range to the ground station is calculated using the visibility calculator function. The strength of the connection between the satellite and the ground station is then evaluated by performing a link budget analysis. This analysis determines the rate at which data can be downlinked from the satellite to the ground station. Now that the downlink rate is established, the data currently stored within memory can be broadcast from the satellite to the ground station and deleted from the satellite, freeing up memory for future data to be stored.

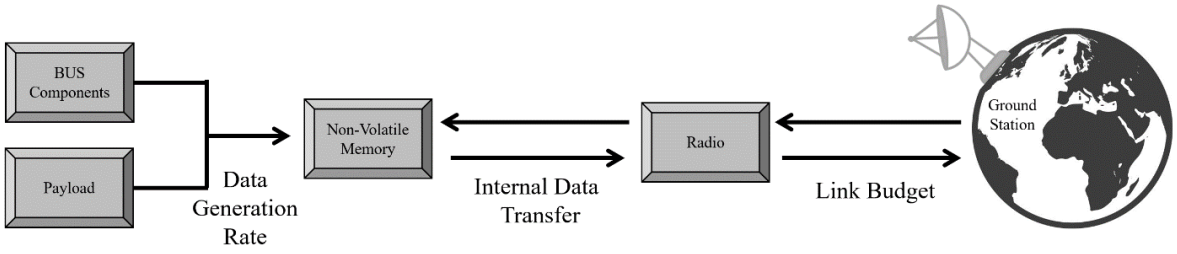

Figure 35: A flow chart illustrating data generation, data storage, and data-downlink. The reverse command-uplink path is also shown.

CHAPTER 5: DESIGN REFERENCE MISSION

To establish the FreeFlyer tool's utility, a design reference mission is defined to demonstrate various features of the tool. Several specific questions about the design can be answered:

1. What is the impact of environmental forces and moments on the lifetime of the satellite?
2. What is the size of the solar arrays required to carry out the mission?
3. How many batteries are required to operate the satellite during the eclipse?
4. How much onboard storage is required to facilitate data downlink to the ground?

For this exercise, a previously defined LASSI mission, LAICE-F [7], serves as a checkpoint with verified performance parameters to compare with this tool's outputs. LAICE-F completed its Preliminary Design Review (PDR) in June 2022. Mission-defining PDR parameters were input into the FreeFlyer tool (APPENDIX A), and the results are compared to those obtained for LAICE-F at PDR to ensure the scripted models were implemented correctly into the FreeFlyer scripts.

Design Reference Mission Profile

LAICE-F is a 6U CubeSat with two payloads. The first payload (known as the VT payload) measures the variations in ion and neutral density gravities that gravity waves create. The second payload (known as the UIUC payload) measures airglow brightness to determine the wave's

parameters. LAICE-F's concept of operations outlines the major functions that the satellite will provide throughout the mission's lifetime (Figure 36).

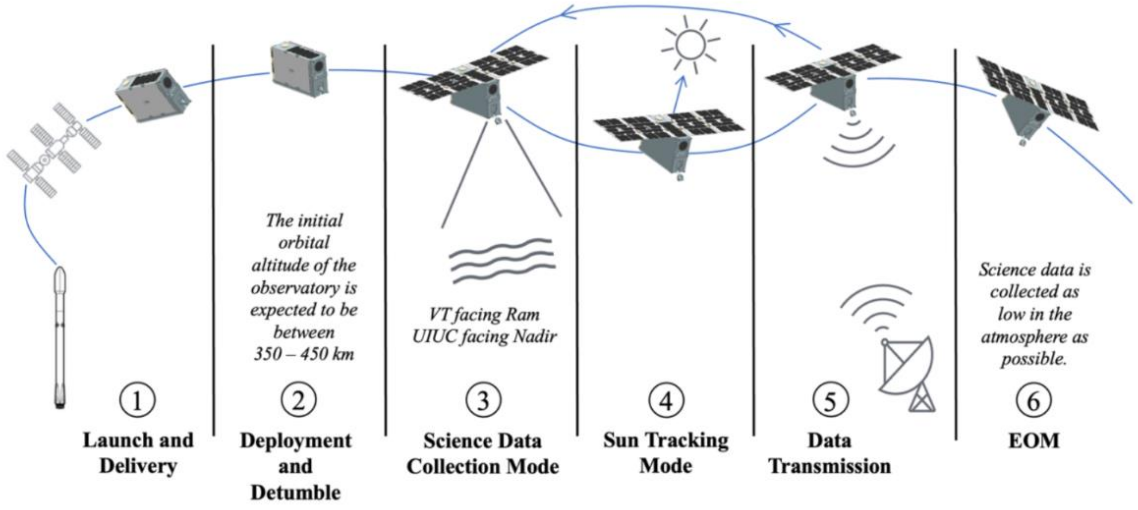

Figure 36: LAICE-F's concept of operations. The major mission phases are defined from the launch to the end of mission. [7]

Steps three, four, and five of the concept of operations are modeled in FreeFlyer. Several constraints have been incorporated into this model, which captures the principal payload requirements listed in Table 3.

Table 3: The LAICE-F requirements that were implemented into FreeFlyer.

Req #	Requirement
1.0	The VT payload shall be aligned to ram ± 10°.
2.0	The UIUC payload shall operate only during eclipse.
3.0	The UIUC payload shall be aligned to nadir ± 10° during operations.
4.0	The UIUC payload shall be kept pointed > 23° away from the Sun.
5.0	The Satellite shall complete its scientific objectives within 9 months and be designed to have a mission lifetime > 2 years.

These payload requirements primarily influence the attitude control and power generation requirements levied on the satellite. Since LAICE-F lacks orbit-raising thrusters in its design, orbital parameters to meet the mission lifetime requirements were selected to complete the mission objective before the satellite's natural orbit decay removed it from orbit. Due to launch availability and the primary orbital regions in which science can be collected, an orbit inclination of 51.6 was selected. Therefore. deployment from the ISS is feasible, so the initial altitude used for the analysis is 420 km [38].

All sizing for the power subsystem design is based on the defined power budget (Table 4) assembled using the initial component selections from LAICE-F's PDR.

Table 4: LAICE-F PDR Power Budget

Component	Voltage (V)	Amperage (A)	Quantity	Duty Cycle (%)	Total Power
Flight Computer	3.3	0.1	1	100%	0.33
Payload Controller	3.3	0.075	1	100%	0.25
Power Distribution Unit	3.3	0.0455	1	100%	0.15
Battery Heaters	3.3	0.1	4	100%	1.33
Magnetorquers	3.3	0.121	6	50%	1.2
Reaction Wheel	5.0	0.03	4	100%	0.6
Sun Sensor	5.0	0.01	6	100%	0.3
GPS Receiver	3.3	0.0378	1	100%	0.12
GPS Antenna	3.3	0.02	1	100%	0.07
IMU	5.0	0.3	1	100%	1.5
Radio Rx	5.0	0.033	1	100%	0.17
Radio Tx	5.0	2.618	1	1.11%	0.09
UIUC Payload (Standby)	5.0	0.036	1	66%	0.12
UIUC Payload (Active)	5.0	1.3	1	33%	2.7
VT Payload (Standby)	5.0	0.634	1	100%	3.17
VT Payload (Active)	5.0	1.73	1	100%	8.65

Power generation is provided by a solar panel deployed across the -X face of the spacecraft. The panel is composed of two sets of solar arrays, initially positioned along the side of the satellite prior to deployment. The solar cells have an efficiency of 30% [39] and an area of 30.18 cm^2. For power storage, the battery cells store 11.59 WHrs each [40], with the batteries configured in a two-series, two-parallel (2S2P) arrangement [7]. Batteries can only be added in sets of two in this configuration. The batteries begin the mission fully charged.

The complete data budget for the bus telemetry is provided in APPENDIX B, and an overview of the satellite's data budget is shown in Table 5. On average, 912 bytes of data are generated per second. LAICE-F utilizes an S-Band radio [41], a patch antenna [42], and a ground station at UIUC with a minimum elevation angle of 5° [7]. The communication systems are modeled using a downlink frequency of 2.2 GHz with a desired bandwidth of 1 MHz. The power and gain link budget info for the radio and ground station are in Table 6.

Table 5: LAICE-F data generation rates.

Source	Data (bits / second)
Satellite Bus	3072
VT Payload	2704
UIUC Payload	168
Overhead (25%)	1464

Table 6: Communication systems performance parameters for LAICE-F

Variable	Value (dBm)
Receiver Sensitivity	-110
Transmitter Power	30
Receiver Power	53
Transmitter Gain	6
Receiver Gain	32.6

With the design reference mission for LAICE-F defined, the new FreeFlyer tool can now be used to iterate through component sizing analysis to establish the orbital lifetime, the size of the solar arrays and batteries, and the data storage requirements that accommodate the downlink schedule for the given set of orbital parameters.

Design Reference Mission Modeling

Power

The first subsystem to be analyzed is the power subsystem. It is necessary to ensure that the satellite bus and both payloads receive sufficient power throughout the mission phases. To ensure this, two questions are asked: How big do the solar panels need to be? How many battery cells are necessary?

The PDR design of the power subsystem for LAICE-F consists of dual-deploying solar panels (Figure 37). Each deployable contains 14 solar cells, and five are located on the center body of the satellite, for a total of 61 solar cells. Eight battery cells are configured into a two-series, four-parallel pack that is used for energy storage during the orbit's eclipse periods.

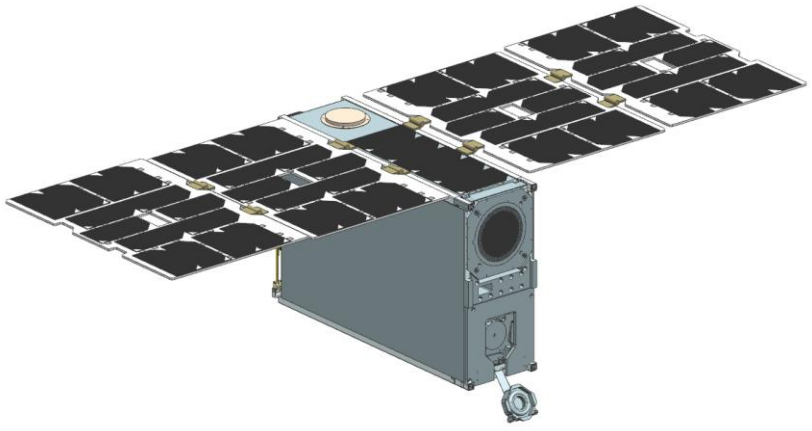

Figure 37: LAICE-F CAD model displaying the double deployable solar cells. [7]

The peak power generation of LAICE-F is predicted using Equation 19, where θ, the incidence angle, is 0, S, the solar constant, is 1361, A_{SP}, the area of the solar panel, is 61 * 0.003018, and η, the solar cell efficiency, is equal to 0.3. A plot shows the power generated during a day-in-the-life of the satellite, moving in and out of eclipse (Figure 38). The predicted peak power estimated from LAICE-F's power budget is 75.16 W. The FreeFlyer tool, incorporating precise orbital and attitude information, yields a close value of 74.73 W.

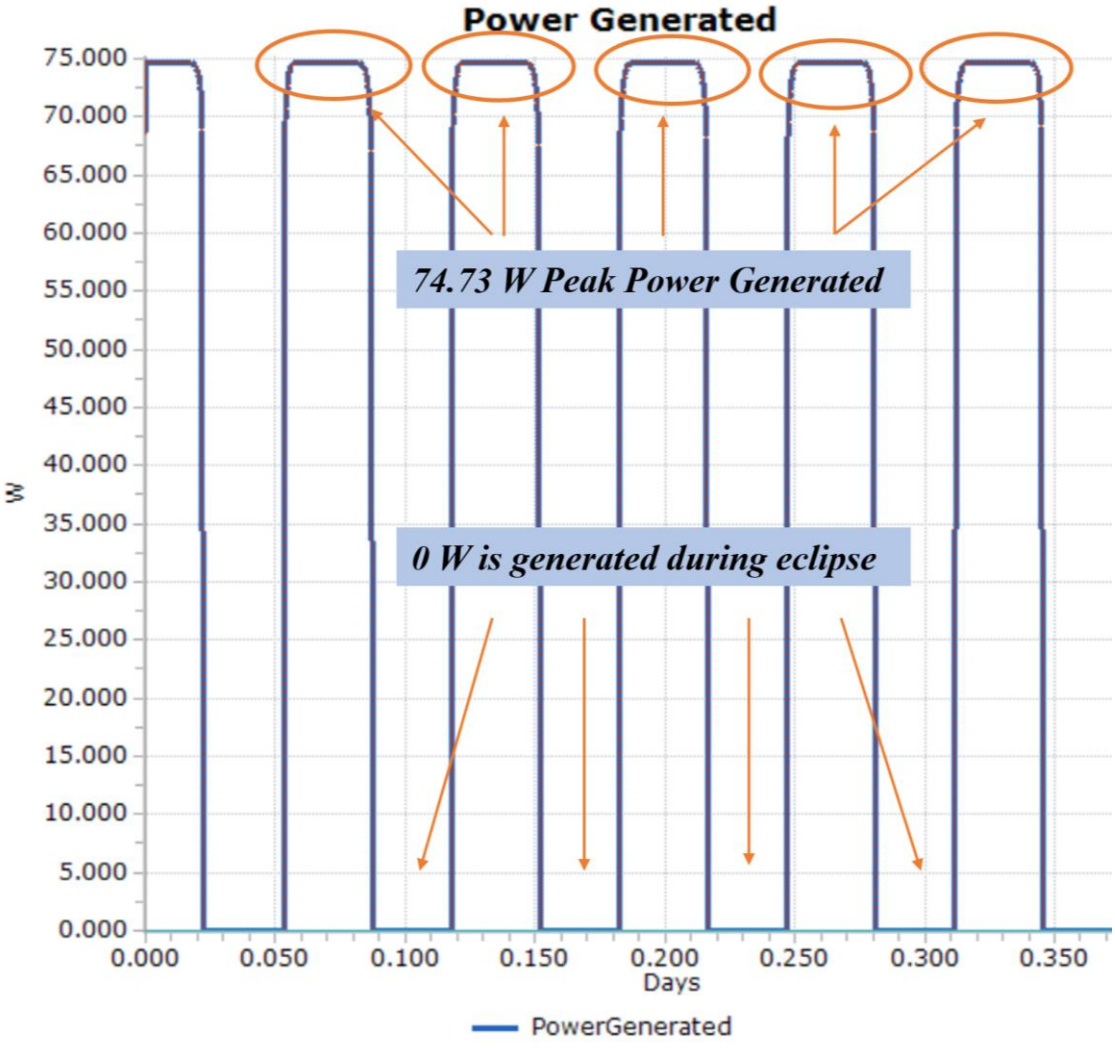

Figure 38: Power generation plot generated by the FreeFlyer tool. Displays a peak power generation value of 74.73 W and a minimum power of 0 W due to the inability of the satellite to generate energy while in the eclipse portion of the orbit.

When expanding the simulation to cover an entire year, it is noted that the power generation unexpectedly drops to zero multiple times throughout the year (Figure 39). After further analysis, it is evident that the inertially stabilized solar panels are turned edge-on to the Sun during these periods.

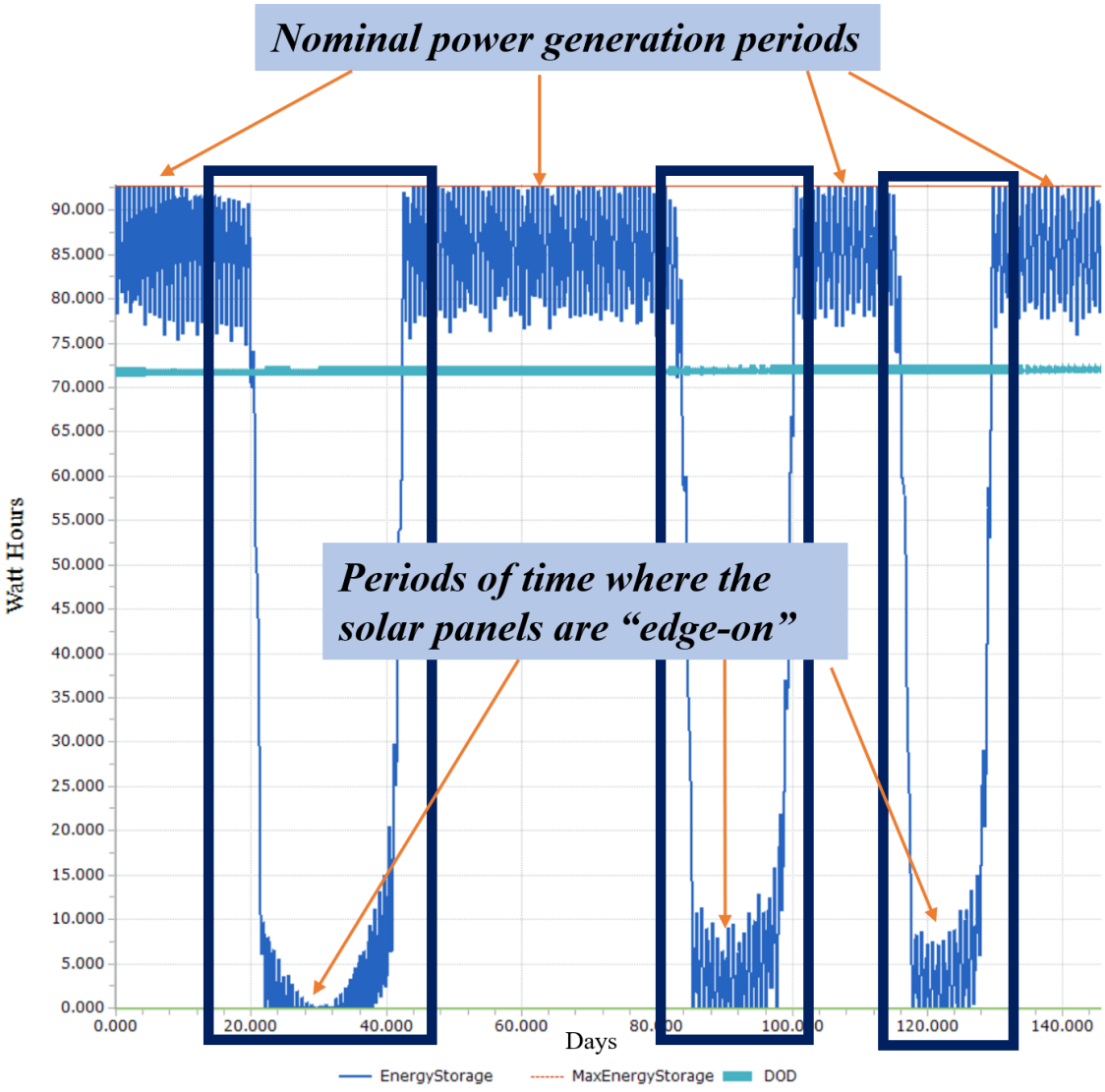

Figure 39: Energy storage plot showing the results of the edge on solar panels. There are multi-week-long periods where the batteries become completely drained and the satellite fails.

This occurs because of the recession of the right ascension of the ascending node (RAAN) when LAICE-F's orbit traces the terminator between the eclipse and sunlit portions of the orbit (Figure 40). To mitigate the edge-on behavior, solar array Sun tracking becomes a requirement. The FreeFlyer simulation is updated to rotate the solar arrays to point directly at the Sun when not in eclipse. The power system maintains power generation at > 73.24 Watts throughout the year, eliminating the seasonal gap that resulted from inertial pointing.

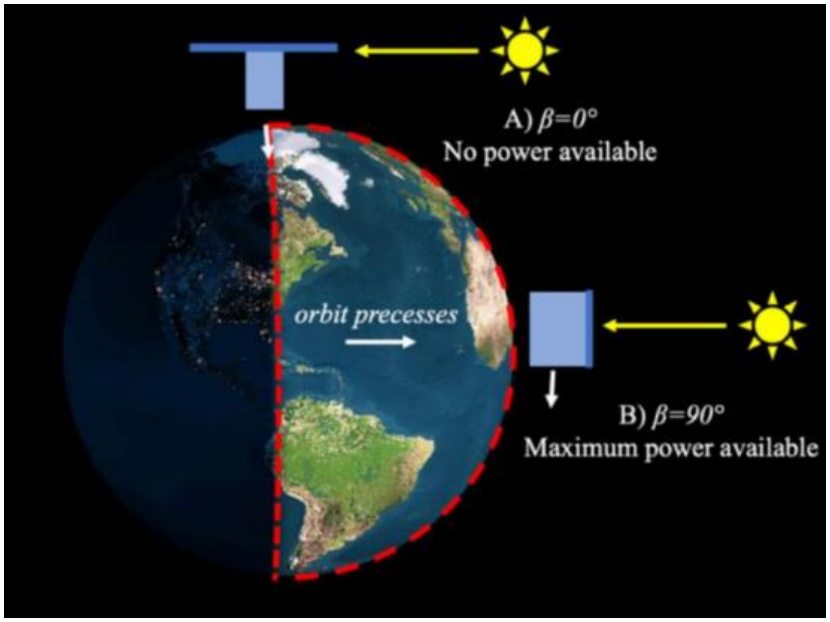

Figure 40: Case A shows an "edge-on" case for LAICE-F, which results in minimal power generation for the spacecraft. Case B shows an ideal maximum power generation case. [7]

To determine if the PDR design incorporating 61 solar cells is sufficient for LAICE-F, the total power generation requirement needs to be calculated. Battery charging power needs to be added to the power that the satellite uses while on the sunlit portion of the orbit, replenishing the energy in the batteries utilized on the eclipse portion of the orbit.

LAICE-F has four operational modes defined for the mission. The Single Payload operating mode consumes ~16 W and corresponds to a powered satellite bus and VT payload. The

Single Payload mode is the default mode of operation and is active during the sunlit side of the orbit. The Dual Payload operating mode consumes ~18.5 W and corresponds to the powered satellite bus, VT payload, and UIUC payload. This is the second most frequent mode of operation, and it is active during the eclipse portions of the orbit. The Single Payload with Comms consumes ~29.5 W and occurs on the sunlit side of the orbit if a communications downlink session occurs while the VT payload and bus are also operating. It is a short duration high-power mode. Similarly, the Dual Payload with Comms consumes ~32 W occurs when a downlink is in progress during the eclipse portion of the orbit while the satellite bus, VT payload, and UIUC payload are operating. These modes of operation are evident in the visualization of power consumption (Figure 41). The power consumption plot is most useful when troubleshooting the implementation of the mission profile to ensure the correct systems turn on and off when they should.

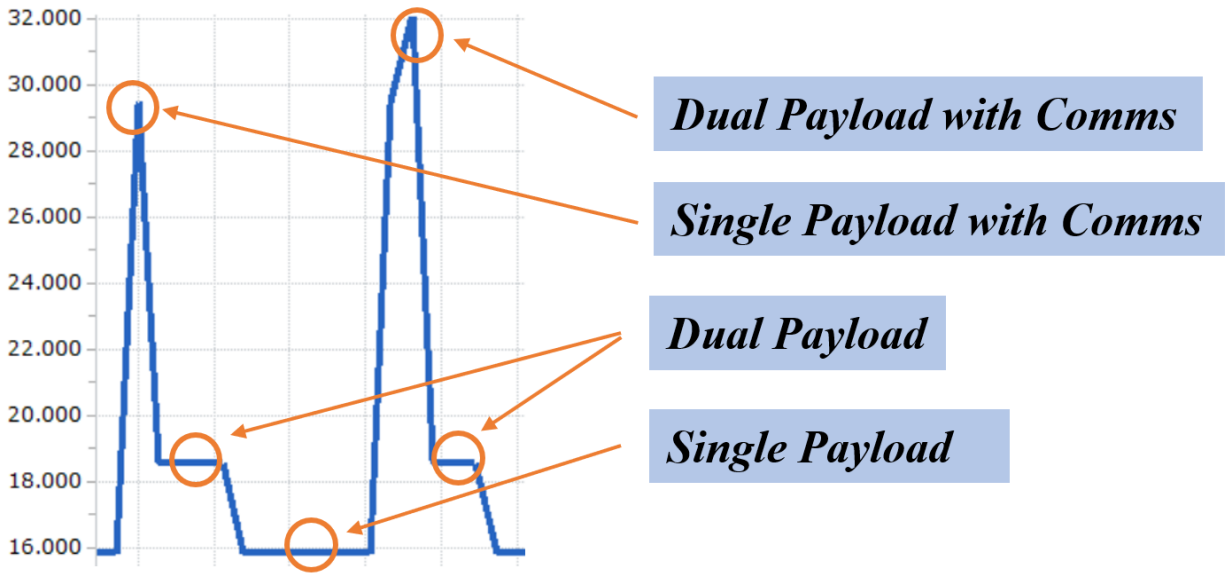

Figure 41: LAICE-F operational modes verified in the power consumption plot that FreeFlyer generated.

During the daylit portion of the orbit, the orbit average power consumed is estimated to be 17.18 W based on the Single Payload mode and the Single Payload with Comms mode. During the eclipse portion of the orbit, an orbit average power is estimated to be 19.92 W based on the Dual Payload mode and the Dual Payload with Comms mode.

By dividing the orbit average energy used during eclipse (19.92 * 0.744 = 14.82 WHrs) deprecated by the DOD (20%) [35], the total capacity of the batteries required to support operations of the satellite during the eclipse is found (90.8 WHrs). By dividing this value by the energy capacity of a single battery cell (11.59 WHrs), the number of battery cells comes out to 7.82 battery cells. Because batteries come in even integer numbers, the number of battery cells is rounded up to eight. Notably, the PDR design for LAICE-F matches this result [7].

The charging power calculation is slightly more involved. It is based on the average power consumption during eclipse (14.82 WHrs) and the time the satellite is charged on the sunlit side of the orbit (0.816 hours). Multiplying these two values predicts that the batteries must charge at a rate of 18.16 W to reach full capacity before the next eclipse period. Thus, to keep the satellite operational and to charge the batteries, it must generate a total of 35.34 W.

LAICE-F has a two-year mission design requirement. Given that solar cells deteriorate at an average annual rate of 3% [34], it's crucial to factor in this degradation in the solar panel's sizing to guarantee the satellite's battery charging and operation as it nears its end of life. To account for the 3% degradation factor per year for two years, the satellite needs to produce 37.49 W at the beginning of its mission.

Going from power generation requirements to solar cell specifications, the 37.49 W generation requirement requires a 31.22 solar cell configuration when applying the worst-case solar constant. However, similar to batteries, manufacturers do not produce deployable solar array

panels in fractions. Each deployable array comprises 14 cells (28 per pair), necessitating the use of a pair of double deployable panels to meet the system power requirements. This results in a significant margin for the design. The plot below shows the power generated and energy stored from eight batteries and 61 cells configured in a double-deployable solar array design (Figure 42).

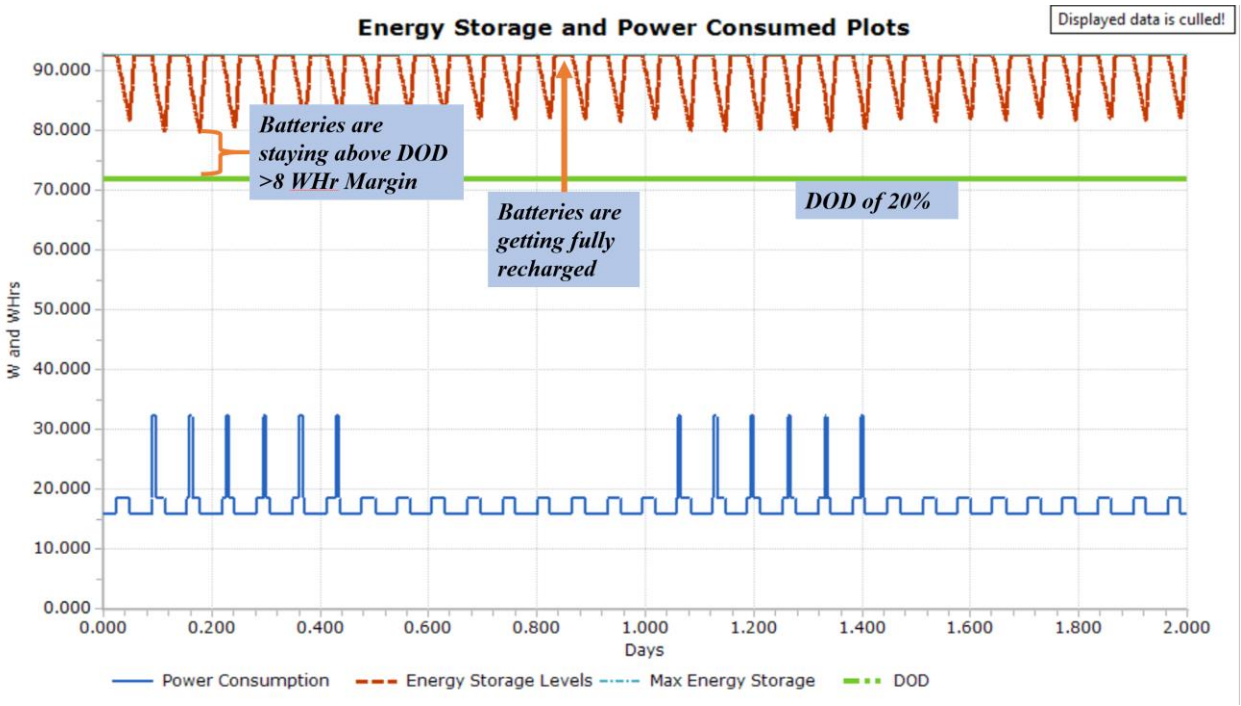

Figure 42: Energy storage and power consumption plot generated by FreeFlyer in the 61 solar cell and 8 battery configuration. Notable features include the margin between the battery charge levels and the DOD, as well as the rate at which the batteries recharge.

Several design features are evident in the plot. The excess capacity of the solar array can be seen in the saturation of the battery "flat-tops" at the point where the batteries have completed recharging for eclipse but the satellite is still on the sunlit side of the orbit, generating excess energy (Figure 43).

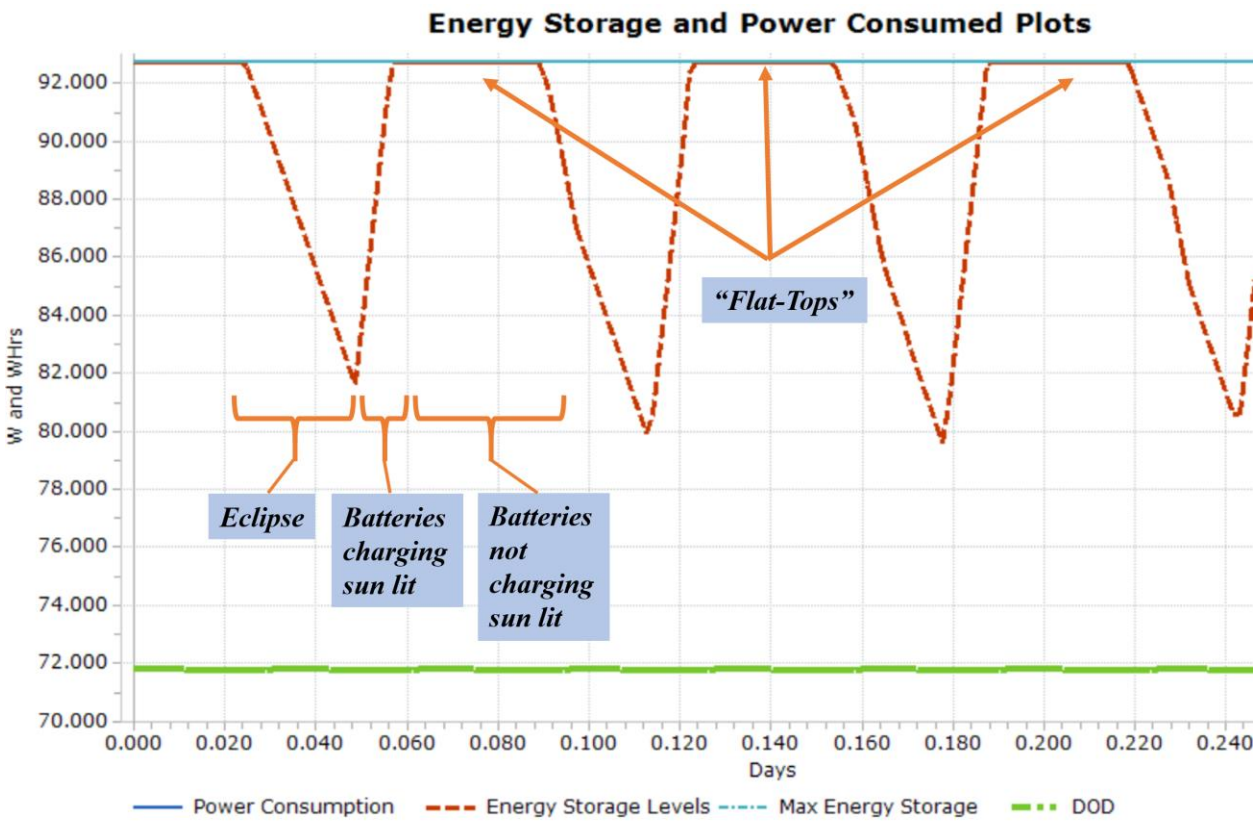

Figure 43: Excess energy production is evident in the "flat-tops" of the energy storage plot. Labels show the proportion of the time of eclipse, batteries charging in the sunlight, and when the batteries aren't charging in the sunlight.

To summarize the results of this design exercise carried out by the FreeFlyer tool, the data generated by FreeFlyer closely matches the initial predictions for the performance of a 61 solar cell and eight battery cell power subsystem (Table 7). The difference in values comes from FreeFlyers' ability to accurately model the attitude and orbital dynamics of the satellite when computing the satellite's power-generating capability.

Table 7: Comparison between the FreeFlyer power calculations and the predicted calculations.

Parameter	Predicted Value	FreeFlyer Value
Solar Cell Quantity	61	61
Battery Cell Quantity	8	8
Max Power Generation	75.16 W	74.73 W

Data and Communications

Now that it has been determined that there is sufficient power for the payloads to run whenever they want in their operating domains, the total amount of data storage necessary to store the payload and telemetry data before downlink needs to be determined.

The first value used in determining the size is the average rate of data generation. Based on a calculated 52.3% daylit and 47.7% eclipse orbit breakdown, the LAICE-F PDR [7] determined that the average data rate for the satellite bus, VT payload, UIUC payload, and overhead is equal to 7.312 kb per second. In one day, 631.72 Mb of data are generated.

The next value to determine is the total time and the frequency of downlink sessions. In a 24-hour period, six consecutive orbits are available for the satellite to downlink data. These passes have an average length of 259.2 seconds. With the downlink sessions happening in sequential orbits and the average orbit being 93.6 minutes long, there is a 14.64-hour period where no data can be downlinked, but data is still collected. During this period, at the average data generation rate, 385.371 Mb of data will be generated. This value is the maximum data storage needed for nominal satellite operations.

The downlink rate of the S-band radio was determined to be 3.3 Mb per second through a link budget analysis calculation with a 10 dB margin. To downlink all the data from the pass,

116.78 seconds of downlink is required. With the average pass being over two to three times this length, LAICE-F can comfortably downlink all the data on any pass.

When simulating this system in FreeFlyer (Figure 44), the amount of downlinked data is accurately calculated to be 385 Mb for the largest downlink. Each downlink was also completed in less than two minutes.

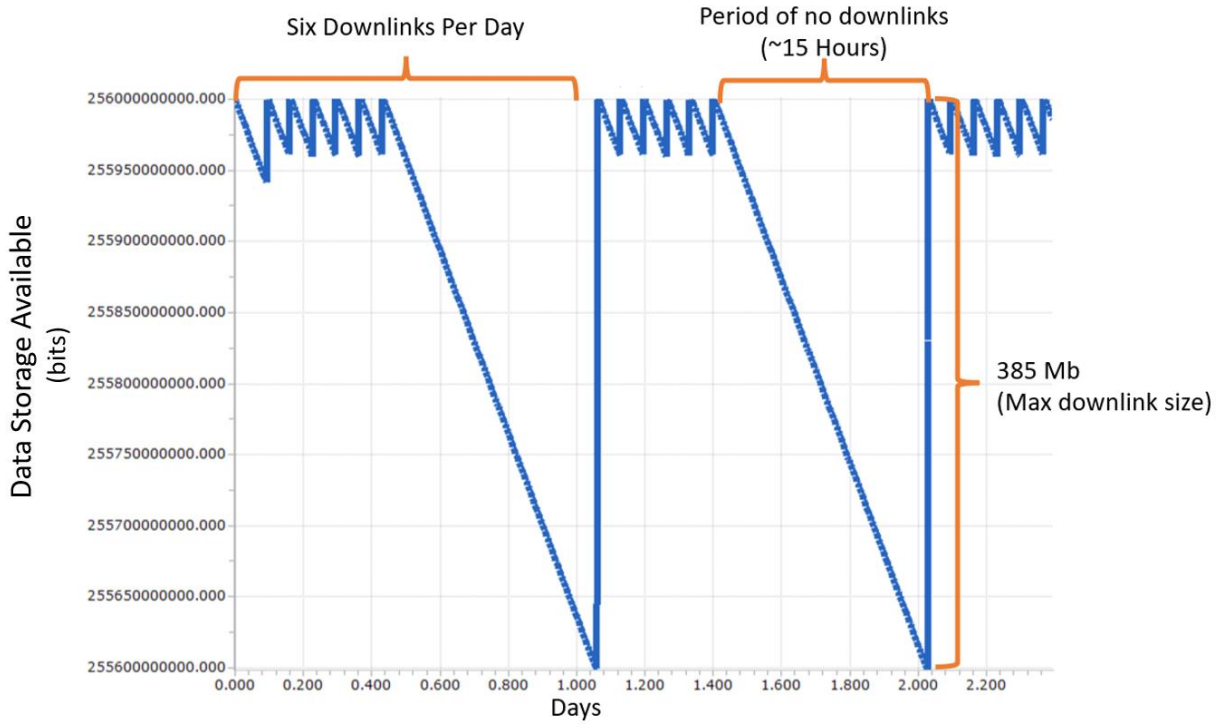

Figure 44: The data storage availability over the course of two days.

The data generated by FreeFlyer matches the initial predictions for the quantity of data storage needed and the radio's capabilities (Table 8).

Table 8: Comparison between the FreeFlyer calculations and the predicted calculations.

Parameter	Predicted Value	FreeFlyer Value
Max Storage Needed	385.371 Mb	385 Mb
Time to Complete Largest Downlink	116.78 s	120 s

Environmental Forces

Another question a satellite designer may have is: What is the lifetime of the satellite? To compute the orbital lifetime, several factors must be integrated to determine the delta-v being applied to the satellite by atmospheric drag. Drag is a function of the solar flux (e.g., how active the Sun is), orbital altitude, and satellite configuration. FreeFlyer may be used to take all of these factors into account to estimate the time for LAICE-F's orbit to decay to 250 km, which is considered the entry interface for satellites with less than a day or two left in orbit [29].

To calculate the drag [26] and lift [28] forces, Equations 14 and 15, respectively, are used. The satellite's relative velocity is a common input between both the drag and lift calculations. At this altitude, a relative velocity of 7850 m/s is used. The second shared variable is the atmospheric density. Referencing the MSISE-90 Model [43] for atmospheric densities, a value of 4.02 E-11 kg/m^3 is found for the deployment altitude of 420 km. The coefficient of drag is 2.2 [27], and the coefficient of lift is 1.2. For the LAICE-F platform configuration [7], the drag area is 0.022 m^2, and the lift area is 0.3036 m^2. This results in a calculated drag force of 0.0000599 N and a calculated lift force of 0.000451 N. Table 9 compares the estimated forces to the FreeFlyer tool's calculations.

Table 9: Summary of predicted versus FreeFlyer-calculated forces.

Force	Predicted Value	FreeFlyer Value
Gravity	215.43 N	215.40 N
SRP	0.00000193 N	0.00000190 N
Drag	0.0000599 N	0.000028 - 0.000063 N
Lift	0.000451 N	0.00124 - 0.00645 N

Figure 45 illustrates the LAICE-F drag and lift forces calculated by FreeFlyer over a couple of orbits. FreeFlyer integrates these forces to calculate the satellite's velocity change for each orbit.

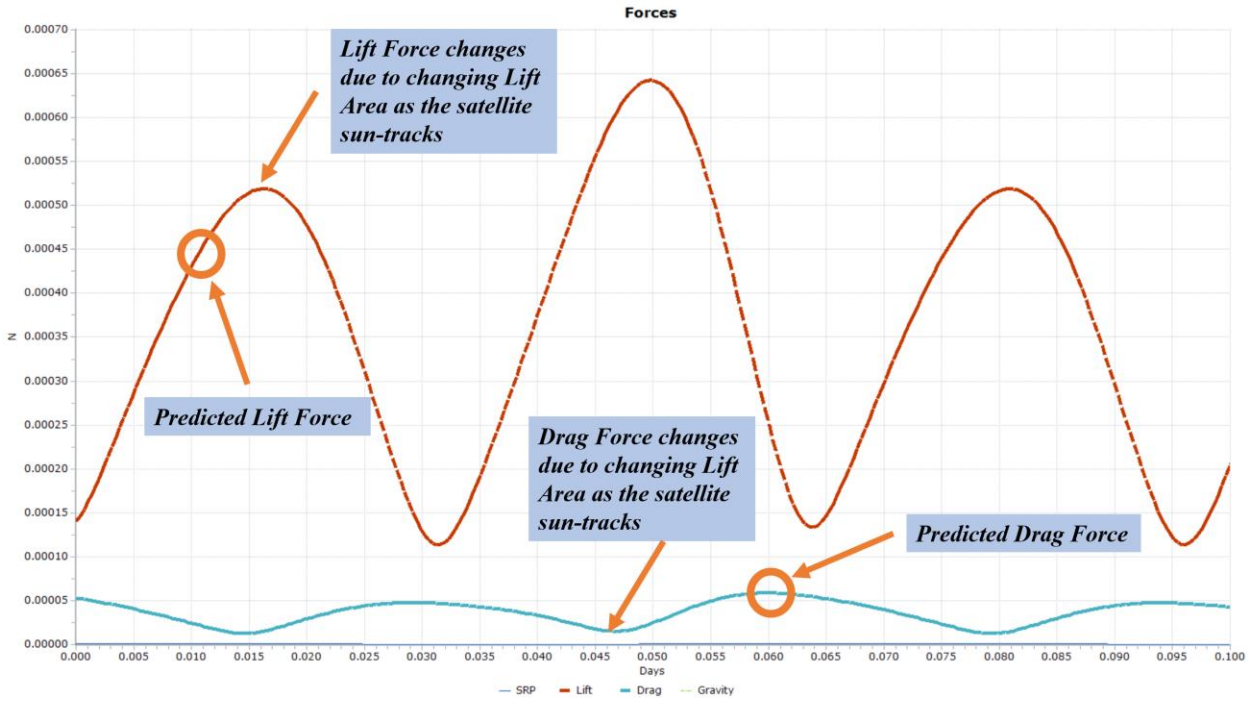

Figure 45: Drag and Lift forces are modeled in FreeFlyer; oscillating behavior results from dynamic lift and drag area calculations.

Integrating the impact of these forces on the satellite results in an orbital decay plot, as illustrated in Figure 46. FreeFlyer provides an estimated lifetime of 327 days, which is in line with the historical lifetimes for 6U CubeSats launched from the ISS [29].

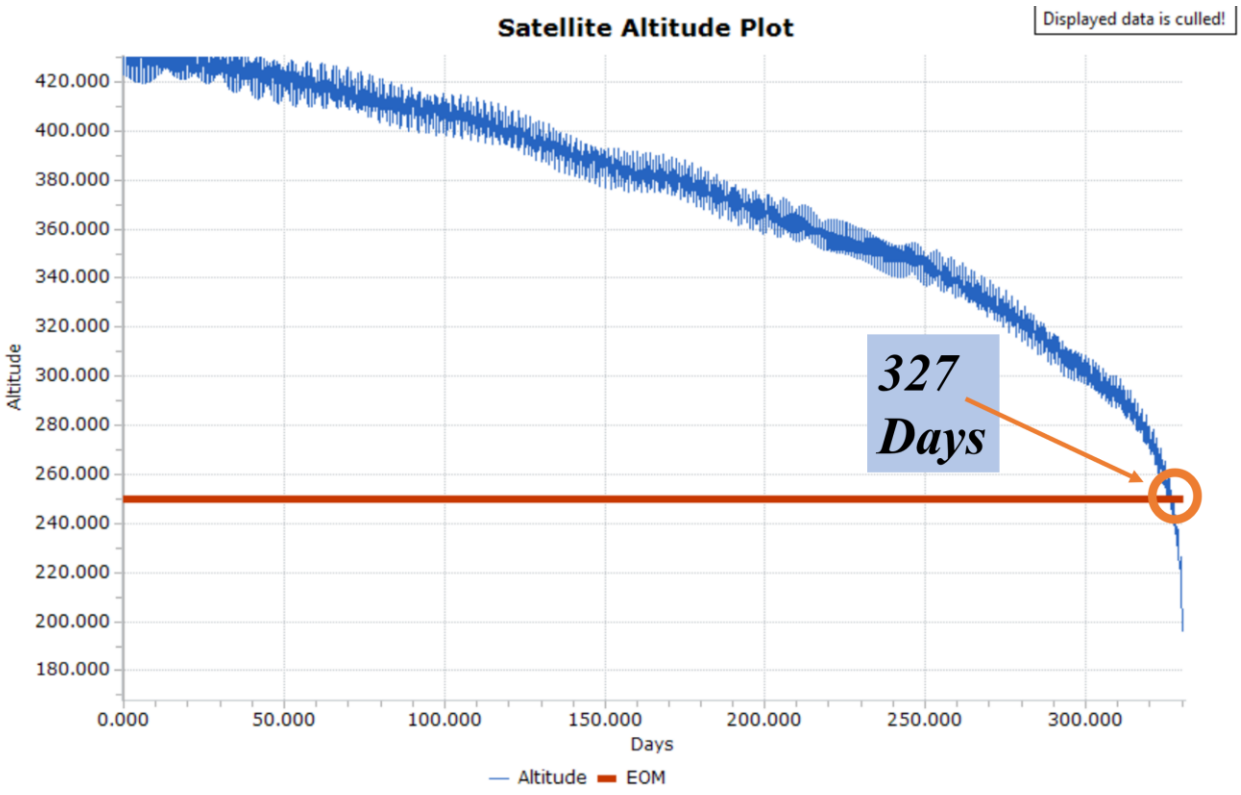

Figure 46: Orbital decay plot generated in FreeFlyer for LAICE-F.

CHAPTER 6: CONCLUSION AND FUTURE WORK

The FreeFlyer-based CubeSat orbital dynamics simulation tool and planform configuration scripts developed for this thesis are used to model the performance of a small satellite. This tool models the satellite's orbital position, attitude, power generation and energy storage, data generation and storage, and communications downlinks to assess the satellite's performance. The tool fulfills the requirement of allowing the user to swiftly evaluate a point solution, enabling the rapid generation of design iterations to identify the optimal configurations for a given mission.

With this base functionality, additional add-ons are envisioned for future work, adding to the utility of this tool for use in the satellite design process. Adding a subsystem performance optimizer based on user-defined parameters would remove the need for the user to interpret the results and update the configuration manually when using the tool. Additional components could be modeled, including various types of attitude determination and control devices, to add higher levels of fidelity to the simulation. Thermal analysis capabilities would also increase the accuracy of the power generation model and allow for the verification of thermally sensitive missions.

In summary, the custom tool developed for this thesis may be used by satellite designers when iteratively evaluating satellite configurations and performance early in the design process. The tool excels in both the accuracy of the simulations and in reducing the time it takes to perform analysis through the incorporation of modeling scripts within the FreeFlyer platform. It has been made available and will find considerable use in future LASSI design exercises.

REFERENCES

[1] The CubeSat Program, Cal Poly SLO, "CubeSat Design Specification Rev 14.1," Feb. 2022. Available: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779/CDS+REV14_1+2022-02-09.pdf

[2] J. Groh, "What are SmallSats and CubeSats? - NASA," *NASA*, Mar. 11, 2024. Available: https://www.nasa.gov/what-are-smallsats-and-cubesats/

[3] T. Roberts, "Space Launch to Low Earth Orbit: How Much Does It Cost?," *Aerospace Security*, Sep. 02, 2020. Available: https://aerospace.csis.org/data/space-launch-to-low-earth-orbit-how-much-does-it-cost/

[4] S. Caldwell, "NASA CubeSats Play Big Role in Lunar Exploration - NASA," *NASA*, Feb. 25, 2020. Available: https://www.nasa.gov/directorates/stmd/small-spacecraft-technology-program/nasa-cubesats-play-big-role-in-lunar-exploration/

[5] *CubeSat 101 - Basic Concepts and Processes for First Time CubeSat Developers* . NASA, 2017. Available: https://www.nasa.gov/wp-content/uploads/2017/03/nasa_csli_cubesat_101_508.pdf

[6] "FreeFlyer® Software," *a.i. solutions*. Available: https://ai-solutions.com/freeflyer-astrodynamic-software/

[7] I. Anderson, "Preliminaryt Design of the LAICE-F Mission to Study Atmospheric Gravity Waves," M.S. Thesis, university of Illinois Urbana-Champaign, 2023. Available: https://www.ideals.illinois.edu/items/128733

[8] A. McInnes, D. Harps, J. Lang, and C. Swenson, "A Systems Engineering Tool for Small Satellite Design," in *15th Annual AIAA/USU Conference on Small Satellites*, USU, 2021. Available: https://s3vi.ndc.nasa.gov/ssri-kb/static/resources/A%20Systems%20Engineering%20Tool%20for%20Small%20Satellite%20Design.pdf

[9] Microsoft, "Microsoft Excel, Spreadsheet Software," *www.microsoft.com*. Available: https://www.microsoft.com/en-us/microsoft-365/excel

[10] Y.-K. Chang, K.-L. Hwang, and S.-J. Kang, "SEDT (System Engineering Design Tool) development and its application to small satellite conceptual design," *Acta Astronautica*, vol. 61, no. 7–8, pp. 676–690, Oct. 2007, doi: https://doi.org/10.1016/j.actaastro.2007.01.067

[11] "SSCM | The Aerospace Corporation," *aerospace.org*, Nov. 14, 2019. Available: https://aerospace.org/sscm

[12] D. Barnhart, T. Kichkaylo, and L. Hoag, "SPIDR: Integrated Systems Engineering DesigntoSimulation Software for Satellite Build," in *CSER 2009*, Jan. 2009. Available: https://www.researchgate.net/publication/228735646_SPIDR_Integrated_Systems_Engineering_Design-to-Simulation_Software_for_Satellite_Build

[13] "Ansys STK | Digital Mission Engineering Software," *www.ansys.com*. Available: https://www.ansys.com/products/missions/ansys-stk

[14] G. Ridolfi, E. Mooij, and S. Corpino, "A System Engineering Tool for the Design of Satellite Subsystems," presented at the AIAA Modeling and Simulation Technologies Conference, AIAA, Aug. 2009. doi: https://doi.org/10.2514/6.20096037

[15] M. Such, "STA," *GitHub*, May 17, 2024. Available: https://github.com/toxygen/STA

[16] M. Novara, R. Reinhard, E. M. Rasel, and R. A. Henderson, “CDF Study Report HYPER Hyper-Precision Atom Interferometry in Space,” Sep. 2000. Available: https://esamultimedia.esa.int/docs/HYPER_Report.pdf
[17] “State-Of-The-Art Small Spacecraft Technology,” NASA, Feb. 2024. Available: https://www.nasa.gov/wp-content/uploads/2024/03/soa-2023.pdf?emrc=8ad1a1
[18] “GMAT Software Details,” *software.nasa.gov*. Available: https://software.nasa.gov/software/GSC-17177-1
[19] “About Orekit,” *www.orekit.org*. Available: https://www.orekit.org/
[20] R. H. Qureshi and S. P. Hughes, “Preparing General Mission Analysis Tool for Operational Maneuver Planning of the Advanced Composition Explorer Mission,” in *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2014. doi: https://doi.org/10.2514/6.2014-4148
[21] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, “RigidBody Attitude Control,” *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 30–51, doi: https://doi.org/10.1109/MCS.2011.940459
[22] R. Goldman, “Understanding quaternions,” *Graphical Models*, vol. 73, pp. 21–49, Mar. 2011, doi: https://doi.org/10.1016/j.gmod.2010.10.004
[23] F Rohrlich, *From paradox to reality : our new concepts of the physical world*. Cambridge ; New York: Cambridge University Press, 1987.
[24] W. Adams and W. Hodge, “Influence of Solar Radiation Pressure on Orbital Eccentricity of a Gravity-Gradient Oriented Lenticular Satellite,” NASA Langly, 1965. Available: https://ntrs.nasa.gov/api/citations/19650009680/downloads/19650009680.pdf
[25] D. Skoulidou, F. Letizia, and S. Lemmens, “Estimation of the Uncertainties of the Orbital Lifetime of Space Debris,” in *European Conference on Space Debris*, ESA Space Debris Office, Apr. 2021. Available: https://conference.sdo.esoc.esa.int/proceedings/sdc8/paper/159/SDC8-paper159.pdf
[26] Edward Michael Gaposchkin and A. J. Coster, “Analysis of satellite drag,” *The Lincoln Laboratory Journal*, vol. 1, pp. 203–224, Jan. 1988.
[27] W. Vries, “CubeSat Drag Calculations,” Lawrence Livermore National Lab, Sep. 2010. Available: https://www.osti.gov/servlets/purl/1124870
[28] S. R. Omar, “Using Differential Aerodynamic Forces for CubeSat Orbit Control,” Jan. 2013.
[29] Sarat Chandra Nagavarapu, A. Chandran, and D. E. Hastings, “Orbital Decay Analysis for Debris Deorbiting CubeSats in LEO: A Case Study for the VELOX-II Deorbit Mission,” Jan. 2021.
[30] “EPS Sizing Tutorial.” Available: https://www.valispace.com/wp-content/uploads/2018/12/EPS-sizing-tutorial-1.pdf
[31] F. S. Johnson, “THE SOLAR CONSTANT,” *Journal of the Atmospheric Sciences*, vol. 11, no. 6, pp. 431–439, Dec. 1954, doi: https://doi.org/10.1175/1520-0469(1954)011%3C0431:TSC%3E2.0.CO;2. Available: https://journals.ametsoc.org/view/journals/atsc/11/6/1520-0469_1954_011_0431_tsc_2_0_co_2.xml?tab_body=pdf
[32] T. Flatley and W. Moore, “An Earth Albedo Model,” NASA, 1994. Available: https://ntrs.nasa.gov/api/citations/19940020024/downloads/19940020024.pdf

[33] "Modelling of Photovoltaic Solar Panel for Maximum Power Point Tracking," *International Journal of Science and Research (IJSR)*, vol. 5, no. 5, pp. 1769–1771, May 2016, doi: https://doi.org/10.21275/v5i5.nov163552

[34] SPGenie, "How Much Do Solar Panels Degrade Each Year?," *SolarPowerGenie.com*, Aug. 08, 2019. Available: https://solarpowergenie.com/how-much-do-solar-panels-degrade-each-year/

[35] E. Rodrigues, C. A. S. Fernandes, R. Godina, Abebe Worke, Bizuayehu, and J. Catalão, *NaS battery storage system modeling and sizing for extending wind farms performance in Crete*. 2014. doi: https://doi.org/10.1109/AUPEC.2014.6966547

[36] H. T. Friis, "A Note on a Simple Transmission Formula," *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, May 1946, doi: https://doi.org/10.1109/jrproc.1946.234568

[37] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, Oct. 1948, doi: https://doi.org/10.1002/j.1538-7305.1948.tb00917.x. Available: https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

[38] "International Space Station - NASA." Available: https://www.nasa.gov/reference/international-space-station/

[39] "30% Triple Junction GaAs Solar Cell Assembly Type: TJ Solar Cell Assembly 3G30A Improved Voltage at Maximum Power Point," 2016. Available: https://www.azurspace.com/images/products/0003401-01-01_DB_3G30A.pdf

[40] "CubeSat Electrical Power System EPS," *NanoAvionics*. Available: https://nanoavionics.com/cubesat-components/cubesat-electrical-power-system-eps/

[41] "SRS-3 Full-duplex S-band Transceiver," *www.satlab.com*. Available: https://www.satlab.com/products/srs-3/

[42] "S-Band Patch Antenna - ISM - Microstrip Antenna | SatCatalog," *www.satcatalog.com*. Available: https://www.satcatalog.com/component/s-band-patch-antenna-ism/

[43] Rocket and Space Technology, "Properties of Standard Atmosphere," *Braeunig.us*, 2019. Available: http://www.braeunig.us/space/atmos.htm

APPENDIX A: FREEFLYER CODE AND USAGE DOCUMENTATION

The tool utilized for the analysis presented in this thesis as well as usage documentation can be requested from the author or accessed at https://github.com/meh4/FFSS_AIO

APPENDIX B: LAICE-F TELEMTRY DATA BUDGET

Table 10 displays LAICE-F's data budget that was utilized to determine the data generation rates of the satellite bus. The Payload generation rates and telemetry totals are provided in Table 5.

Table 10: LAICE-F Telemetry Data Budget

Data Field	Units	Data Format	Cadence
Mag Time Tag	Unix Time	Unsigned long int	
Magnetorquer X-1 Tag	Amps	Double	1 second
Magnetorquer X-2 Tag	Amps	Double	1 second
Magnetorquer Y-1 Tag	Amps	Double	1 second
Magnetorquer Y-2 Tag	Amps	Double	1 second
Magnetorquer Z-1 Tag	Amps	Double	1 second
Magnetorquer Z-2 Tag	Amps	Double	1 second
Attitude Time Tag	Unix Time	Unsigned long int	
Attitude Pitch	Deg	Double	1 second
Attitude Yaw	Deg	Double	1 second
Attitude Roll	Deg	Double	1 second
Propagated Orbit Time Tag	Unix Time	Unsigned long int	
On-board Propagated Altitude	Km	Double	1 second
On-board Propagated Latitude	Deg	Double	1 second
On-board Propagated Longitude	Deg	Double	1 second
Reconstructed Orbit Time Tag	Unix Time	Unsigned long int	
Reconstructed Altitude	Km	Double	1 second
Reconstructed Latitude	Deg	Double	1 second
Reconstructed Longitude	Deg	Double	1 second
Satellite Time	Unix Time	Unsigned long int	1 second
Orbit Number	#	Long int	1 second
Battery SOC	V/%	Double	1 second
VT Heater Cmd State	Boolean	Unsigned int	1 second
VT Heater Power	Amps	Double	1 second
VT payload Power Cmd State	Boolean	Unsigned int	1 second
VT Payload Power	Amps	Double	1 second
Battery 1 Temperature Sensor	Deg C	Unsigned int	10 seconds
Battery 2 Temperature Sensor	Deg C	Double	10 seconds
Battery 3 Temperature Sensor	Deg C	Double	10 seconds
Battery 4 Temperature Sensor	Deg C	Double	10 seconds
Radio Temperature	Deg C	Double	10 seconds

CDH Temperature	Deg C	Double	10 seconds
Radio Power Cmd	Boolean	Unsigned int	1 second
Gyro 1	Deg / S	Double	1 second
Gyro 2	Deg / S	Double	1 second
Gyro 3	Deg / S	Double	1 second
Gyro 4	Deg / S	Double	1 second
Wheel Speed 1	RPS	Double	1 second
Wheel Speed 2	RPS	Double	1 second
Wheel Speed 3	RPS	Double	1 second
Wheel Speed 4	RPS	Double	1 second
Wheel Current 1	Amps	Double	1 second
Wheel Current 2	Amps	Double	1 second
Wheel Current 3	Amps	Double	1 second
Wheel Current 4	Amps	Double	1 second
Battery Current	Amps	Double	1 second
Battery Voltage	Voltage	Double	1 second
Solar Panel Current	Amps	Double	1 second
Battery Temp 5	Deg C	Double	1 second
Battery Temp 6	Deg C	Double	1 second
Battery Temp 7	Deg C	Double	1 second
Battery Temp 8	Deg C	Double	1 second
Array 1	W	Double	1 second
Array 2	W	Double	1 second
Array 3	W	Double	1 second
Array 4	W	Double	1 second
Array 5	W	Double	1 second