Optimal Engine Selection and Trajectory Optimization using Genetic Algorithms for Conceptual Design Optimization of Reusable Space Launch Vehicles

Steven Cory Wyatt Steele

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

Master of Science
in
Mechanical Engineering

February 19, 2015
Blacksburg, Virginia

Keywords: Trajectory Optimization, Engine Selection, Reusable Launch Vehicle (RLV), Two Stage to Orbit (TSTO)

Optimal Engine Selection and Trajectory Optimization using Genetic Algorithms for Conceptual Design Optimization of Reusable Space Launch Vehicles

Steven Cory Wyatt Steele

Abstract

Proper engine selection for Reusable Launch Vehicles (RLVs) is a key factor in the design of low cost reusable launch systems for routine access to space. RLVs typically use combinations of different types of engines used in sequence over the duration of the flight. Also, in order to properly choose which engines are best for an RLV design concept and mission, the optimal trajectory that maximizes or minimizes the mission objective must be found for that engine configuration. Typically this is done by the designer iteratively choosing engine combinations based on his/her judgment and running each individual combination through a full trajectory optimization to find out how well the engine configuration performed on board the desired RLV design.

This thesis presents a new method to reliably predict the optimal engine configuration and optimal trajectory for a fixed design of a conceptual RLV in an automated manner. This method is accomplished using the original code *Steele-Flight*. This code uses a combination of a Genetic Algorithm (GA) and a Non-Linear Programming (NLP) based trajectory optimizer known as GPOPS II to simultaneously find the optimal engine configuration from a user provided selection pool of engine models and the matching optimal trajectory. This method allows the user to explore a broad range of possible engine configurations that they wouldn't have time to consider and do so in less time than if they attempted to manually select and analyze each possible engine combination.

This method was validated in two separate ways. The codes ability to optimize trajectories was compared to the German trajectory optimizer suite known as ASTOS where only minimal differences in the output trajectory were noticed. Afterwards another test was performed to verify the method used by *Steele-Flight* for engine selection. In this test, *Steele-Flight* was provided a vehicle model based on the German Saenger TSTO RLV concept and models of turbofans, turbojets, ramjets, scramjets and rockets. Steele-Flight explored the design space through the use of a Genetic Algorithm to find the optimal engine combination to maximize payload. The results output by *Steele-Flight* were verified by a study in which the designer manually chose the engine combinations one at a time, running each through the trajectory optimization routine to determine the best engine combination. For the most part, these methods yielded the same optimal engine configurations with only minor variation.

The code itself provides RLV researchers with a new tool to perform conceptual level engine selection from a gathering of user provided conceptual engine data models and RLV structural designs and trajectory optimization for fixed RLV designs and fixed mission requirements.

Acknowledgements

There are many people that have helped me over the years to get to where I am now so I had to do my best in limiting this list.

Firstly, I would like to thank Dr. Walter F. O'Brien for giving me a chance to "prove my stuff" as a graduate student and letting me work on a creative and interesting thesis topic. He has been very kind and patient with me over the years and has always been a source of motivation to continue and do the best I could. Thank you, Professor.

I would like to thank the creators of ASTOS for allowing me to use their software to compare my software results with in order to verify its accuracy and to acknowledge Dr. Joachim Kurzke for the use of GasTurb11 for all my Turbine engine modeling needs.

I would like to thank Dr. Anil V. Rao and Dr. Michael A. Patterson of Florida University for their code GPOPS-II which really helped me out of a pickle when trying to perform decent trajectory optimization and for the over-the-phone support from Dr. Rao.

I would like to thank Dr. John R. Olds, Dr. John E. Bradford and Georgia Tech for the use of the executable version of the code SCCREAM for RBCC engine modeling. I would like to thank both Dr. Olds and Dr. Bradford and all of SpaceWorks Enterprises Inc. for allowing the academic use of RedTop-Lite for all my rocket modeling needs.

I would like to thank my labmates for all their support, gags, advice, StarTrek nerd debates and arguments over comic book movies and occasional hard work. This includes: Tony Ferrar, Bill "Schneckums", Justin "Beiber", Todd "Double D" Pickering, Chaitanya Halbe, German Montalvo-Catano and Kathleen Tran.

Now for the personal acknowledgments. I have a whole slew of teachers from over the years to acknowledge but they won't all fit. So here is an attempt: Thank you Mary Jane Steggal, Paul Hatter, Scott Miller and finally, Diane Mehrten, my first grade teacher for her loving encouragement years after leaving first grade.

I would like to thank the engineering teachers at the Lyles College of Engineering at California State University Fresno, my undergraduate school, for their incredible encouragement, laughs, tough assignments and guidance. Personally, I would like to thank Dr. Satya Mahanty, Dr. Maria Sanchez, Dr. Ira Sorenson and Mr. Walter Mizuno. I had a lot of fun there and still believe in the importance of applying engineering principles we learn in the numerous design clubs available.

To my friends in Blacksburg, Adam Czesnowski, Caleb and Christina Simon, Brad Miller, Bill Jones, Hanni and Andrea Abiyounes, Steve and Ali Trice and Charles and Yvan Dean and many others, thank you for your support and friendship over the years. To my best friend Peter D. Lee, thanks man. I couldn't have completed this work without you.

Also a personal thanks go to my friends Carl and Cheryl J. Morrison. Thank you for your wisdom and guidance. They were much needed.

Finally to my family. I could not have been blessed with a more loving family than the one I have. To my 3 brothers, Alex, Justin and Patrick Steele: you know that I love you and thank you for being patient with me. Thank you again for your support. To my two cousins Jared and Joseph Zurita: thank you both for your brotherly love and fine examples. To my Uncle Tommy and Aunt Diana Zurita; thank you for always looking out for me. To all my many other cousins, aunts and uncles: Lots of Love! Most importantly, I must thank my parents David and Patricia Steele… I guess they did some important things for me, like raise me, teach me, feed me, support me and keep me going when I was feeling down. I have the most loving parents anyone could ask for and I am proud to be their son. I love you both so much.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms

RLV	Reusable Launch Vehicle
TSTO	Two Stage to Orbit
GTOW	Gross Takeoff Weight
RBCC	Rocket Based Combined Cycle Engine
TBCC	Turbine Based Combined Cycle Engine
AFIT	Air Force Institute of Technology
AFRL	Air Force Research Lab
LEO	Low Earth Orbit
AOA	Angle of Attack
GA	Genetic Algorithm
NLP	Non-Linear Programming or Non-Linear Program
T/W	Thrust to Weight Ratio
DMSJ	Dual Mode Scramjet
TJ	Turbojet
TF	Turbofan
RJ	Ramjet
SJ	Scramjet
PL	Payload
PR	Pressure Ratio
LPC	Low Pressure Compressor
HPC	High Pressure Compressor
BPR	Bypass Ratio

Symbols

δ	Declination
R	Radius from the center of Earth
υ	Velocity relative to Earth's surface
γ	Flight Path Angle
χ	Velocity Azimuth
Ω_E	Earth's rotational Speed
θ	True anomaly
a	Semi-major axis
e	Eccentricity
Ω	Right ascension of the ascending node
i	Orbital Inclination
ω	Argument of the periapsis
t	time
F	Function
α	Angle of Attack
Φ	Engine Throttle
$\emptyset$	Penalty function
r	multiplier
G	Heaviside Operator
g	constraint violation function
$f(x)$	Objective function

Chapter 1- Introduction

1.1 Background

Reliable access to space has been a goal for the past half-century. Today, space access not only serves a national interest, but commercial interests as well. However, launching humans and/or cargo into space comes at great cost in terms of fuel costs, mission turn-around-time, and purchase of new expendable launch equipment.

Currently, payloads are launched into space using massive rocket based systems that are boosted to a high altitude using large solid rocket boosters that are then detached and fall into the ocean and typically are not reused. The spacecraft's themselves also separate into multiple stages, all of which are typically not reused (the major exception being the Space Shuttle and planned Space X systems). Besides this, all current Expendable Launch Vehicles (ELVs) launch vertically upward, which requires a complex launch pad setup to position the rockets in an upright position. This further increases costs and time between missions.

Due to the negative results of continued use of ELVs, Reusable Launch Vehicles (RLVs) that use air breathing propulsion systems have been a popular research topic since the space race of the 1960's. Their appeal lies in their use of launch sites which require only a runway, fuel efficient reusable air-breathing engines to get through the atmosphere and their reusable structures that can reduce time between missions.

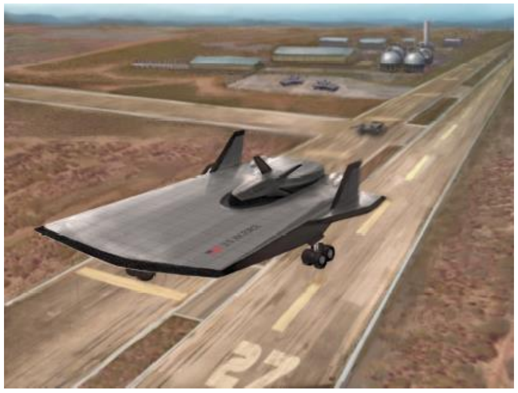

Figure 1. Artist depiction of the QuickSat TSTO vehicle. Eklund, D., *Quicksat: A Two Stage to Orbit Reusable Launch Vehicle Utilizing Air Breathing Propulsion for Responsive Space Access*, in *Space 2004 Conference and Exhibit*. 2004, American Institute of Aeronautics and Astronautics [5]. Used with permission of Ashley Russ, SpaceWorks Enterprises, Inc., 2015.

Many recent research studies into reusable space launch systems tend to focus on Two-Stage-To-Orbit (TSTO) launch vehicles such as those seen in references [5, 8-10] to name a few. These systems, as pictured in Figure 1, are intended to use a first stage booster (typically powered by air-breathing engines) to carry the final stage up to a reasonable altitude and Mach number (typically to the limit of air breathing operational conditions) before launching the second stage which will reach a desired orbit. The booster stage will then turn around and fly back to base where it can be prepared for another launch. The upper stage could be either reusable or expendable depending on the needs of the mission.

Despite all the benefits mentioned above, none of the many previous attempts to design a low cost, reusable, space transportation system have ever made it to the assembly line. This has been due primarily to costs of development, projected maintenance costs for the vehicles and dependence on propulsion technology that has not been fully developed. This thesis will focus on the propulsion issues faced.

1.2 Propulsion Challenge

One significant challenge to the RLV designer is the choice of engines to power them. Air-breathing engines such as turbine engines and ramjets have specific impulse (Isp) values that are far superior to rocket engines since they can utilize atmospheric air for an oxidizer rather than carrying the massive weight of stored oxidizers. However, each type of air-breathing engine has a limited range over the Mach regime. They also vary in performance across different Mach numbers and altitudes and finally, they cannot operate outside of the atmosphere and thus cannot achieve orbit. Meanwhile rockets have much more consistent performance over a wide range of Mach numbers and can be used in the vacuum of space, but have very low Isp since they must carry their own oxidizer.

In addition to the above considerations, the designer must also consider new technologies on the rise that support both rockets and air-breathers. The scramjet (Super Sonic Combustion Ramjet) engine is still in the experimental phase, but could potentially bring the air-breathing Mach regime up to Mach 12 or higher. Also, in-flight oxygen collection systems are also being considered as a way to either supply rocket engines with liquid oxygen during flight or to store up for second stage Rocket engines [11, 12].

TSTO RLVs that use air-breathing engines can takeoff horizontally as “spaceplanes” [8] and require longer flights through multiple Mach regimes and thus would likely require engine configurations that are composed of multiple types of engines working in sequence (or parallel). Systems that combine turbine engines and ram/scramjets into a single housing are termed Turbine-Based-Combined-Cycle (TBCC) engines and those that combine rockets and ram/scramjet engines into a single flow path are called Rocket-Based-Combined-Cycle (RBCC) engines. In order for a designer to assess which engine configurations would best suit the needs of the mission and the RLVs structural and aerodynamic design, they must have some idea as to how there RLVs would perform with the different engine configurations. Each mission and aero/structural design could require a different engine configuration in order to maximize its payload and/or minimize its GTOW.

Comparing these different engine configurations is no easy task. In fact if the structural design concept for a particular RLV was held constant along with its mission objective, the best trajectory to achieve that mission goal would be determined by which engines were selected. Therefore, in order to really know which engine configuration is the best for a particular mission and vehicle design would require trajectory optimization for the selected engine configuration.

There have been many methods developed over the years that attempt find an optimal RLV engine configuration design. Each seems to tackle the problem from a different angle. However, when trying to determine the proper engines, the only method available is to try one engine combination at a time. Keep in mind that this involves deciding how many of each engine are on the RLV and in what sequence they should be used in. Needless to say, this can be a very time consuming task, especially since this step is one step of the overall iterative vehicle design method that includes the vehicle structure, aerodynamics and trajectory which are also affected by engine selection.

1.3 Objective of Thesis

The objective of this thesis is to provide RLV researchers with a powerful new tool, named *Steele-Flight* by the author, which will allow them to determine the proper choice of engines and how to use them for specified RLV structural and aerodynamic concepts and for desired missions in a fast organized manner. Determination of the proper engine choices is done by: 1) using a Genetic Algorithm to search through many possible combinations of engines to power a launch vehicle and 2) by finding the corresponding optimal trajectories for the engine configurations to allow optimum engine use with regard to the mission objectives. The method used in this study will have the objective of maximizing payload delivered to Low Earth Orbit (LEO).

Chapter 2- Review of Literature

Since design of the different aspects of RLVs has been a popular research topic for numerous decades, it is important at this point to review some of the more important contributions to this field of study by past researchers and to clarify the distinctiveness of this particular study.

2.1 Similar Studies

2.1.1 –2004 Air Force Institute of Technology (AFIT) study [8]

The goals of this study were very similar to the goals of the studies performed in this thesis as will be seen in the next chapter. This study focused on determining the performance of TSTO RLVs using multiple engine configurations in an effort to determine the best engine configurations with regard to maximizing payload delivered to orbit and minimizing vehicle inert mass. Data on aerodynamics, engine performance, trajectories and vehicle weights were all taken from available literature and data from the Air Force Research Laboratory (AFRL). The results of this test were obtained by testing the sensitivity of the payload delivered to orbit for each RLV configuration to variations in the vehicles stage inert mass fractions while holding the GTOW of the vehicle constant. Different takeoff methods were tested for some of the RLV configurations and all RLVs had their trajectories simulated with the industry standard trajectory optimization program known as POST, using the defining constraints from the literature. However, the effects of the variation of trajectory constraints on the RLV configurations were not tested thoroughly because the high sensitivity of the trajectory optimization program to small changes in trajectory constraints required large time investments to obtain reasonable results [8]. It is also of note that all the engine models used in this study were fueled by hydrocarbons. Also, the position of the engines on the vehicle models was ignored and all losses from engine airframe integration were assumed negligible.
This study yielded the following results:

- None of the RLVs using the RBCC engine model was able to achieve orbit due to the large inert mass of the RBCC system and possible non-optimal aspects of the engine model.
- RLVs using turbojets in the first stage must start from a horizontal takeoff, because vertical takeoff requires an extremely large amount of turbojets.
- The payload mass of RLVs using turbojets is very sensitive to the inert mass fraction of the second stage of the vehicle in comparison to the first stage.
- The RLV that offered the highest figures of merit was the design that was powered purely by rocket engines in both stages.

2.1.2 –2004 SpaceWorks RLV study [5]

In 2004, SpaceWorks Engineering Inc. (SEI) performed a study into the design of a TSTO RLV concept known as the Quicksat for both military and commercial space flight. The entire physical design of the vehicle, including its costs and trajectory, was modeled using a suite of high-fidelity analysis codes that were linked together using the integrated modeling framework known as ModelCenter© by Phoenix Integration Inc. This model was then used to perform a number of trade studies on the design to determine sensitivities to the Staging Mach number, alternative turbojet engines for the first stage and different propellants. Rather than being an analysis comparing multiple launch vehicle engine configuration concepts like the other studies mentioned, this study focused on just one design [5].

2.1.3 –2006 AFIT/AFRL study [13]

The goals of this study were to determine the best TSTO RLV designs relative to minimum vehicle empty weight and vehicle wetted area by varying: the engine combinations used in the RLV; the sequence the engines are used in the trajectory; the fuel options for the engines and whether the vehicle would takeoff vertically or horizontally. The RLV stage structures were designed differently depending on what engine configuration they used and these configurations were tested on multiple mission trajectories that had different objectives. The empty mass and the wetted area were chosen as the figures of merit for the study because these figures typically drive the cost of the vehicle and the cost of maintaining it. In every mission tested, vehicles taking off with rockets from a vertical takeoff position had lower vehicle empty masses and wetted areas than vehicles taking off with turbines from a horizontal takeoff position. Also, in both of these cases, the engine configurations that used the ramjet/scramjet engines on the second stage of the vehicle yielded the best results for lowering vehicle empty weight and wetted area [13].

The modeling of the vehicle structures, systems, fuel tanks, aerodynamics and engines was handled using the code known as Hypersonic System Integrated Design and Environment (HySIDE) developed by ASTROX Corporation. The code has built-in modeling programs that allow the user to create their own models for vehicle geometry, subsystems, tanks and engines, and also allows users to import in their own data models [13]. Data from previous research was used to model the engines in this study. The code includes routines that integrate all these models into a single vehicle model that satisfies all geometric interactions, scales vehicles and their engines in order to meet the thrust requirements of a given trajectory and analyzes the performance of a vehicle model with its engines over the entire trajectory. However, HySIDE is not fully automated and requires much user interaction during iterative design routines to converge on designs that satisfy the vehicle geometry and meet the requirements of the trajectory.

2.1.4 –2007 Auburn University Study by Douglas Bailey [14]

The dissertation by Douglas Bailey focused on complete vehicle optimization using a Genetic Algorithm (GA) with the goals of reaching a desired orbit and minimizing both the total vehicle weight and vehicle cost. The GA was used to optimize the actual physical dimensions of both the vehicle and the engines used to propel it along with the engine characteristics in order to find a complete vehicle model that was capable of achieving the desired orbit while minimizing the total weight of the vehicle. The workings of the engines themselves were modeled using custom built performance models and the objective function of the GA included a trajectory simulator that would fly the vehicle through the trajectory in an attempt to reach orbit. This thesis focused only on the design of multistage launch vehicles using both solid and liquid propellant rocket engines. This method did not perform trajectory optimization since the vehicles only flew along a ballistic trajectory in an attempt to reach the desired orbital altitude and velocity [14]. Also, while this method was useful for the design of rocket propelled launch vehicles, it would be limited if applied to launch vehicles using lifting ascent trajectories and air breathing engines. The reasons for this are as follows: rocket engines offer much more consistent performance during their flight through the atmosphere and are not subject to the same constraints and sensitivities as air-breathing engines, thus requiring much less trajectory controls; rocket vehicles are much easier to design than air-breathing vehicles due to much less complex geometry, less moving parts and not having to condition the incoming atmospheric air and finally the aerodynamic shapes of most rocket propelled vehicles are simple to model because they typically don't rely on lift in comparison to vehicles using aerodynamic lift because they typically use very basic cylindrical shapes.

2.2 ASTOS®

The code known as Aero-Space Trajectory Optimization Software (ASTOS), developed by ASTOS Solutions GmbH, is a program designed for the simulation and optimization of vehicle trajectories for just about any application. It contains a suite of optimization codes and databases gathered from previous research efforts. ASTOS allows users to upload vehicle models and engine models into the database or construct their own simplified engine models with built in modeling elements. ASTOS uses a GUI interface that allows users to define mission constraints, mission objectives, automated flight controls and perform trajectory simulation or optimization studies all with relative ease.

This code also comes with a number of example tutorials for new users to use to get familiar with the code. However, the example tutorial for an advanced air-breathing TSTO trajectory [15] is of great significance to this thesis, because it is used as a means to validate the trajectory optimization abilities of the code developed in this thesis. This tutorial includes information on the conceptual TSTO RLV known as the Saenger Launch Vehicle. ASTOS provides estimations of the vehicles physical attributes such as stage empty weights, allowable fuel mass, an aerodynamic library and data for the engine models. While this data was useful in performing experiments to verify the methodology used in this thesis, the data provided for the aerodynamic coefficients, while being similar to published data concerning the Saenger aerodynamic coefficients did not match up exactly with this published data [6, 10, 16]. These relatively small differences between the aerodynamic data provided by ASTOS and published data, particularly for the lift coefficient, are not of particular significance in this study.

2.3 Motivation of this Study

2.3.1 –Deficiencies of previous work

The research presented in this chapter shows that significant effort has gone into the conceptual design of TSTO RLVs and that useful tools such as HySIDE and ASTOS already exist to aid in these design efforts. However, the majority of the studies and tools seem to lack the ability to do completely automated analysis and instead require tedious manual interference by the user. Also, trajectory optimization was not significantly used, if at all, in the majority of the studies considered. Proper engine selection is sensitive to the trajectory that it flies, since such trajectories determine what Mach regime the vehicle flies at and the vehicle stage-separation times. In the studies considered, baseline trajectories were already available, but new RLV designs could require completely new trajectories. Without an optimal trajectory being known for each engine configuration, the true potential of each engine configuration cannot be known.

2.3.2 –Uniqueness of current study

The method used in this study will simultaneously look at multiple engine configurations mounted to a single RLV structure and optimize the trajectories used by these configurations to accomplish the desired mission. Traits of the engine configurations such as number of engines on board, types of engines used and the sequence in which they are used can be varied in order to find out the best conceptual engine configurations for a given RLV structural design and a given mission objective in a fast manner.

It is also acknowledged that RLV structures are designed with the type of propulsion systems that they will use in mind. This is done throughout the previous studies examined. Therefore, since this study is only focused on one vehicle structural concept, all the engines that will be considered for it must be compatible with the structure.

It is also noticeable that the figures of merit optimized in the studies mentioned above were usually concerned with minimizing the empty weight (that is, the weight of the vehicle without fuel) and geometric size of the launch vehicle used to put a specified payload into orbit. Since this study will only be concerned with one fixed vehicle geometry, the empty weight will not be of great concern. Therefore, the study will instead focus on the maximization of payload carried to orbit as the primary figure of merit. Since the cost of a space launch is greatly affected by the cost per pound of payload delivered to orbit, maximizing the amount of payload that a spaceplane at its base-size can lift can go a long way toward reducing space launch costs.

Chapter 3- Modeling Approach

3.1 Steele-Flight Description

3.1.1 –What is the purpose of this code?

The method for engine selection and trajectory optimization for Launch Vehicles used in this thesis is implemented through a MATLAB based code called Steele-Flight. The purpose of this code is to allow RLV researchers to have an automated method to determine at the conceptual level, the optimum choice of propulsion engine concepts for a chosen aerospace vehicle with a defined mission objective.

This method will allow researchers to see the maximum payload potential of a vehicle at a nominal size before they were to begin scaling the vehicle and engines.

Like the 2004 AFIT study [8], this study shall also neglect engine airframe integration but will attempt to stay within reasonable limits of the allowable volume of the reference vehicle used in this study.

The experiments performed in this thesis are not meant to determine which vehicle design is the optimum choice for an RLV since data for the design of multiple launch vehicles was not currently available at the time of this study. Also due to the limited amount of validated modern engine data available in open literature, the turbine engine models and the RBCC engine models used in this study don't have a way to be validated completely. Their design was, however, guided by models from other conceptual studies. Therefore, the experiments performed are only for validating the method.

The method itself is not capable of designing engines or vehicles, therefore the results presented are governed by the validity of the models input into the code. A code such as this would likely be most useful being used in combination with a vehicle designing program such as HySIDE.

3.1.2 –The Method used by the code.

Selection of optimum propulsion is accomplished by providing the code with data libraries for the engine models, the launch vehicle aerodynamics, and information concerning the weight and volume of the launch vehicle. The user then must define the mission constraints and objective function. The goal of the optimization is to maximize or minimize the objective function for a fixed user defined mission by:

1. Varying the type of engine used during different periods of flight, the throttle settings of the engines, the physical number of each type of engine on the vehicle and the burn time of each engine while staying within the limits of the physical geometry of the vehicle with respect to engine size and volume of fuel that can be carried on board.
2. Using a trajectory optimization code to find a trajectory for the chosen engine configuration that maximizes or minimizes the chosen objective function while satisfying the mission constraints on flight conditions and final orbit.

The optimization routine can yield a group of the most promising engine configurations and the corresponding flight trajectories used by these configurations to accomplish their intended mission while optimizing the objective function.

This optimization is performed by means of a combination of the Genetic Algorithm built in to MATLAB® and a commercial MATLAB based code using Non-Linear Programming (NLP) methods known as GPOPS-II (General-Purpose Pseudospectral OPtimal Control Software) [17]. The source code for both of these optimizers was modified to suit the needs of this thesis.

3.2 Model Element Breakdown

Before continuing on to describe the steps taken to optimize the decision variables in the problem, it is first necessary to define some very important terms in order to understand the steps taken in the optimization method.

3.2.1 –Trajectory Modeling

Steele-Flight involves trajectory optimization and thus will be optimizing variables that define the trajectory. Before those variables are defined, certain terminology first needs to be made clear. The trajectory is broken into multiple *Phases* while the vehicle is broken into different *Stages*.

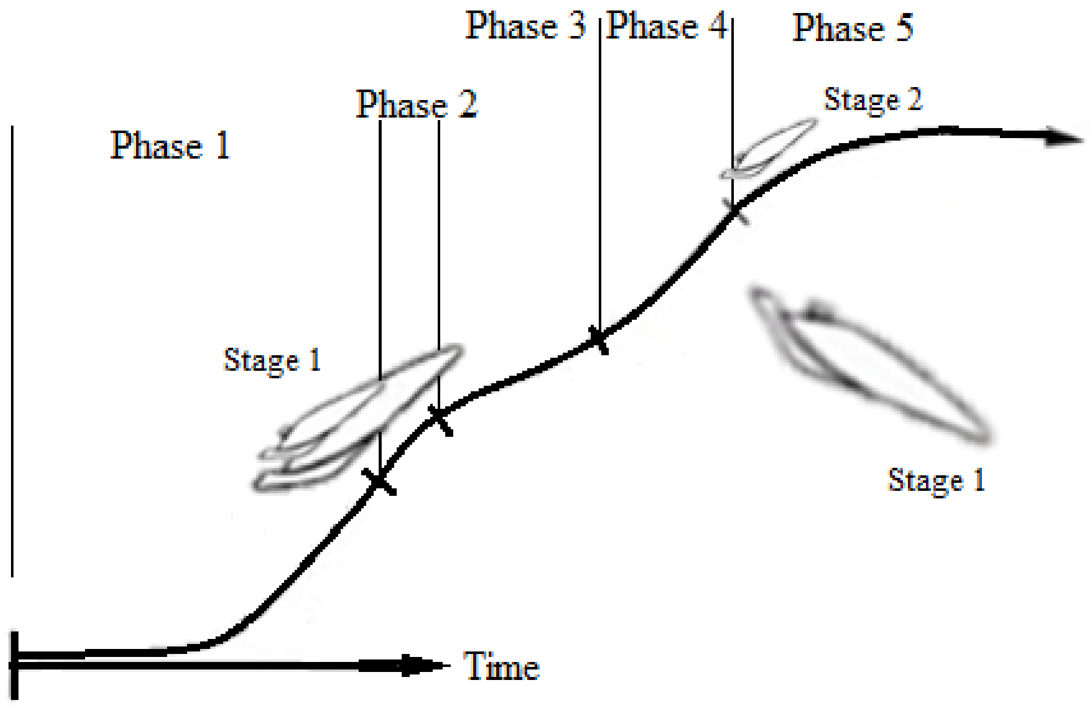

Figure 2. An example trajectory displaying the different phases and stages.

The phases divide different time periods in the trajectory when a particular engine choice can be running. Essentially they present a break-down of the trajectory where the user can decide what engines are allowed to operate in each phase. Such phases are defined by their variables and their constraints, both of which greatly affect the paths taken by the vehicle during the trajectory. The first of these variables is the length of time that they are allowed to last. These time spans are very important because they decide how long an engine is allowed to operate. Having multiple phases with varying time spans allows the code more flexibility in deciding how to properly use an engine type in the time stream. Thus if a large thrust engine is required only briefly, the phase can accommodate and only spend the required fuel to produce the large thrust for a brief period. Also, if a slower and longer ascent is more fuel efficient, that portion of the mission can be expanded to last longer.

The direction of the flight in each phase is controlled by the vehicles *attitude angles*. Attitude angles, such as the vehicles Angle of Attack (AOA) refer to the orientation of the vehicle relative to a given reference frame. More about the attitude angles will be explained in Section 4.3 of this thesis. For now it is important to know that the attitude angles at different points in the flight path are decision variables that allow the optimization routines in this thesis to manipulate the trajectory to find the best paths for the vehicle to fly according to.

The phases of the trajectory are also subject to constraints on flight properties such as altitude, dynamic pressure and Mach number. This must be the case since RLVs fly across numerous layers of the atmosphere which feature fluctuating gas properties. Such constraints on the phases limit what engines could be used in each phase of the flight, especially since air-breathing engines can only operate in limited Mach regimes. As a result, the constraints should have bounds large enough to accommodate different engines that can be varied for use in each phase.

Meanwhile, stages divide up physical parts of the vehicle. Different engines, fuel tanks and structural weights can be attached to different stages. Which stages operate in which phases can be user specified or optimized, as can be seen in Figure 3 in the next section. When a vehicle separates into a new stage at the beginning of a phase, this is termed as a '*staging*' event. At what point in the flight the staging occurs at can have drastic effects on the mission being performed, because at this point all the fuel, structural weight and engines that were used in a prior stage is released and thus, can no longer affect the payload. After staging, only the trajectory of the second stage of the vehicle is modeled in the phases that follow this event. This does not mean that the disconnected stage is always ignored. If the user wishes to analyze the fly back procedure, the trajectory of the previous stage can still be analyzed in separate phases. This study, however, is only concerned with the vehicle ascent and does not consider fly back procedures.

3.2.2 –Engine Configurations

In order to find out which engine configurations are best to power a given vehicle concept, an effective means of breaking them up into multiple variables was developed. The engine variables for each phase of the trajectory are broken up into engine 'Types,' engine 'Models,' engine 'Modes' and engine 'Controls.'

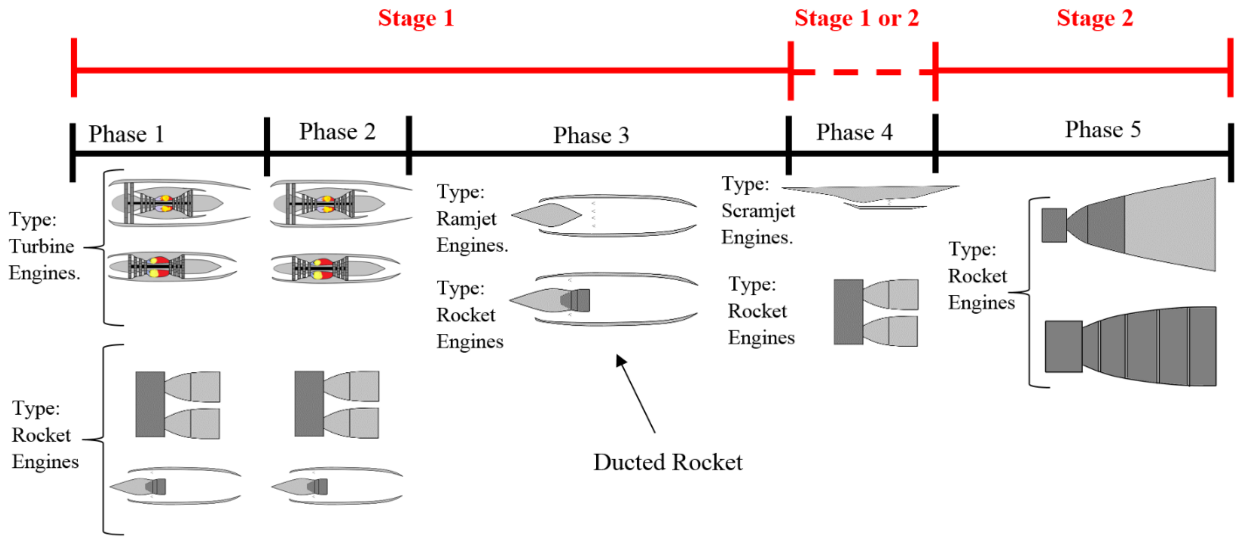

Figure 3. Chart describing the contents of each stage and phase.

Starting with the largest category, engine *Types* group together engines that have common propulsion methods and operate in similar Mach regimes. The four engine Types are ‘turbine engines’, ‘ramjet engines,’ ‘scramjet engines’ and ‘rocket engines.’ It is also important to note that each phase can only have one engine type running in it at a time.

The next largest category of engine variables is the engine *Model*. The user can have multiple engine Models to choose from to represent the engine Type. A Model contains all the thrust, mass flow and physical data of an individual engine. For example, one could have a Model of a turbojet, a Model of a turbofan with high bypass and a Model of another turbofan with low bypass. Each of these could compete against the other Models in their Type (see Figure 3), because each stage of the vehicle can have only one Model from any given Type. This is done because it seems unreasonable and costly to have multiple Models of the same Type aboard the same stage of the vehicle.

Each engine Model can have multiple *Modes* of operation. For example, a turbofan could be operating with the afterburner turned on in one phase or operating with afterburner turned off in another phase. Therefore, a Mode is a constant control setting for the engine during a phase. What Mode an engine is set to during a phase can be variable.

Engine *Controls* are dynamically changing variables that control the engine throughout a phase. The time history of the Controls is optimized in the trajectory optimizer to control the engines performance. Currently Steele-Flight only allows engines up to two controls. The first one is usually throttle.

The engine variables can alter the trajectory of the vehicle. Since what engine is chosen adds or subtracts weight from the Gross Takeoff Weight (GTOW), the amount of payload is greatly affected. The fuel efficiency of the engines is also of great concern because the weight of fuel required can in some cases take up half of the GTOW! More fuel weight can translate into less payload weight delivered to orbit. However, more thrust may be necessary regardless of fuel efficiency during a given phase of the trajectory. In this case, getting that thrust from additional engines would be undesirable because that would mean more weight to carry after that phase has ended, whereas additional fuel used due to a higher throttle setting would be spent and not affect later parts of the mission. Engine throttling can also have an effect on the trajectory when used in conjunction with modified attitude angles to exploit vehicle maneuvers that could slightly reduce fuel requirements. Considering such possibilities is tedious work and is where this code comes in handy.

3.2.3 –Figures of Merit

The figures of merit used in this study are payload delivered to orbit and the amount of propellant used by the chosen engine configurations. As mentioned previously, the amount of payload delivered to orbit is the true objective, but vehicles that sacrifice too much propellant for too little payload will not be considered optimal.

3.3 How does it work? Optimization steps.

How the code accomplishes its intended use is best explained through steps. A more detailed look at these steps is the subject of Chapter 4.

3.3.1 –Step 1 supply data

The user must first supply the code with data libraries for the engine models. The libraries themselves will have been created by the user on separate programs that will be described in Chapter 4. The libraries should supply information on engine performance such as thrust and mass flow rate and should also include the engine weight, what type of fuel the engine uses and the fuel density. This information will be important for making sure the volume of fuel and the volume of the engines do not exceed the allowable geometry of the vehicle.

Every vehicle model must be supplied with information for each stage including information on the weight, volumes of the fuel tanks, engine bay and cargo bay and an aerodynamic library for each stage. The vehicle parameters must remain fixed for the optimization.

3.3.2 –Step 2 Define mission and constraints

After inputting the required data to model the vehicle and potential engines it could use, the mission itself must be defined by specifying the desired target orbit for the vehicle to fly towards, by specifying how many phases the flight will be broken up into and by specifying what vehicle stages operate in each phase. The objective function can be defined by the user, but the current set up of Steele-Flight provides the user with the options of having the objective function be to maximize payload, minimize fuel for a fixed payload or maximize payload while constraining GTOW.

Because all vehicles have limits on dynamic pressure and external heating (and acceleration in the case of manned space launches), and because air breathing engines also impose limits on the flight regime, the user must properly define the constraints on the flight path. In the code the user has the option to impose constraints in each phase on different properties of the flight such as altitude, Mach number, velocity, dynamic pressure, acceleration etc. Further information on the constraints will be given in Chapter 4.

3.3.3 –Step 3 Define decision variables

The variables that the code will modify to optimize the vehicle and trajectory are chosen by the user depending on what he wants to vary and what he wants to keep fixed. These variables can be broken into two major categories: The *continuous* variables and the *integer* variables. The continuous variables are referred to as such because they can take on an infinite number of decimal values between the variable bounds. The continuous variables include the time history of the attitude angles and the engine controls, the length of time each phase lasts, the mass of fuel aboard each stage of the vehicle and the payload mass. Meanwhile the integer variables are referred to as such because they can only take on a user defined amount of values between the variable bounds. The integer variables are composed of every engine variable except the engine controls. That is, the integer variables are the engine types, models and modes used in each phase of the trajectory, and the number of each engine type on each stage of the vehicle.

3.3.4 –Step 4 Use GA optimization to find an initial guess trajectory

Genetic Algorithms (GAs) use principles of natural selection to vary variables to optimize objective functions. While this is not a very robust method to optimize a trajectory, GAs do have an advantage in that they don’t require initial guesses to optimize, they can optimize both continuous and integer variables and they can make use of multiple computer processors to speed up there performance (as is the case in this thesis). Meanwhile, NLP based methods such as the ones used in GPOPS-II are very accurate when finding an optimal trajectory given constraints and an objective function, however, these methods require an initial guess to start the optimization and since they depend on the derivatives of the variables and constraints, they cannot handle integer variables. Therefore, Steele-Flight begins with the use of a modified version of the GA built in MATLAB to find an initial guess for the best engine configuration and the best flight trajectory.

The GA works by running a flight simulation for each iterative guess at the engine variables. This is done by using a ‘Shooting Method’ that uses a Runge-Kutta algorithm to perform marching integration on the dynamic flight equations across the time span. While doing so, the engine libraries and the flight control histories are interpolated at each time step. The resulting trajectory yields information about constraint violations and how well the objective function was satisfied. This step will eventually yield an initial guess trajectory that can be input into GPOPS-II for further optimization. This step can also be seen in Figure 4 at the end of this section.

3.3.5 –Step 5 Hybrid optimization

After the GA has found an initial guess at the optimal trajectory and engine configuration, the time histories of all the flight properties taken from the initial guess are then used to start the next step in the optimization procedure.

The next step uses both GA and the NLP solver. This time, the GA is only allowed to alter the integer variables since the NLP solver cannot do this. Also, instead of the GA running a flight simulation using the Shooting Method for each guess at the variables, it runs the NLP solver to optimize the trajectory for every engine combination guessed by the GA by using the initial guess from the previous step.

The methods for using an NLP solver to solve a trajectory optimization problem are more complex than that used by the GA and are explained more in depth in Chapter 4.

The Hybrid method cycles through many possible engine configurations and picks either the best or a group of the best configurations. However, in order to allow the Hybrid method to finish in a timely manner, the NLP solver within GPOPS-II has constraints set that constrict it from running too long. Thus, while the trajectories that it outputs are very close to the optimal trajectory for a given configuration, they are not refined to the best of the GPOPS-IIs abilities. However, this does not hinder the selection of the best engine configurations but it does allow the Hybrid method to discern which engine combinations are not feasible in a quicker fashion. This is discussed more in depth in Section 4.5 of this thesis.

3.3.6 –Step 6 Refined trajectory optimization using GPOPS II

After the hybrid method outputs a group of the best engine configurations, each engine configuration can be run through GPOPS-II set to its full optimization potential, yielding a final look at the best possibilities for engine selection. A chart describing the steps taken in Steele-Flight is shown below in Figure 4.

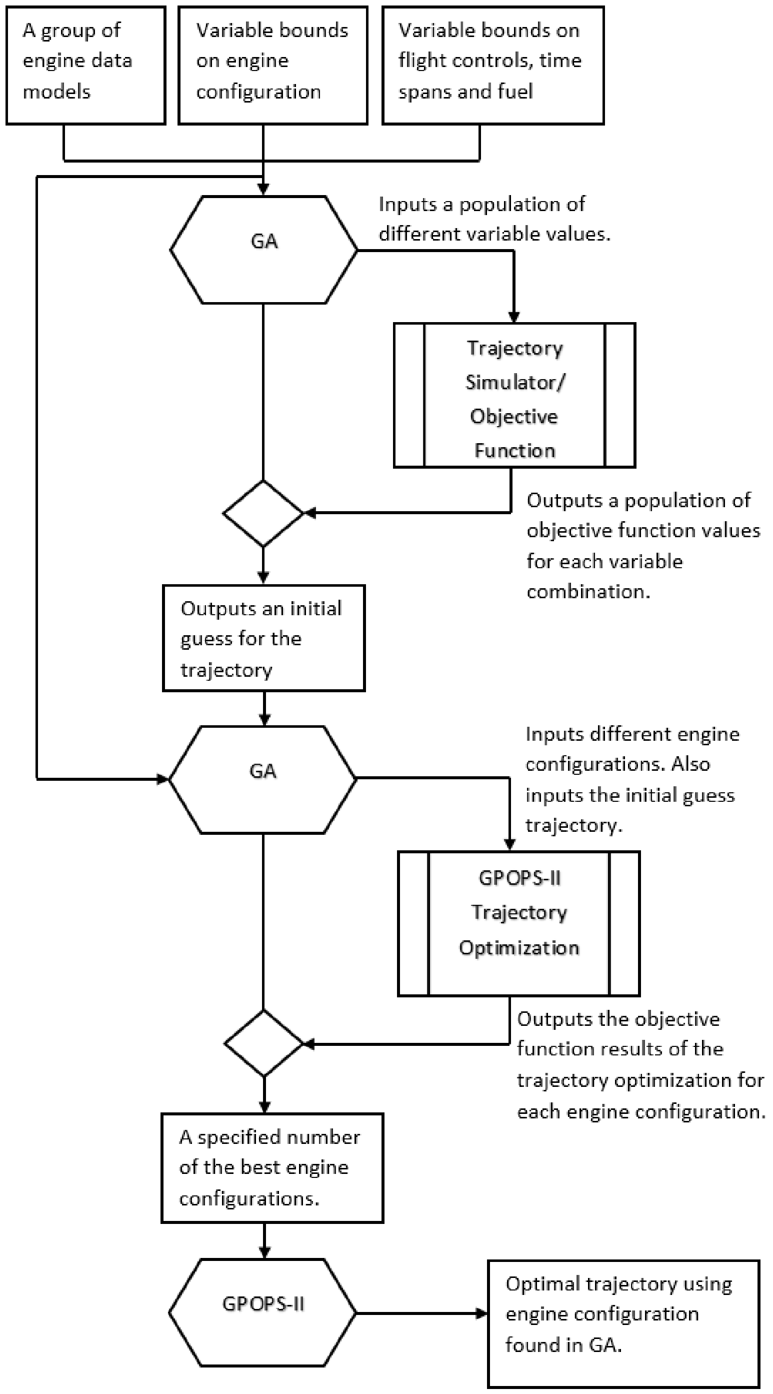
A group of engine data models
Variable bounds on engine configuration
Variable bounds on flight controls, time spans and fuel
GA
Inputs a population of different variable values.
Trajectory Simulator/ Objective Function
Outputs a population of objective function values for each variable combination.
Outputs an initial guess for the trajectory
GA
Inputs different engine configurations. Also inputs the initial guess trajectory.
GPOPS-II Trajectory Optimization
Outputs the objective function results of the trajectory optimization for each engine configuration.
A specified number of the best engine configurations.
GPOPS-II
Optimal trajectory using engine configuration found in GA.

Figure 4. Flow Chart displaying the steps taken in Steele-Flight.

Chapter 4- Details of Modeling Method

4.1 Vehicle Models and Aerodynamics.

Steele-Flight does not currently have any means of optimizing vehicle size or structural design. Thus the user must provide the vehicle data including an aerodynamic library, structural weight and volume data on the engine bays and fuel tanks, all for each stage of the vehicle.

4.1.1 –Saenger (Sänger): The TSTO RLV Reference Concept

The vehicle data for all the experiments performed in this thesis come from the data for the German designed Saenger Space plane since data for this concept is readily available[3, 6, 10, 16].

The Saenger was developed to be a TSTO RLV capable of delivery of both manned and unmanned orbiter stages to LEO using a hydrogen fueled TBCC configuration (without a scramjet mode) for the bottom stage booster and a LOX-LH2 fueled rocket for the upper stage orbiter (either the unmanned CARGUS orbiter or the manned HORUS orbiter) The published data for this vehicles structure provided a reference point for the studies performed in the next chapter.

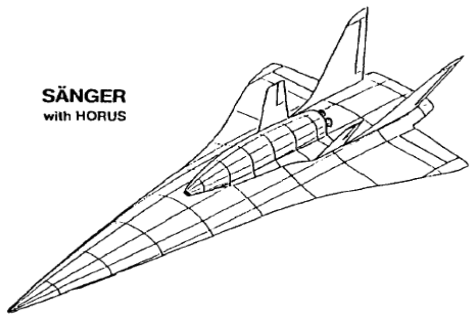

Figure 5. Image of the Saenger launch vehicle with the HORUS manned second stage. Sachs, G. and J. Drexler, *Optimal ascent of a Horus/Saenger type space vehicle*, in *Astrodynamics Conference*. 1988, American Institute of Aeronautics and Astronautics [4]. Used with permission from Heather Brennan, American Institute of Aeronautics and Astronautics, Inc., 2015.

4.1.2 –Aerodynamic Library

The aerodynamic libraries are matrices of values for lift and drag coefficients. Depending on the vehicle model, they can be functions of Mach number and the attitude angles (AOA, Sideslip, etc.). For this level of analysis, the effects of rotational moments about the vehicle center of gravity are ignored. An aerodynamic library should be supplied for each stage of a vehicle in order for the vehicle aerodynamics to be incorporated into the flight simulation.

Currently, Steele-Flight cannot model effects of propulsion systems on the vehicle aerodynamics as integration of engine inlets and vehicle aerodynamics are beyond the scope of this conceptual level analysis.

4.1.3 –Volume. What can fit?

The user supplied data for the engine bay in each stage of the vehicle provides a means for Steele-Flight to determine what will fit and what will not. Since the volume of the Saenger engine bay is never explicitly stated, the data for these dimensions was “eyeballed” from published images such as those seen in Figure 17 in Chapter 5.

If the base mass of fuel is known along with its density, the volume of the fuel tanks on board the vehicle can be determined using the known relation:

$$Volume = \frac{mass}{density} \qquad \text{(Eq. 1)}$$

The engines themselves will have a fuel specified for them along with the fuels density and since fuel mass is decision variable, Steele-Flight will have to make sure the allowable volume specified by the vehicle model is not violated when a certain fuel mass is chosen.

The weight of the fuel tanks currently must be integrated into the vehicle structural weight. Therefore, for the time being, all possible fuels that could be selected for a stage of the vehicle should require fuel tanks of roughly the same weight.

4.2 Engine Models

As stated before, the engine models used in this code must be created independently by the user in another program. Once the model is created, all necessary information for the model must be stored in MATLAB data structures. These structures should contain the weight and volume of the engines, what type of engine they are and multidimensional matrices for the thrust and mass flow rate as functions of flight properties, attitude angles and engine controls. These matrices are for interpolation during trajectory simulation.

The effects of the engine inlet and nozzle integration into the vehicle structure should be accounted for in the engine models and libraries. It is up to the user to do this.

For the engine libraries that were used in the computational experiments performed in this thesis, modeling programs from literature and industry were used to create the engine models. These models and the programs from which they were built are described in the following subsections.

4.2.1 –Turbine Engines and GasTurb 11

Turbine engines are used for flight speeds between 0 and Mach 3.5, although some studies have attempted to design engines to go up to Mach 4. Unlike ram/scramjet engines, turbine engines can operate in subsonic flight and have a far superior Isp in comparison to a rocket. However, due to the turbomachinery housed within, turbine engines exhibit very variable performance across both the altitude and Mach regimes and are quite heavy.

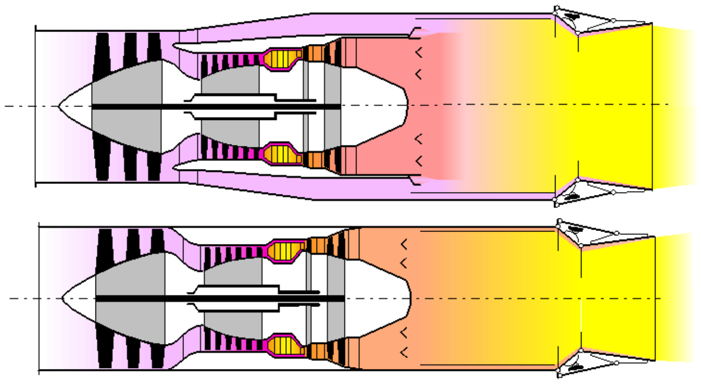
Figure 6. GasTurb models for a Two Spool Mixed Flow Turbofan (top) and a Two Spool Turbojet (bottom). Kurzke, J., GasTurb11. 2007 [1]. Used with permission from Dr. Joachim Kurzke, 2015.

The turbine engine models used in this thesis were created using a code called GasTurb 11 [1]. GasTurb 11 is a program for modeling gas turbine engines created by Dr. Joachim Kurzke. It uses a graphical user interphase (GUI) that allows the user to select different engine models. Each engine model is distinguished from one another by whether the engine is a turbojet, turbofan, turboprop or turboshaft. These categories are also divided by details such as the number of spools, how the bypass duct is used, if any are present, and other physical details as can be seen in Figure 6.

The code is provided with empirically-determined maps for the compressor and turbines that were taken from open literature. These maps are scaled to approximate the performance of the turbomachine according to the physical engine being designed.

To give a more accurate model of the effects of combustion, the code makes use of the code 'Chemical Equilibrium with Applications' (CEA) to get good estimates of the gas flow behavior. More information on this code can be found in [18]. The fuel used is user selected from a list of options.

GasTurb 11, uses basic principles of thermodynamics and compressible flow applied to turbomachinery and propulsion to model the performance of different systems within the engine. All the engine models have default set ups; thus the user has only to use the GUI to modify design variables and then run a simulation. Besides performance calculations, it also allows the user to get estimates as to the size and weight of the engines being designed based on:

1. The geometric information from the performance design such as flow area through the different parts of the engines.
2. Variable materials for the different parts of the engine (including the casing and ducts).
3. Variable dimension ratios for engine parts, such as casing thickness and inlet cone size among others. By having these parameters as ratios connected to flow areas of the engine, the entire engine geometry is scaled according performance design.

The variable materials and dimension ratios are already set to a default setting set by the codes creator and in this thesis they are left in default setting. More information in this design methodology can be found in Chapter 5 of [19].

The process of using this code to create a design involves two steps. Upon selecting a model of engine, the user must create the design point for the engine, in which the designer selects a design pressure ratio for the compressor. In this mode, the engine geometry is generated and fixed. Afterwards, in a separate mode, the user can run off-design simulations of the engine, in which the geometry remains fixed. In this mode, the user can create a library of thrust and mass flow rate data as functions of altitude, Mach number and spool speed. Also, off-design mode allows the user to define a map of inlet losses for the engine at different points in the Mach regime. This map can give some limited ability to simulate engine airframe integration.

4.2.2 –DMSJ, TBCC and RBCC Engines and SCCREAM

Ramjets and scramjets are essentially ducts that are shaped to capture and diffuse incoming supersonic air, combust fuel with the air and expand the combustion products back to supersonic speeds out a nozzle. In the case of a ramjet, this involves diffusing the incoming supersonic air down to subsonic speeds and high static pressure for combustion, after which the products are expanded back to supersonic speeds using a converging-diverging nozzle. This system, which relies on the dynamic pressure of incoming air and is very inefficient below Mach 2.5. On the other hand, the ramjet cannot go above Mach numbers around Mach 6 because the inlet diffusion system generates extremely high losses due to the shocks involved in slowing down such high speed air and also greatly raises the temperature of the ingested air entering the combustion chamber leading to severe loss due to dissociation. To mitigate this effect, scramjet engines diffuse the incoming air down to low supersonic speeds and combust the air while still at supersonic speeds and expanding the products using a purely diverging nozzle. As a result, the shock losses and temperature increase are greatly lowered. Thus, the scramjet engines are most effective from Mach 5 up to the accepted theoretical limit (Mach 12 in this thesis).

In most conceptual designs these two engines are combined into what is called a Dual Mode Scramjet (DMSJ) set up where the ramjet and scramjet are simply different operating modes of the same engine flow path. Also, since each of these engine modes rely entirely on the inlet geometry to compress the incoming air over a wider Mach regime than a turbine engine, they both require variable geometry inlets in order to keep their performance as consistent as possible over the Mach regime. Changing internal geometry can be quite significant when the two engines, which operate at different internal Mach regimes are combined into one flow path. In a scramjet the air being combusted is supersonic and thus uses a purely diverging nozzle to expand the supersonic air flow, while ramjets use converging diverging nozzles. Thus there must exist a significant number of actuators to power the altering internal geometry that divides the two modes. The addition of such actuators and variable geometry can add a significant amount of weight to the vehicle design.

It can be noted that scramjet engines require very long inlets and nozzles to properly compress and expand the captured air. Because of this, the underside of the hypersonic vehicle is usually used as an external diffuser providing shock ramps to compress the air before entering the internal inlet geometry. The aft side of the vehicle underbody serves as a nozzle to expand the products of combustion. Because of these requirements, the design of a scramjet engine cannot be decoupled from the design of the vehicle on which it is installed.

Both modes of the DMSJ engine require RAM compression of the supersonic air and thus they are both incapable of generating static thrust and are very inefficient in the subsonic Mach regime (in which the scramjet cannot operate at all). Besides this, the scramjet is still an air-breathing engine and cannot operate outside of the atmosphere. Therefore, no matter what the application, another propulsion system must be used in combination with these engines to generate static thrust and fly through the subsonic to low supersonic Mach regime and then again in case of orbital requirements. To accommodate these needs, there are two types of Combined Cycle engines that can be considered, known as Turbine Based Combined Cycle (TBCC) engines and Rocket Based Combined Cycle (RBCC) engines.

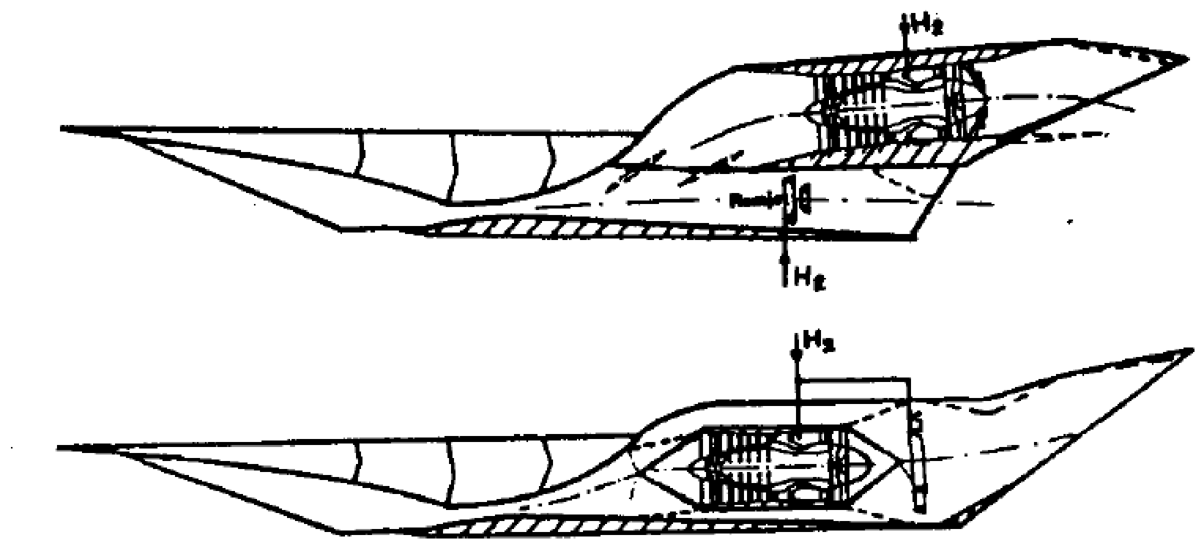

Figure 7. Parallel TBCC configuration (Top) and Coaxial TBCC configuration (Bottom). The image is adapted from Dietrich, K., *Advanced two-stage launch vehicle concepts (SANGER)*, in *26th Joint Propulsion Conference.* 1990, American Institute of Aeronautics and Astronautics [3]; reprinted by permission of Heather Brennan, American Institute of Aeronautics and Astronautics, Inc., 2015.

TBCC engines come in two usual configurations as seen in Figure 7. The parallel flow path configuration, has the two engines situated parallel two one another, sharing only a common inlet and nozzle. The coaxial arrangement has the Ram/Scram duct bypass around the turbine engine core to the afterburner which functions as a DMSJ burner when in one of the DMSJ modes. However, this configuration requires both engines to be designed together, increasing the complexity of the individual engine designs. Also, since TBCC engines are purely air-breathing they can only be used on the first stage of a TSTO RLV.

An RBCC engine combines a DMSJ engine with an additional rocket engine that can be used once the limit of air-breathing operation has been reached. Thus the rocket nozzle and the DMSJ are combined into a single flow path for weight savings. Besides this, the rocket can also be used to generate static thrust at the beginning of flight in an additional mode known as Ducted Rocket mode. In the case of a ducted rocket, incoming air is ingested and compressed by the engine inlet and is then entrained by the flow of rocket propellant, thus augmenting the thrust by providing additional pressurized propellant flow through the nozzle. While RBCC engines appear to be very attractive in terms of weight savings, they still require additional stored oxidizer to fuel the rockets, which results in very low Isp during the subsonic ascent regardless of air-augmentation. Because the rocket uses the same flow path as the DMSJ, this system is not optimized for Rocket operation and will therefore produce lower Isp for pure Rocket operation.

The ramjets, scramjets and ducted rockets used in this thesis were modeled using the code known as SCCREAM (Simulated Combined-Cycle Rocket Engine Analysis Module) which was originally developed by Dr. Bradford and Dr. Olds at Georgia Tech [7, 20-22]. For use in this thesis, the code was provided as an executable by the author.

SCCREAM is a modeling tool meant for the conceptual level design of RBCC engines and DMSJ engines [20]. It models these engines by using quasi-one-dimensional flow analysis and principles of compressible flow and thermodynamics to solve the conservation equations at each station shown in Figure 8 below. With the exception of the external compression system, almost all the geometry that can be specified in SCCREAM are area ratios for each station relative to the inlet area. The program does make some simplifying assumptions using component efficiencies and a Mil-Spec inlet pressure recovery schedule for subsonic combustion analysis [7, 20]. For supersonic combustions analysis, the pressure recovery was determined by modeling the oblique shockwaves in the external and internal inlet.

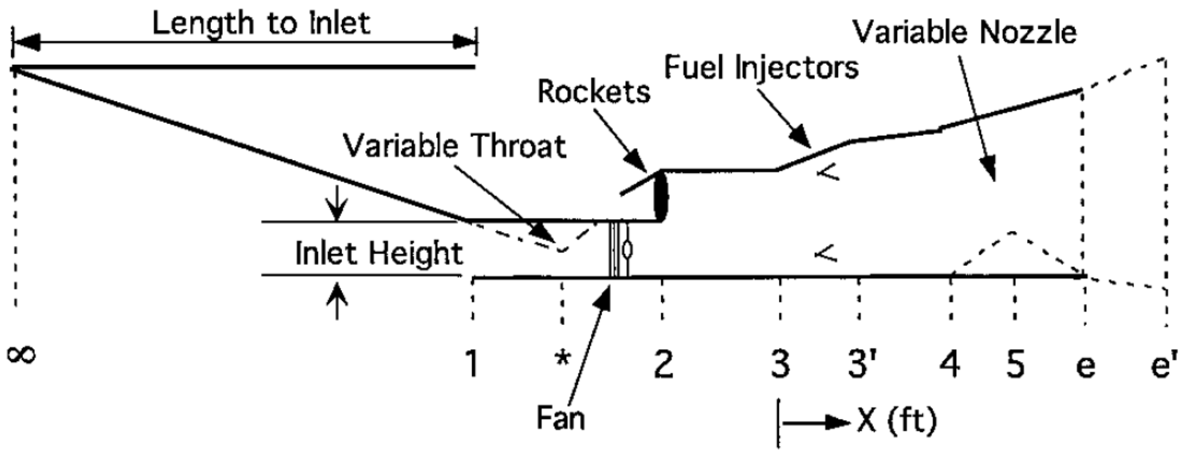

Figure 8. SCCREAM RBCC diagram. Since SCCREAM can also model fan augmented Ramjets, there is a fan option that can be seen in this figure, even though it is not used in this thesis. Olds, J.R. and J.E. Bradford, *SCCREAM: A Conceptual Rocket-Based Combined-Cycle Engine Performance Analysis Tool*. Journal of Propulsion and Power, 2001. 17(2): p. 333-339 [7]. Used with permission from Ashley Russ, SpaceWorks Enterprises Inc., 2015.

The combustion process was modeled 1-Dimensionally using a marching integration technique to solve the dynamic Mach number equation for the Fanno-Rayleigh flow while taking into account the effects of area change and mass addition.

The nozzle flow composition can be found using calculations for both frozen or equilibrium flow after the combustor. Whether the flow maintains an equilibrium composition throughout the nozzle, has a frozen composition throughout the nozzle or a bit of both is up to the user to define using a ratio of frozen to equilibrium flow. Essentially it means that a percentage of the total thrust of the nozzle is determined from frozen composition calculations and is summed together with a percentage of thrust determined from equilibrium composition calculations [20].

SCCREAM does not include any method of determining the weight of the engine.

4.2.3 –REDTOP-Lite

All non-air-breathing rockets used in the experiments in this thesis were modeled using the code known as REDTOP-Lite (Rocket Engine Design Tool for Optimal Performance –Lite) [23] which was created and distributed by Spaceworks© Enterprises Inc.

While REDTOP-Lite is meant for conceptual-level design, it also includes a number of premade engine models that are based on existing engines. In this thesis, only these premade models of existing engines were used and the known weights and dimensions of these engines were also used. In all of these models, the nozzle is assumed to have 100% equilibrium flow.

4.3 Trajectory Optimization

Trajectory optimization takes up the majority of the run time of this code and is very important to the overall optimization method being employed. In order to determine whether a given engine configuration is the best choice, one must know the best trajectory that is unique to that engine configuration in order to properly compare to other engine configurations. Therefore this section will summarize the mathematics and theory for the trajectory optimization methods used in this thesis.

4.3.1 –The Trajectory Simulation

A trajectory can be modeled using a system of Ordinary Differential Equations (ODEs). The ODEs are the derivative functions of the flight properties defining the trajectory which in this thesis will be referred to as the *States*. Different sets of States can be used to define a trajectory but the States used to define a trajectory in this thesis, with their corresponding derivative functions, are:

Longitude (λ): $\dot{\lambda} = v\cos\gamma\frac{\sin\chi}{R\cos\delta}$ (Eq. 2)

Declination (δ): $\dot{\delta} = \frac{v}{R}\cos\gamma\cos\chi$ (Eq. 3)

Radius from the Center of the Earth (R): $\dot{R} = v\sin\gamma$ (Eq. 4)

Velocity relative to planet surface (v):

$\dot{v} = \boldsymbol{X} + \Omega_E^2 R\cos\delta\,(\sin\gamma\cos\delta - \cos\gamma\cos\chi\sin\delta)$ (Eq. 5)

Flight-Path-Angle (γ):

$\dot{\gamma} = \frac{v}{R}\cos\gamma - \frac{\boldsymbol{Z}}{v} + 2\Omega_E\sin\chi\cos\delta + \frac{\Omega_E^2}{v}R\cos\delta\,(\cos\gamma\cos\delta + \sin\gamma\cos\chi\sin\delta)$ (Eq. 6)

Velocity Azimuth (χ):

$$\dot{\chi} = \frac{v}{R}\sin\chi\tan\delta\cos\gamma + \frac{Y}{v\cos\gamma} + 2\Omega_E^2(\sin\delta - \tan\gamma\cos\chi\cos\delta) + \frac{\Omega_E^2}{v\cos\gamma}R\sin\chi\sin\delta\cos\delta \quad \text{(Eq.7)}$$

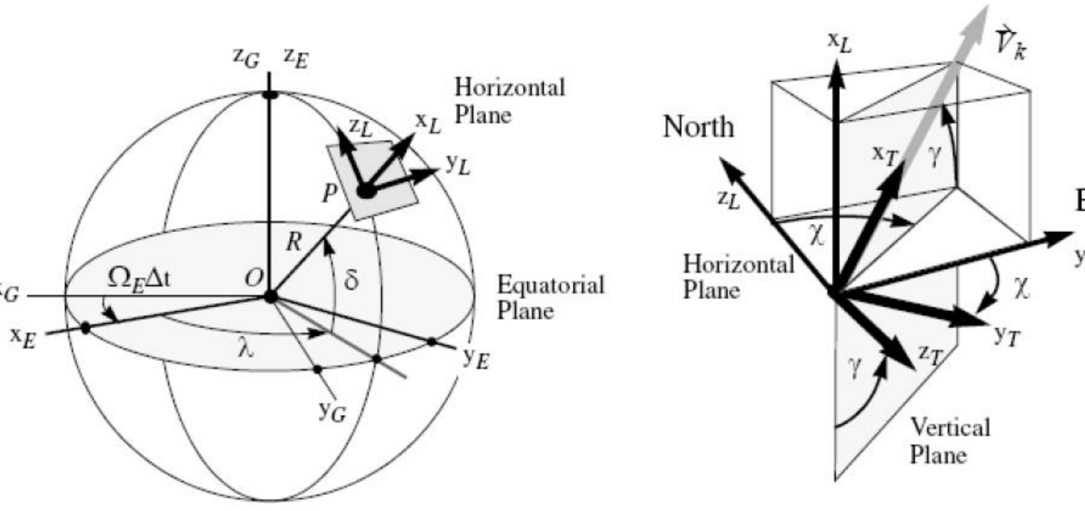

Figure 9. "Definition of the planet-relative position and flight-path state variables. The inertial planetocentric axis system (index G) and the rotating planet-fixed axis system (index E) span the equatorial plane, their z axes coinciding with the planets axis of rotation. The trajectory axis system (index T) is attached to the flight-path velocity vector $\vec{V}_k$" [2]. The Local axis (index L) is always tangent to the surface of the planet with the Z axis pointing toward the North Pole and the Y axis pointing East. Fischer, D., P. Gath, and K. Well, *ASTOS Model Library Version 7.2.0*. 2006-2014, pg 481 sec 13.3.3. Used with permission from Francesco Cremaschi, ASTOS Solutions GmbH, 2015.

In the equations just described the symbol Ω_E represents Earth's rotational speed. The **X**, **Y** and **Z** forces represent the total forces per unit mass in the X, Y and Z coordinates of the Trajectory axis system as seen in the figure above. They are the sum of the Thrust, Weight, Lift and Drag forces applied to the plane divided by the instantaneous mass of the vehicle and are functions of the thrust libraries from the engines, the Lift and Drag coefficient libraries from the aerodynamic model, the gravity effects on the mass of the vehicle and the atmospheric data libraries. A sixth State can be added for the mass flow rate of propellant from the Space plane, which is dictated by the engine libraries.

It can be noticed in Eq.2-7 that the ODEs are functions of the states: *λ(t); δ(t); R(t); v(t); γ(t)*; *χ(t)* and *m(t),* which are all continuous functions of time. In order to simulate a trajectory, and thus find the complete time history of the States, the initial value of each state at the start time must be provided and then the ODEs must be integrated numerically over a span of time. The values of force per unit mass in the X, Y and Z directions will be found via interpolation from the data libraries provided for the engines and aerodynamics at each time step during the integration. The libraries themselves are functions of the states, attitude angles and engine controls.

This set of ODEs indicate that the trajectories being modeled will have '3 degrees of freedom' (3DOF). That is, they model the movement of an aircraft in 3-directions (up-down, backward-forward, side-to side). This dynamics model does have some drawbacks, because it cannot model the dynamic changes in the vehicle's rotation about its axes, as this would require a 6DOF model.

4.3.2 –Trajectory Controls

The flight of the vehicle through the atmosphere can be controlled by aerodynamic surfaces that increase drag or lift on certain parts of the vehicle in order rotate the vehicles forward axis relative to the direction of velocity vector and thus modify its attitude angles. Also, some vehicles, such as rocket boosters, use a *thrust vectoring* system to angle the direction of the engines output flow in order to guide the vehicle relative to the planet or its own axis.

In this thesis, details such as rotational moments resulting from control surfaces and thrust vector control cannot be modeled without a 6DOF dynamic model. Therefore, the vehicles attitude angles are controlled directly by optimizable input.

4.3.3 –Orbits

Five Keplerian orbital elements can be used to describe an elliptical orbit. These elements are the 'semi-major axis' (a), the 'eccentricity' (e), the 'right ascension of the ascending node' (Ω), the orbit 'inclination' (i) and the 'argument of the periapsis' (ω). An additional parameter, the 'true anomaly' (θ), is used to describe the position of a satellite in that orbit. The role of these elements and the true anomaly can be seen below in Figures 10 and 11.

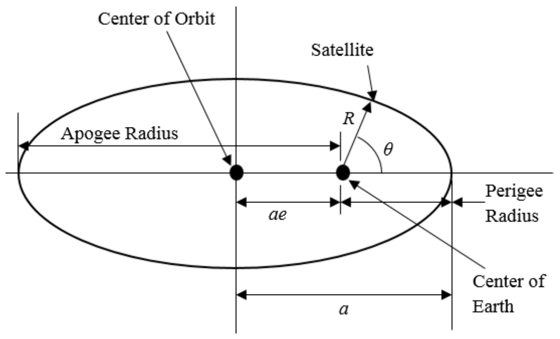

Figure 10. View of orbit about center of Earth. The Apogee Radius is the point in the orbit that is furthest from the center of the planet. The Perigee Radius is the point that is closest to the center of the planet.

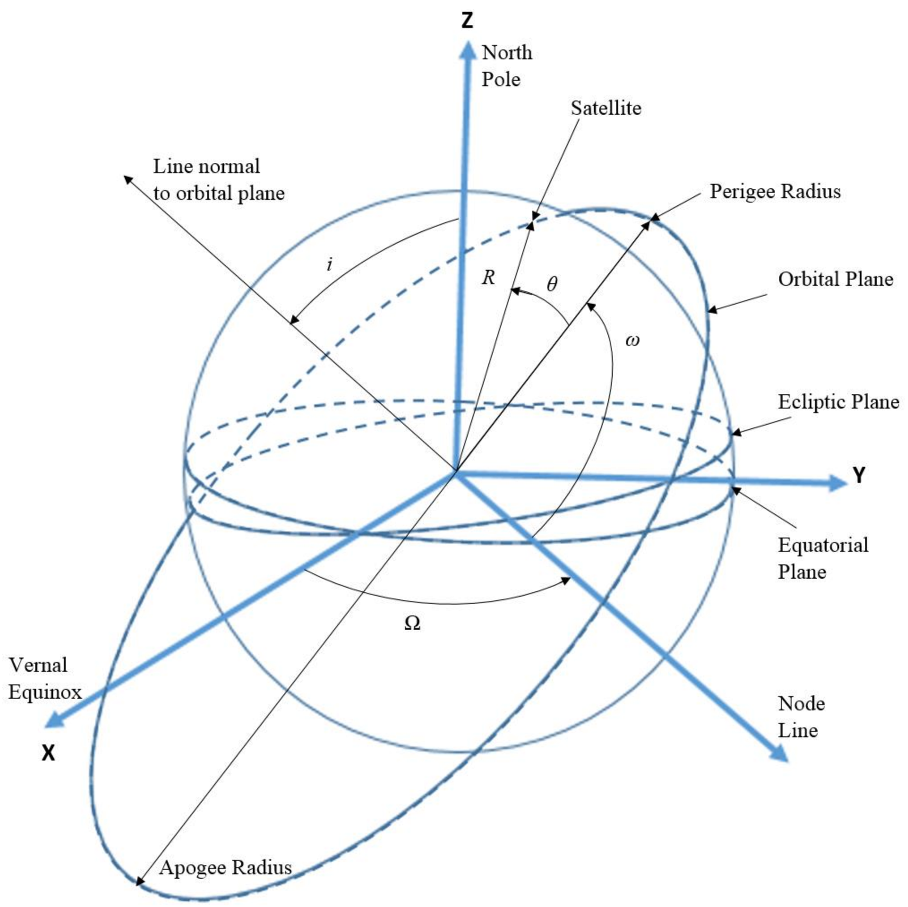

Figure 11. View of an elliptical orbit around the Earth.

In this thesis, the term ‘the right ascension of the ascending node’ is ignored and thus the orbital plane is independent of the Vernal Equinox and longitude. The other four orbital elements and the true anomaly can be determined from the State values at the final point in the trajectory simulation.

State values at the final point of a trajectory simulated in Steele-Flight can be constrained to match a specific orbit plane with a user specified apogee and perigee radii and inclination with respect to the equatorial plane (these are the only parameters that need constraining since **Ω** and longitude are ignored in this thesis). For further discussion on orbit mechanics, the five orbital elements, the true anomaly and their relationship with the State parameters chosen in this thesis, the reader is directed to [24].

4.3.4 –The Function Generator

Optimization programs minimize or maximize an objective function by variation of its variables. These equations are often subject to constraint equations that constrain the objective function to output values within a certain range only using variable values within the variable bound constraints as seen in this example:

Minimize objective function: $F(\boldsymbol{x})$ (Eq. 8)

Subject to the constraints: $\boldsymbol{c}(\boldsymbol{x}) \geq 0$ (Eq. 9)

$\boldsymbol{x}_{lb} \leq \boldsymbol{x} \leq \boldsymbol{x}_{ub}$ (Eq. 10)

The example above is a basic *Nonlinear Programming Problem* or NLP in which ***c*** is a vector of nonlinear constraints functions and *F* is the scalar objective function. Both the constraints and the objective function are functions of the input variable vector ***x***.

Meanwhile, the system of ODEs used to model the trajectory, as seen in Eq. 2-7, are continuous functions of the States, which are unknown continuous functions of time. Optimization problems involving the manipulation of such functions are called *Optimal Control Problems* and can be solved using the calculus of variations [25]. However, this method is extremely tedious and it is often more convenient to *transcribe* the optimal control problem into an NLP problem with a finite set of variables and constraints such as the one seen above. In this form, the objective function can be optimized with a GA or an NLP solver. Transcribing the problem is approached differently for the two optimizers used in Steele-Flight, however the objective is the same. The objective is to generate a function with a finite amount of variables and constraints from the continuous problem described by the system of ODEs so that a finite variable optimization method can be used to alter the variables in order to satisfy the constraints and minimize the objective function. Thus, the next two sections will cover two different ways to build a *Function Generator* [26].

4.3.5 –GA Transcription: The Shooting Method

In order to transcribe the optimal control problem, the trajectory is treated as a Boundary Value Problem (BVP) with fixed values for the States at the start time of the trajectory (t_o) and is constrained by boundary conditions and other constraints on the flight path throughout the trajectory. In order to solve the BVP, one must integrate the ODE system across the time span of the trajectory to find the time histories of the states and see if they satisfy the constraints imposed on the states at different points in the trajectory. In order to ensure the integrated solution of the system of ODEs satisfies all constraints the optimizer needs to be able to manipulate variables that control the direction of the trajectory. The variables for the form of

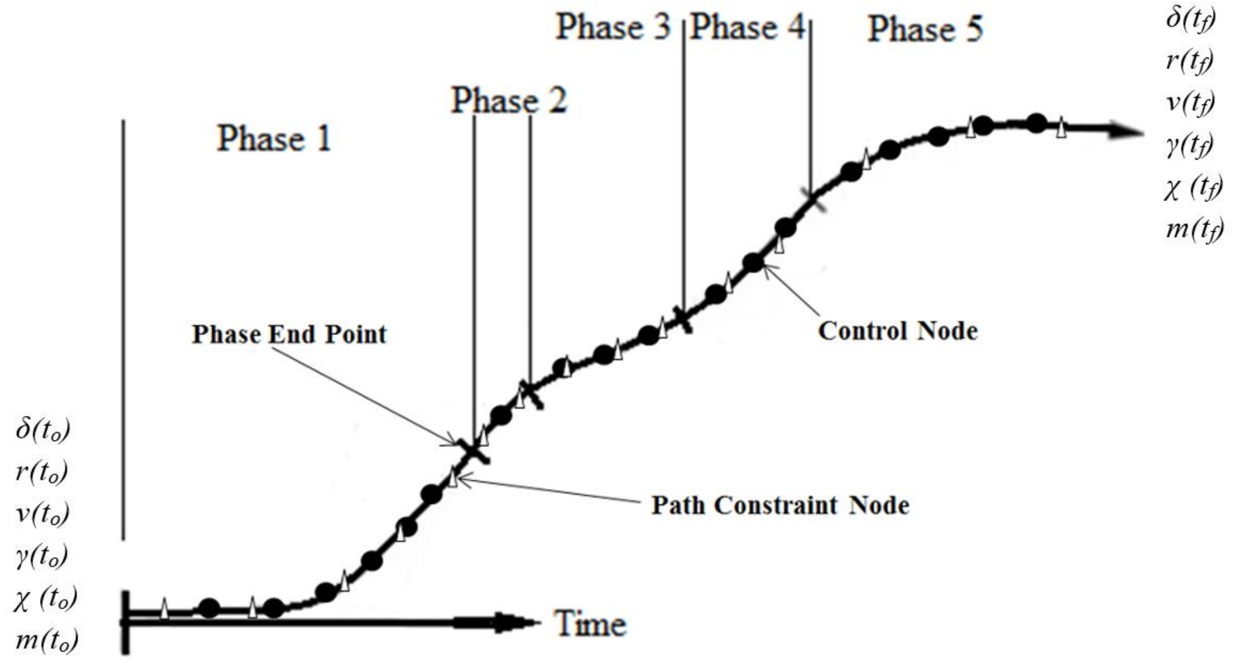

Figure 12. An example trajectory displaying the phases, control nodes, path constraint nodes and phase end points. In this case, the mass of the vehicle is treated as one of the states.

transcription used for the GA, are the values of the attitude angles at the “control nodes,” and the time span of each phase as seen in Figure 12.

The control nodes are points in the time span where the user can specify values for the attitude angles. The actual number of control nodes in each phase is user defined and they are distributed evenly within the phase in which they are defined. Thus their position in the time stream depends on the time spans of each phase which are also variable. In addition to the control nodes, there are also “path constraint nodes” where the user can specify constraints on the flight properties. Like the control nodes, their position in the time stream depends on the time spans of each phase.

The technique used to solve the BVP is called the *Shooting Method* which is performed in the following four steps [26]:

1. Guess values for the variables (that is, the attitude angles at the control nodes along the trajectory and the values for the time span of each phase).
2. Propagate the differential equations from t_o to t_f using an explicit integration method such as the Runge-Kutta method. During the integration method all data libraries plugged into the dynamic system will be interpolate at each time step. Also the values of the controls are found by interpolating their guessed time histories at the node points for any given time step in a phase. Since the trajectory is broken up into different phases, each phase will have its own boundary conditions which will be called here “end point constraints.” At the end point of each phase the user has the option to specify constraints for each of the states, as well as on Mach number and Acceleration. The next phase will be propagated from the end conditions of the previous state. Additionally, since different types of engines can be used in different phases of the trajectory, the change in engine operation is assumed instantaneous across the phase bounds. The same also applies to vehicle separation, where the vehicle mass and aerodynamics change instantaneously across the phase bounds. The marching integration should yield a complete time history of all the states (that is, it outputs a complete trajectory).
3. Evaluate the error in the boundary constraints.
4. Use an optimization program to alter the input variables and rerun the first three steps until the constraints are satisfied and the objective function is minimized.

Thus these steps input variables into a function generator which outputs the constraint error and the value of the figure of merit. A GA optimizer is used to drive the Shooting Method until the boundary conditions and constraints are satisfied and any objective function is minimized.

As a final note, Steele-Flight GA does not currently use the engine controls as variables because having another flight control would result in a very large number of variables which would adversely affect the GAs abilities. This will be discussed further in section 4.4. Also Steele-Flight currently only allows two attitude angle controls, the AOA and Pitch. Thus Steele-Flight cannot optimize trajectories with side-to-side motion.

4.3.6 –NLP Transcription: LGR quadrature orthogonal collocation

For the NLP based optimization method employed in GPOPS-II, a *collocation method* is used to transcribe the problem. According to the abstract of reference [17] “The software employs a Legendre-Gauss-Radau quadrature orthogonal collocation method where the continuous-time optimal control problem is transcribed to a large sparse nonlinear programming problem (NLP).” The method will be referred to as Radau collocation for the remainder of the thesis. The resulting NLP is then solved with an NLP solver to get an optimized answer.

Unlike with Shooting method, which only uses the time spans and the controls at each control node as variables to solve the BVP, a collocation method breaks up the time span of each phase of the trajectory into a grid of points and treats the values of the States and controls at those grid points as decision variables along with the time spans of each phase.

Since the decision variables include the time history of the States, marching integration of the system of ODEs is not necessary. Instead, this method uses a collocation method employing a numerical quadrature method to ensure the accuracy of the guessed time history of the States and controls at every grid point. In order to do this, each State and control across each phase of the trajectory is approximated by a piecewise Lagrange polynomial that is created from the input State and control variables at the grid points. The derivatives of the Lagrange polynomials at each grid point are then found and compared with the analytical derivatives (the left hand side of the ODEs in section 4.3.1) at the same grid point (also known as collocation points), after which the error is recorded. With this method, termed 'collocating', the accuracy of the polynomial approximation of each State and control is assessed. Detailed information on the methodology and significance of how the grid points are arranged into meshes and the positioning of the collocation points at the LGR points is not covered in this thesis but the interested reader is referred to [17].

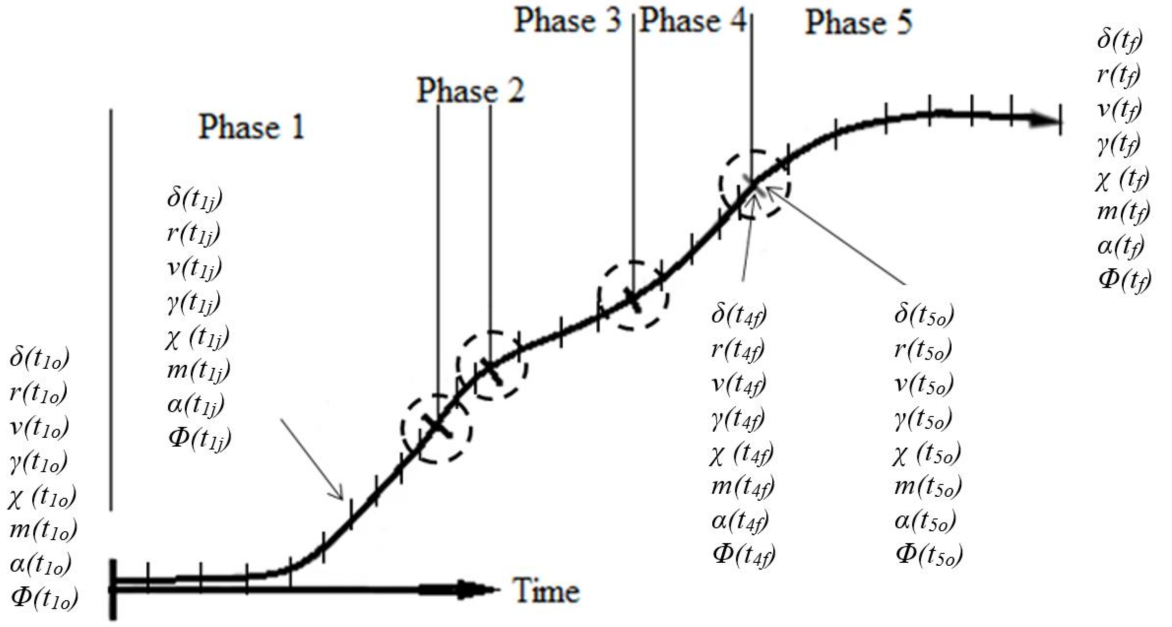
Phase 1
Phase 2
Phase 3
Phase 4
Phase 5
Time
δ(t1o) r(t1o) v(t1o) γ(t1o) χ (t1o) m(t1o) α(t1o) Φ(t1o)
δ(t1j) r(t1j) v(t1j) γ(t1j) χ (t1j) m(t1j) α(t1j) Φ(t1j)
δ(t4f) r(t4f) v(t4f) γ(t4f) χ (t4f) m(t4f) α(t4f) Φ(t4f)
δ(t5o) r(t5o) v(t5o) γ(t5o) χ (t5o) m(t5o) α(t5o) Φ(t5o)
δ(tf) r(tf) v(tf) γ(tf) χ (tf) m(tf) α(tf) Φ(tf)

Figure 13. An example trajectory displaying the phases, the 'j' amount of mesh points in each phase and how the end points of each phase can be linked. It is also noticeable that α (AOA) and Φ (throttle) is included with the states. This is because in the method employed in this thesis, the controls are treated as States and the derivatives of the controls are seen as the actual controls by the NLP solver.

The accuracy of these piecewise polynomial approximations adds another kind of constraint to the trajectory optimization that is not present in the Shooting method. Another new constraint would be the boundary constraints are applied at the end points of each phase requiring the final state of the one phase to match the initial state of the following phase as seen in the figure below. As with the GA, a boundary constraint can be applied to the final state ensuring that it corresponds to the desired trajectory. All other flight constraints that were mentioned in the previous section are enforced at every grid point the user assigns them to.

To reiterate, the steps taken in the Radau Collocation method are:

1. Provide a guess for the variables (that is, the States and controls at every grid point along the trajectory and the time spans of each phase).
2. Find the analytical derivatives at every grid point in the trajectory using the guess that was input. Like with the shooting method, this includes interpolating the values of thrust, mass flow and aerodynamic forces from the engine and vehicle libraries at every point in the grid. Also create Piecewise Lagrange Polynomials with the input State and control variables.
3. Evaluate the error in the constraints on the States at every grid point, the boundary constraints on each phase and compare the derivatives of the Lagrange Polynomials at each grid point with the analytical derivatives (the left hand side of the ODEs) at every grid point.
4. An NLP solver is then used to modify the variables and repeat the same steps.

Thus these steps input variables into a function generator that outputs the constraint error and the value of the figures of merit. An NLP solver is then used to optimize the variables input into this function so that the constraints are satisfied and the objective function is minimized.

It is important to note that the accuracy of the piecewise Lagrange polynomial approximations of the States and Controls is dependent on the number of grid points that are used to model the trajectory. This would be expected given the wavy nature of Lagrange polynomials. Therefore, GPOPS-II contains a grid refinement routine that takes this into account by increasing the order of the Lagrange polynomials by adding additional grid points in each phase and determining if the new Lagrange polynomials also match derivatives with the analytical derivatives at the new collocation points. If not, then the grid will have additional grid points added and the NLP solver will be run again. This cycle will repeat until the error in the Lagrange polynomials is within a certain tolerance. The grid refinement is explained in more detail in reference [17].

4.4 Genetic Algorithm Optimization

In this section, the workings of a Genetic Algorithm will be described along with how it was employed in this thesis.

4.4.1 –Basics

Genetic Algorithms (GAs) optimize design problems using a method based on the ideas of natural selection and survival of the fittest. The GA does this by randomly creating a large group of possible solutions to the objective function. Each possible solution is a set of guessed values for the variables and is known as a *chromosome*, while the variables within a chromosome are known as *genes* (this method for representing chromosomes and genes is known as floating point representation and is one of multiple methods listed in chapter 3 of [27]. The GA then tries out every chromosome in the whole group (known as a *population*) by running them through the function generator (in this case the Shooting Method). The chromosomes are then ranked based on how well each satisfied the constraints and minimized the objective function (collectively called the chromosomes performance in this thesis). Afterwards, the genes (or variables) of the previous population are exchanged between the different chromosomes in the population in a process called *crossover* (analogous to mating) and some genes have their values randomly changed in a process called *mutation* (analogous to genetic mutation). A Selection function is used to decide which chromosomes are involved in these reproduction processes based on the rank each individual achieved due to its performance. Thus a new population of chromosomes is created and the GA repeats all the previous steps. Every time a population is run through the function generator, a *generation* has passed. This routine is continued until a specified number of generations has passed or if no more significant changes in the objective functions can be achieved. This is the method employed by the GA to attempt to find a feasible and optimal answer for an objective function. How well this method explores a design space is dependent on the user selected size of the population and the number of generations it is allowed to run. The details of how this method works, how it is applied to this thesis and how reliable it is are discussed in the following subsections.

4.4.2 –The penalty function and the Objective function

The processes of the basic Genetic Algorithm contain no restrictions on feasibility of the solutions obtained, unlike with NLP solvers. Thus there are no convergence conditions for the constraints requiring that the GA converge on a feasible solution. As a result, different methods and techniques have been developed to work with the basic GA processes in order to ensure that only feasible solutions will be selected. However, placing too much of a restriction on the feasibility of the GA solutions can, at times, hinder its ability to search the design space. Therefore, Steele-Flight uses a common method for guiding GAs toward feasible solutions while preventing it from converging toward a local minimum. The Penalty Function method penalizes solutions for violating the bounds of the constraints by adding a positive quantity to the objective function that the GA is trying minimize. Thus, the GA attempts to minimize the constraint error in order to minimize the objective function. The Penalty Function used in this thesis is based off of the one given in ref [27] pg75 Eq 4.15 and is only slightly modified into the equation below:

$$\emptyset(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{k=1}^{K} r_k G_k [g_k(\boldsymbol{x})]^2 \qquad \text{(Eq. 11)}$$

In this equation, *f(x)* is the objective function as a function of the input variables, while the rest of the equation refers to the penalties. The operator G_k is a Heaviside operator that is either 0 or 1 depending on whether or not the kth constraint is violated. The function $g_k(x)$ is the error in the kth constraint which has already been scaled to be of similar magnitude to all the other constraints. The operator r_k is kth multiplier for kth constraint. Its value can be user selected for each constraint in order to control which constraints are the most important. In this thesis the values of the multipliers used in the penalty function were found by trial and error process in numerous beta tests during the creation of this code.

The Objective function *f(**x**)* is meant to be user supplied depending on the desires of the user. In this thesis, the objective function of the GA when being used to find an initial guess trajectory took into account the payload and the fuel mass as the major figures of merit.

$$f(\boldsymbol{x}) = \frac{MaxPayload - Payload}{scale\_factor} \qquad \text{(Eq. 12)}$$

The *MaxPayload* term refers to the upper bound on the payload variable that is the code is attempting to reach, while the *Payload* term is the amount of payload that the current solution has guessed. The scaling factor in the denominator is chosen by the user based on the unit system being used. Additional specialized penalty factors can be added to the objective function for specialized situations. One such specialized factor is the volume constraint on the engine bays.

4.4.3 –Reproduction

GAs use two primary means of reproduction: crossover and mutation. In the case of crossover, which mimics sexual reproduction in a population of individuals, two individual chromosomes exchange genes which are chosen at random to produce new chromosomes for the next generation. However, if this was the only means of reproduction, no new genes would ever be introduced into the population after the initial population was created. Therefore, mutation occurs for certain chromosomes in order to introduce new genes. This process takes place by randomly changing a number of genes in the chromosome chosen to undergo mutation.

The third means of creating a new generation of chromosomes is by *Elitism*, which is an important concept in this thesis. When Elitism is used, the highest ranking chromosomes go on to the next generation to compete against their children for reproduction. In this manner, the best chromosomes will be preserved from generation to generation until superior chromosomes displace them. How many Elite there are in a population is user selected.

Which chromosomes undergo crossover and mutation are chosen in a Selection function which uses one of many standard or custom methods to select which chromosomes reproduce and how many times they reproduced, based on the ranking of each chromosome. However, almost all Selection functions involve some element of random selection to ensure some diversity in the Selections. Thus probability plays a role in the Selection of chromosomes in reproduction. For different types of selection functions the user is directed to reference [27].

4.4.4 –Migration

Migration is a method used in a GA to preserve genetic diversity and promote a more thorough exploration of the design space. Using this method, the total population is broken into multiple subpopulations that reproduce separately from one another except at certain intervals in the number of generations where the best members of one subpopulation “migrate” over to another subpopulation thus introducing new genes to the other subpopulation. In this manner, unique genes can be preserved in separate subpopulations, reducing the risk of these genes being wiped out early if they happened to be on inferior chromosomes. In addition to this, different subpopulations can have different constraints, thus favoring different propulsion methods, thus allowing a designer to explore multiple options.

4.4.5 –Drawbacks and benefits of GAs

The major drawbacks of using a Genetic Algorithm is the fact that they is not guaranteed to converge to the global optimum. Since GAs employ some element of randomness in their selection process they are referred to as probability based optimizers and the probability that the GA will converge to a near-optimum answer is dependent on the size of the population chosen and the number of generations that it is allowed to run. In addition to this, GAs are not very good at finding optimal trajectories since the precision of the trajectories that they can find is heavily dependent on the number of control variables used to manipulate it. Also, the more controls required, the more variables and thus the larger the populations must be and the more generations needed to reach convergence. The need to reduce the number of variables input into the GA is the primary reason why the Shooting method is used for the GA in this thesis instead of Collocation method and the reason why engine controls are not used for the Shooting method.

While the GA cannot guarantee convergence to a global optimum, they are much more robust at exploring design spaces than calculus based methods. Since they are not constricted by the derivatives of a problem they are free to explore large design spaces in a fast manner not requiring a large amount of tedious mathematics. Also, GAs are capable of exploring design spaces that are made up of both continuous variables (that is, variables with an infinite amount of possible values between their bounds) and integer variables (variables that can only take on integer values). Dependence on the derivatives of the input variables makes NLP methods incapable of handling integer variables and increases the likelihood that they will converge to local optimums. Thus, while GAs may not converge on a precise representation of the optimal trajectory, they are still useful for exploring the design space and finding a solution that is located in the vicinity of the global optimum. This solution will subsequently be very useful as an initial guess for starting the NLP method.

It is however, important to note that the "optimal" trajectories found in this thesis are only optimal relative to the initial guess trajectory found by the GA. While there is no complete guarantee that the GA will output an initial guess that is near the global optimum, it is still very useful in finding an initial guess that is better than one that would be guessed by a human. GAs have been used in this capacity in numerous other research efforts as will be briefly discussed in section 4.5.

4.4.6 –Implementation in Steele-Flight

The benefits of using GAs described in the previous section are exploited in Steele-Flight for two purposes: 1) To find an initial guess trajectory to start the NLP optimization and 2) to manipulate the integer variables that define the engine configuration while the Hybrid optimization is running.

4.5 Hybrid Optimization Using GA and NLP Together

The results of the GA optimization should yield a complete time history of all the states of the trajectory. This will then form the initial guess of the next portion of the thesis, the Hybrid Optimization section that uses both GA and the NLP method together. However, before proceeding, it should be noted that using GAs to find initial guess trajectories for an NLP optimizer is not an original idea, nor Hybridizing the GA with an NLP function. There are also numerous variations to methods of using GAs and NLP solvers together and this thesis gives yet another way to do so. However, these previous studies were focused purely on trajectory optimization. This study is unique because it includes engine selection in the process. For an example of a previous study involving Hybrid GA and NLP optimization methods the reader is directed to references [28].

4.5.1 –Overview

The GA is used once again, however, this time it only manipulates integer variables being input into the function generator. Also, this time, the GAs function generator is replaced by the program GPOPS-II which contains its own function generator (the Radau collocation method) and optimizer (the NLP solver). As seen in Figure 4 of Chapter 3, the GA inputs the integer variables for an engine configuration into GPOPS-II. GPOPS-II transcribes the trajectory optimization problem into an NLP problem using the initial guess trajectory as the start point for the search. It also uses the integer variables fed to it by the GA to define an engine configuration whose performance will be interpolated at each time step in the trajectory. GPOPS-II then attempts to optimize the trajectory using an NLP solver for each new engine configuration the GA feeds it.

4.5.2 –GPOPS-II modifications

As mentioned before, for the Hybrid method, the GA only manipulates the integer variables. This is because, GPOPS-II uses NLP methods to optimize the trajectory. These methods depend heavily on the derivatives of the input variables, the objective functions and the constraints. Since integer variables are not continuous, they have no derivatives and therefore cannot be optimized using an NLP. However, NLP methods are very robust at finding the true local optimal solution in the locality of the initial guess, thus the NLP is very necessary for finding the true optimal trajectory for the engine configuration in the neighborhood around the initial guess trajectory. For the remainder of the thesis, GPOPS-II will refer to both the collocation transcription method and the NLP solver.

A major drawback of using GPOPS-II as the function generator for the GA is the amount of time taken to optimize the trajectory. Each chromosome in each population for every generation of the GA must have its trajectory optimized in order to ensure that all possible engine combinations the GA can generate will get a chance to have its trajectory optimized so that its true potential can be seen. This leads to an enormous time requirement that would hurt the usefulness of this method.

In order to prevent the code from running too long, certain actions were performed to allow the user to control the amount of time the NLP solver was allowed to run. What actions were taken are based on the observations of the designer of Steele-Flight and will now be described.

Naturally, not every engine configuration would be able to produce a feasible trajectory that was capable of converging in the NLP solver. If the GA supplied the genes for an engine combination that was unable to power a feasible trajectory that satisfied all the constraints, then eventually, the NLP solver would spend an enormous amount of time trying to get the problem to converge before eventually reaching its iteration limit and citing the reason for its inability to converge in a numbered output flag. The NLP solver used by GPOPS-II in this thesis is SNOPT [29]. It is not built in to GPOPS-II and requires a separate license to be used with GPOPS-II. For detailed information about how both SNOPT and NLPs work, the reader is directed to references [29, 30].

Since SNOPT does output a numbered response indicating what happened and the fact that SNOPT does allow the number of optimization iterations to be modified, a method was devised to limit the number of optimization iterations used by SNOPT at different points in the generations of the GA. This is done because in the early generations, the GA will pick very random choices of engine configurations. Because of the lack of order in the selection, many of these choices are not likely to converge. By setting the amount of SNOPT iterations very low, all the runs will finish quickly and output a reason for the termination of the SNOPT run. If the reason is due to the number of iterations being reached, then the optimality of the outputted trajectory is assessed (despite the fact that it is not completely feasible because the optimization didn’t finish) and given a rank just as was done in the GA initial guess optimization. If however, SNOPT says that the reason for the termination was due some inability to converge, it is more than likely that this engine combo would never have converged and is heavily penalized reducing its likelihood of reproducing in the GA generations. Thus flawed engine combinations are caught early.

After a number of generations, the number of SNOPT iterations can be raised, so that better engine configurations can be more fully optimized. Also, faulty engine configurations that made it through the first ‘culling’ without showing any signs of non-convergence may be caught in later iterations.

The number of iterations used for SNOPT at different generations of the GA were determined by trial and error methods performed on initial trail runs to find an estimate of the minimum amount of iterations required by SNOPT to discover a convergence problem.

4.5.3 –The GAs role: Clans and Chiefs

Over the generations, eventually the best engine configuration possible for the initial guess trajectory will eventually dominate the populations, resulting in all the Elite looking very similar. However, it may be advantageous to know the full potential of multiple unique engine configurations. Since the method of ‘culling’ the initial populations of defective configurations may have the unintended result of eliminating the genes for more unique engine configurations later on, special steps are taken to preserve genetic diversity in the generations of the GA.

The method used to preserve diversity and find multiple optimal engine configurations is a ‘Niching method’ that was customized for use in this thesis. Niching methods are meant to allow multiple optima to be found by ensuring that certain gene combinations continue to be used even if there are other better gene combinations present. More on Niching methods can be found in Chapter 5 of [31]. The custom Niching method used in this thesis creates the conditions that chromosomes that have specific variables that match the same specific variables in another chromosome are a part of the same group called a *Clan* in this thesis. The Clans defined in this thesis are identified by the types of engine models used in each phase of the trajectory.

The Elite chromosomes of a population are composed of the best members of the best Clans and are known in this thesis as the *Chiefs*. How many Clans there are in a population is user defined and the GA in this thesis is modified to ensure that all Chiefs come from different Clans. Thus, the differing Clans that develop will compete to fill in the allowable number of Chiefs that will survive to the next generation. In this manner, genetic diversity is ensured since the elite members of the populations will always be genetically distinct and therefore multiple Chiefs will be found for comparison.

In order to use the Clans and Chiefs method effectively it is often best to not implement it in the early generations. Thus the user can specify which generation that the method should start being used in. The GA is also modified to record a time history of the evolving designs so that the user can see the potential of multiple designs and see the different Chiefs that have been modified over the generations.

4.5.4 –The Objective Function

The Hybrid method employs two objective functions to drive the NLP solver and the GA. The objective function for the NLP solver used in this thesis is relatively simple and is designed to maximize the payload and smooth the control history of the vehicles as seen below

$$f(\boldsymbol{x}) = -payload + Total\ Integral\ Cost \qquad \text{(Eq. 13)}$$

The parameter “payload” is already scaled down to match the magnitudes of other input variables and the negative sign in front is present because the NLP method used minimizes an objective function while payload must be maximized. The parameter “Total integral cost” is the scalar results of the integration of the time histories of the flight controls and is added to the objective function to prevent unrealistic fluctuations in the vehicle controls over the trajectory and thus "smooth" the trajectory.

The GA objective function for the Hybrid method is what determines the best engine configurations for the mission and vehicle based on maximizing the amount of payload delivered to the specified orbit without significant increase in the GTOW relative to other vehicle designs and is shown below:

$$f(\boldsymbol{x}) = -\left[\frac{payload * scale1}{GTOW} + \frac{payload}{scale2}\right] \qquad \text{(Eq. 14)}$$

The parameter "payload" is the payload mass in an unscaled form. The factors *scale1* and *scale2* are chosen to accommodate the differences in magnitude between the payload and GTOW and to scale the objective function. This objective function is set up so that the payload is the primary figure of merit driving the objective function. However, the GTOW is added to ensure that configurations that have only a slightly higher payload and vastly larger GTOW than other configurations are not treated as superior due to the massive amounts of additional costs incurred by higher GTOW just to increase the payload by a slight amount. Once again the negative sign causes the GA to see the maximization problem as a minimization problem.

The GA also employs a penalty function strategy as it did for the initial guess portion of the code. The equation is as follows:

$$\emptyset = f(\boldsymbol{x}) + Total\ Integral\ Cost + Total\ penalty \qquad \text{(Eq. 15)}$$

This equation adds the total integral cost from NLP output to penalize trajectories that show unrealistic fluctuations in the flight control time history. The total penalty term only takes into account discontinuities in State approximations from the NLP solver. This is only necessary in the first few generations of the GA because during this time, as mentioned previously, the NLP is being stopped prematurely and therefore not satisfying the feasibility requirements right away.

4.5.5 –Refined GPOPS-II run

The NLP method employed in this thesis optimizes the trajectory based on the number of grid points provided to represent the trajectory by the user. GPOPS-II checks the accuracy of the polynomial representations of the states of the trajectory based on the provided grid to ensure a realistic polynomial representation by performing a mesh refinement algorithm. This algorithm is time intensive, because it reruns the NLP solver multiple times, increasing the amount of points in the grid with each mesh iteration until the specified mesh tolerance is satisfied.

Due to how time intensive the mesh refinement can be, the process is not performed during the Hybrid Optimization step, but rather, is performed afterwards on the group of the highest ranking engine configurations in order to determine the best configurations with their best trajectories.

The optimum trajectory without mesh refinement, still allows the optimum engine configurations to be identified in the Hybrid process regardless of if the trajectories did not conform to the tolerance of the mesh refinement. This was found to be true by experience in running these tests and is verified in the next chapter.

Chapter 5- Implementation of Steele-Flight

Steele-Flight does not have another code to compare to directly, since no other code currently runs with the same objectives and because the majority of all similar studies use classified vehicle information that is not available to the public. Therefore, verification of Steele-Flight's individual functions must be done separately from one another.

5.1 Validation of Trajectory Optimization

5.1.1 –Problem Setup in Steele-Flight and ASTOS

The first function that was validated was Steele-Flight's use as a trajectory optimizer. If engine selection is turned off, then the code should function as a typical trajectory optimizer. The method used to verify Steele-Flight's ability to optimize trajectories is by having it optimize an example trajectory that is also run in the commercial code known as ASTOS.

This example trajectory optimization problem along with all specifications and data libraries on aerodynamics and propulsion are provided with ASTOS as an example for newer users to run in order to understand the code. This experiment is documented in [15], which is one of ASTOS' example pdfs

The trajectory to be optimized is the flight of the conceptual TSTO RLV known as the Saenger Space Plane. The objective is to find a trajectory that will allow the vehicle to deliver the maximum amount of payload possible into Low Earth Orbit (LEO) given the vehicle and engine design and the constraints the mission is subject to.

The trajectory itself is separated into three separate phases of flight, each of which use a different type of engine. The first stage of the vehicle only uses air breathing engines throughout phases 1 and 2. The second stage of the vehicle separates from the first stage at the beginning of phase 3 and takes the payload into orbit. The engines used, as listed in Table 1, are provided in the ASTOS example as data libraries that are functions of different flight properties.

Phase	Engine Type	Data Library was a function of	Number of Engines	Stage
1	Turbojets	Altitude, Mach	5	1
2	Ramjets	Altitude, Mach, AOA, Equivalence Ratio	5	1
3	Rocket	No libraries used.	1	2

Table 1. Engine information for Saenger Engines

All the engine libraries were provided in the example set up in ASTOS and are meant to represent the engine models intended for the Saenger Space plane, but where the data provided comes from is unknown to the author of this thesis and no information on weight and size of these engines was provided. Thus, all physical characteristics are incorporated into the ASTOS provided dry mass of the vehicle structure specified. The turbojet engine libraries for thrust and mass flow rate were a function of altitude and Mach number. The ramjets libraries were similar but were also a function of vehicle angle of attack and Equivalence ratio. On the other hand, data for the rocket was provided in the form of the exit area of the nozzle, the vacuum Isp and the propellant mass flow rate, and was subject to the effects of back pressure. All engine libraries used linear interpolation.

The aerodynamic data for both stages of the Saenger combined together and for the second stage alone, is provided by ASTOS in the form of data tables for the lift and drag coefficients that are functions of Mach number and AOA. A reference area is also supplied for both stages. While published data for the Saenger does exist [16], it doesn't appear to correspond exactly with the data provided in this example. It is possible that these aerodynamic libraries already take into consideration the effects of engine integration but it is unknown to the author of this thesis. Knowing the source of all the data used in this example is not of extreme importance as this test is only being performed to validate Steele-Flight's ability to optimize trajectories in general.

All data concerning the weight of the engines and the weight of the fuel tanks is already incorporated into provided weights of the first and second stage. The type of fuel is not specified, but the mass of fuel on board each stage, as well as the payload mass are decision variables.

The trajectory is set up in ASTOS in a fashion that allows the plane to maneuver in three dimensions. The mission begins in Istres France headed South toward the equator. The plane must turn east along the equator in order to launch its payload into its target orbit. The final state of the trajectory must correspond with a target orbit which has an apogee altitude of 450km, a perigee altitude of 70km and is at an inclination of 28.5^o to the equator. The orbit location with regard to the longitude is not taken into account. In addition, the final altitude was constrained to be equal to or above 74km. The trajectory is optimized by modifying the input variables which consist of:

- The amount of flight time taken in each phase.
- The total amount of fuel aboard each stage of the vehicle.
- The flight control angles which consist of the Angle of Attack (AOA), the Pitch Angle, the Yaw Angle and the Bank angle.
- The throttle for the phase 2 ramjets.
- The total payload.

All the given mission data and required constraints were input into Steele-Flight so that both codes could optimize the same trajectory given the same input data and the same constraints. However, there were some compatibility issues that first needed to be addressed. Steele-Flight does not currently support 3D analysis (motion in the vehicles horizontal plane), thus the example in ASTOS had to be changed into a 2D flight along a vertical plane that starts at an arbitrary point on the equator (longitude is ignored since it doesn't affect the other State ODE's) and flies along the equator rather than starting in France and turning along the equator. The result of switching to a 2D vertical plane is a reduction in mission complexity due to a reduction in the amount of flight controls requiring optimization. The AOA and the Pitch angle are the only two flight controls that remain. Also, while the perigee and apogee altitudes of the orbit remain unchanged, the orbit's inclination is now fixed along the equator. Once the changes were implemented the initial values of the trajectory States at beginning of the trajectory were as seen in the table below:

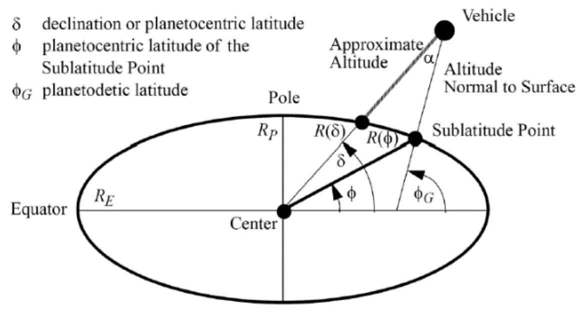
δ declination or planetocentric latitude
ϕ planetocentric latitude of the Sublatitude Point
ϕ_G planetodetic latitude
Vehicle
Approximate Altitude
α
Altitude Normal to Surface
Pole
R_P
R(δ)
R(ϕ)
Sublatitude Point
δ
ϕ
ϕ_G
Equator
R_E
Center

Figure 14. Example of planet modeled as Oblate Spheroid. Fischer, D., P. Gath, and K. Well, *ASTOS Model Library Version 7.2.0*. 2006-2014 [2]. Used with permission from Francesco Cremaschi, ASTOS Solutions GmbH, 2015.

Declination (deg)	Altitude (km)	Velocity (m/s)	Flight Path Angle (deg)	Velocity Azimuth (deg)
0	0	210	0	90

Table 2. Initial flight States at the initial time.

Yet another compatibility obstacle was the fact that ASTOS modeled the planet as an Oblate spheroid in order to be more realistic. Steele-Flight only models the earth as a perfect sphere in order to reduce the number of possible errors that could occur in the optimization process. Thus, the ASTOS model was changed to using a spherical earth model (which it has as an option). Again, this also reduces the complexity of the problem, since finding the Altitude for an Oblate Spheroid at each Latitude line requires an iterative process.

A final modification made was the removal of the throttle control for the ramjet. Although, Steele-Flight is more than capable of controlling the throttle of the engine, this control was removed from the verification process to save time and reduce the complexity of the problem.

A final note concerning this test is the fact that all inputs and outputs presented for this test will be displayed in metric units since these are the units that the ASTOS test uses.

5.1.2 –Finding the initial guess

In order to optimize a trajectory using a Non-Linear Programming solver an initial guess trajectory is required to begin the optimization. In this work, the method used by ASTOS to find an initial guess trajectory (which is the default method used in the ASTOS TSTO example) is to use control laws. These are closed-loop controllers that control the flight control angles (AOA, Pitch, etc.) while the dynamic flight equations are propagated in a Runge-Kutta solver. Thus, while the flight is being simulated, the closed-loop controllers control the AOA during the first two phases of flight, causing the plane to fly toward the specified maximum dynamic pressure limit and then controlling the pitch in the third phase to cause the plane to fly toward the specified Orbit. This initial guess method is used for comparison purposes since ASTOS was already set up for this method. Since this is just an initial guess, the final orbit obtained is far off target.

Steele-Flight uses the GA built in to MATLAB to generate an initial guess since the GA is Steele-Flight's only option for generating initial guesses. However, all the same constraints and variables that were used in the ASTOS final optimization were used in both the Steele-Flight GA initial guess generator and final optimization with only minor variation. The results can be seen in Section 6.1.

5.2 Engine Selection Validation Study

Steele-Flight's main function (optimal engine selection and use) was also put to the test in the study that is described below. This study exercised Steele-Flight's ability to find optimal engine configurations and corresponding optimal trajectories and is described in subsections 5.2.1 and 5.2.2. Afterwards, the method used to validate the results of this study is documented in subsection 5.2.3.

5.2.1 –Steele-Flight setup

For convenience, Steele-Flight was set up using the same model of the Saenger Space plane as was used in the Trajectory verification portion. The same aerodynamic libraries for each stage and similar structural weight were also used and the mission was to reach the same LEO. Since the estimated weight of the fuel tanks was already integrated into the ASTOS guess at the structural weight, these tanks will be assumed to be a constant size and weight. According to published documents concerning the Saenger design, the intended engines were all Hydrogen powered [6, 10]. Also, the Isp values seen in the ASTOS provided data for the turbojets and ramjets are far too high for any hydrocarbon powered engines. Therefore, it is reasonable to assume that the tanks in the ASTOS model are meant to handle Hydrogen and Oxygen (for the rocket). Because of this, all the engines that are used for this study are Hydrogen powered (with Oxygen for the rockets). The engine pool used for this study is detailed in the next subsection.

The trajectory was broken into 5 phases, with stage separation happening at the beginning of the 5th phase. The 1st and 2nd phases of the trajectory could use either turbine engines or the ducted rocket mode of an RBCC system. The 3rd phase of the trajectory is fixed for ramjet operation and the 4th phase is fixed for scramjet operation. The reason for this is due to the lack of competition against ramjets and scramjets in their respective Mach regimes. The 5th phase represents the flight of the second stage of the vehicle which enters orbit, thus the only options for this phase are pure rocket engines.

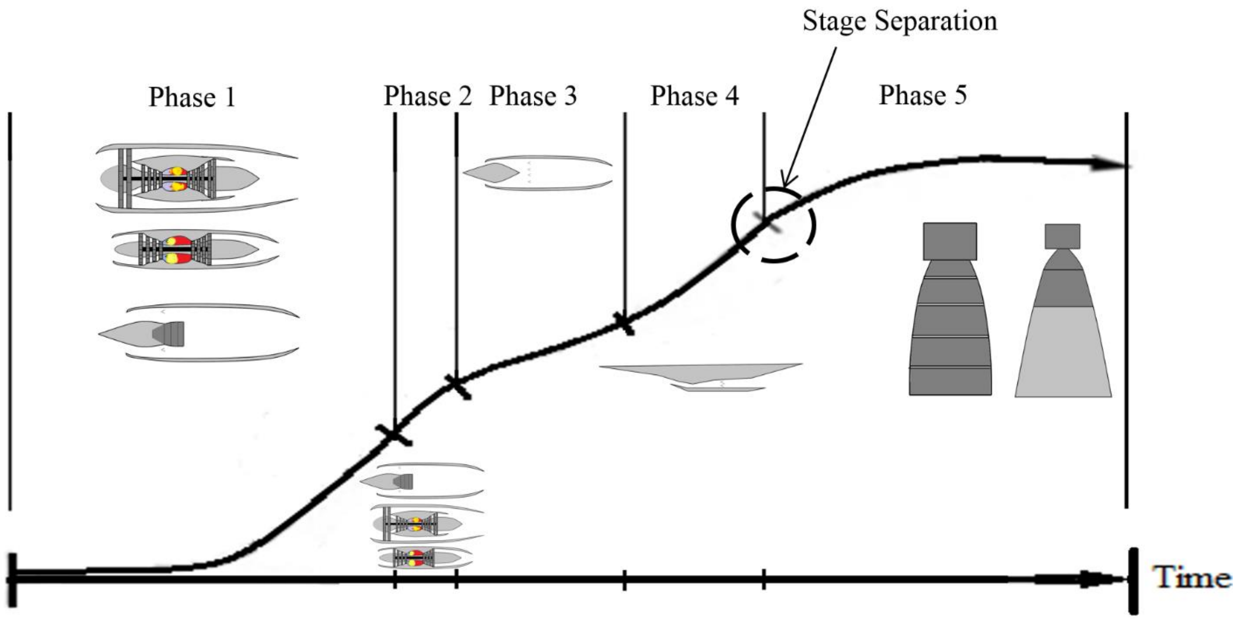

Figure 15. Trajectory displaying the engine options in different phases of the trajectory.

Steele-Flight optimized by modifying the input variables which consisted of:

- The flight control angles which consisted of AOA and Pitch angles
- The amount of flight time taken in each phase.
- The total amount of fuel aboard each stage of the vehicle.
- The payload mass.
- The amount of first stage turbine engines allowable.
- The amount of first stage ramjets allowable.
- The amount of first stage scramjets allowable.

- The amount of first stage rockets allowable.
- The amount of second stage rockets allowable.
- The choice of turbine engine model to use.
- The choice of second stage rocket model to use.
- The engine type to use in phase 1.
- The engine type to use in phase 2.
- Engine mode in phase 5 (for GA initial guess)
- The Flight path states (GPOPS-II only).
- The engine controls (GPOPS-II only).

The bounds for these variables are listed in Appendix A.

The only engine mode variable was for the final stage rocket and is only applicable to the initial guess generator because the two settings are max throttle and min throttle. The modes are constant in the initial guess, however, in the NLP method, the throttle setting is an optimizable parameter and thus this particular engine mode no longer applies.

This study was also subjected to constraints on dynamic pressure, Mach number and altitude. The constraints on dynamic pressure were kept the same as in the ASTOS example because they identified the structural limits of the vehicle, while Mach number and altitude constraints were chosen based on the maximum engine limitations. The acceleration was constrained in the orbital phase to stay within 3gs thus allowing the possibility of human passengers.

Additional constraints were also added to ensure that the engines and fuel stayed within the geometrical limits of the vehicle using the techniques outlined in Section 4.1. The density of the liquid hydrogen and liquid oxygen was assumed to be the same as that used on the Space Shuttle main fuel tank.

The initial guess GA run was set up to run for 200 generations using the migrations model which divided the population into three subpopulations which had the following number of chromosomes {440 352 352}. The largest subpopulation did not favor any particular species of chromosome, while the two smaller subpopulations were restricted to only allowing turbine engines or ducted rockets in the first phase respectively. This measure was taken to ensure that neither method of first stage propulsion dominated too early in the generations.

While the initial guess part is mainly for finding an initial guess trajectory, initial guess engine configurations are part of this process because feasible engine configurations are not initially known.

Once the initial guess was obtained another set of GA options was set up for the Hybrid method. The Hybrid method was set to run 20 generations with a single population of 88 chromosomes. The amount of major iterations that the NLP solver was restricted to was 100 for the first nine generations, then increased to 150 for generations ten through fourteen and then set 500 from the fifteenth generation to final generation. These numbers were selected based on experience from previous trial tests to see how long the NLP solver typically took to converge to a solution and to determine how long the NLP took to determine whether or not the problem would ever converge. Also, the Clans and Chiefs model was used, starting in the fourth generation and allowing 5 Chiefs.

The outputs of this test will be English Engineering units (EE).

5.2.2 –Steele-Flight Experimental Engines

The two engines used to represent the turbine engine type were built in GasTurb11. Both used performance libraries that were functions of Altitude, Mach number, AOA and Engine Throttle. The larger engine is a two-spool low-bypass ratio mixed flow afterburning turbofan. The engine is not based significantly on any currently existing engine. Rather it was designed to get thrust and mass flow value at its design point to be similar to the turbojet engine used in the ASTOS example. However, this only occurs at certain conditions around the design point. For most of the trajectory, the turbofan engine gives a higher thrust and Isp than the Saenger turbojet library provided by ASTOS.

The smaller of the two turbine engines used for this study is a two-spool afterburning turbojet engine with the same core design as the turbofan model. That is, it uses the same compressor and turbine on the inner spool and has the same maximum burner exit temperature. This engine model ended up having a slightly larger average Thrust-to-Weight (T/W) ratio over the flight regime than the turbofan model and it had a larger average Isp. However, it was smaller and had a lower thrust.

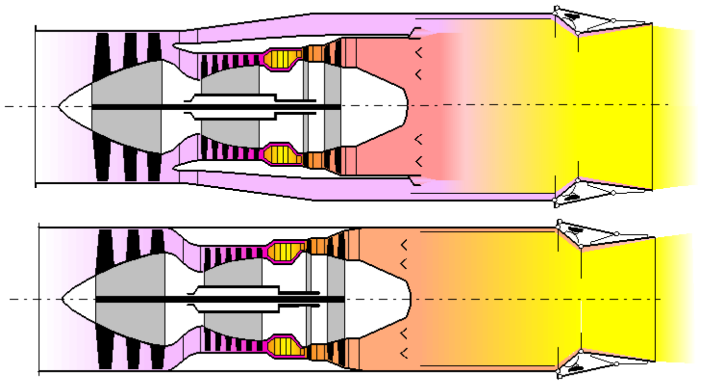
Figure 16. GasTurb models for a Two Spool Mixed Flow Turbofan (top) and a Two Spool Turbojet (bottom). Kurzke, J., GasTurb11. 2007 [1]. Used with permission from Dr. Joachim Kurzke, 2015.

The only method supplied by GasTurb11 that was used to account for the effects of the airframe on the engine inlet was the use of an inlet pressure loss map which was used to account for the effects of different vehicle AOA on the performance of the engine. This data map was obtained from the SCCREAM model for the ramjet performance, since this model takes into account the pressure losses due to engine airframe integration. The data was then imported into GasTurb 11 and used when creating the data libraries for the turbine engines.

An RBCC model created in SCCREAM was used in this study. This model allowed for multiple modes of operation from Mach 0 to 10.3 using only one engine model. The system incorporated a ramjet mode, a scramjet mode and a ducted rocket mode that allowed atmospheric air intake into the engine, while the engine was being run as a rocket.

The design of this engine was based off of another Conceptual RBCC engine design used for a conceptual RLV known as the ABLV-GT2 which was modeled in SCCREAM by Dr. Bradford in reference [20]. In this reference, the most important input parameters used for the 2D design of the engine in SCCREAM were given. The few parameters not given, were estimated based on knowledge of ramjet and scramjet design. For this experiment, the model was greatly reduced in width to accommodate the Saenger vehicle geometry. Since SCCREAM doesn't estimate the weight of engines, this value was assumed based on comparisons with similar systems.

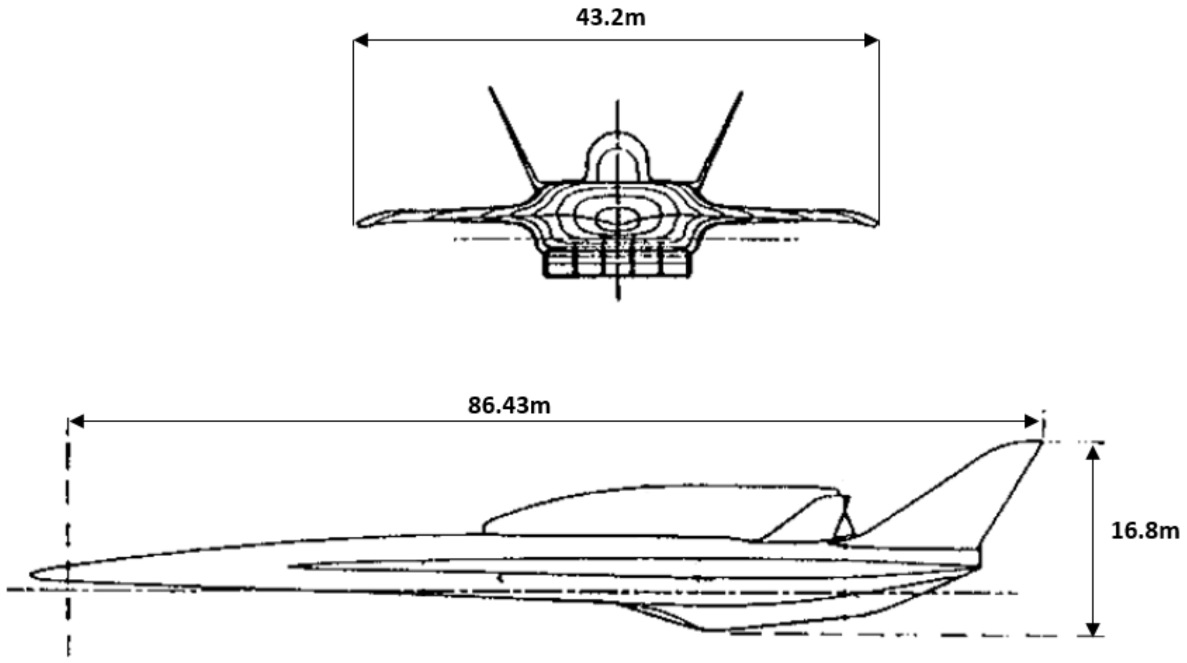

Figure 17. Geometry of the Saenger Vehicle. The image is adapted from Weingertner, S., *SAENGER - The reference concept of the German Hypersonics Technology Program*, in *5th International Aerospace Planes and Hypersonics Technologies Conference*. 1993, American Institute of Aeronautics and Astronautics [6]; reprinted by permission of Heather Brennan, American Institute of Aeronautics and Astronautics, Inc., 2015.

The Saenger vehicle is shaped as a double delta wing structure rather than a wedge shaped lifting body as seen in the Figure 17 above. Because of this, the inlet ramp for the RBCC system would, in reality, vary in the width direction. However, due to time constraints, and the fact that this would not impact the purpose of this study (which is simply to see if the code can choose the best engines based on what it is provided), this inconsistency was ignored as it was in the ASTOS example. Also, any Aerodynamic effects that would be caused by having more engines on the underside of the vehicle is also ignored due to limited information.

This engine model is the only choice for ramjet and scramjet engine types provided for this experiment. Therefore, every possible design that Steele-Flight could come up with uses a ramjet and a scramjet.

The rocket engines used for this study were created in Red-Top Lite. They are taken from premade models that come with the software and are not modified in any way for this thesis. The two engine models that were used were the Space Shuttle Main Engine (SSME) and the J-2X rocket engine. Both are fueled by liquid Hydrogen (LH2) and liquid Oxygen (LOX). The SSME is more powerful and has a higher Isp due to its higher O/F mixture ratio. It is however, much heavier than the J-2X. Details of these engines can be seen in Appendix B.

5.2.3 –Verification of Output using the 'Brute Force' method

As mentioned before, Steele-Flight works in the following steps (see Figure 4):

1. Steele-Flight uses a GA to find an initial guess trajectory.

2. The Hybrid method is used in which the GA optimization is performed in tandem with the NLP optimizer, GPOPS-II. This time the GA only varies the engine variables to create populations of engine configurations. Each engine configuration generated by the GA is plugged into GPOPS-II, which searches for the optimal trajectory for that engine configuration using an NLP solver and the previously determined initial guess. The GA takes the output of GPOPS-II to determine how well each engine configuration was able to minimize the objective function. In this manner, the GA determines which engine configurations are the best.
3. Once the GA has chosen the best engine configurations, Steele-Flight plugs these configurations into GPOPS-II which performs trajectory optimization with its mesh refinement algorithm turned on in order to get a more accurate trajectory.

In order to confirm that the engine configurations chosen by the GA optimizer (in Step 2 of the above process) are the optimal choices of engines given the chosen structural geometry, aerodynamics data and pool of engine choices, the following verification method was used.

The author manually picked nearly every possible engine configuration that could come from the pool of engine choices mentioned in sections 5.2.2 and plugged each one into GPOPS-II. Meanwhile GPOPS-II was set to run with the mesh refinement algorithm turned on and used the initial guess found in Step 1. Thus, the optimal engine configuration was found by an exhaustive search of the available design space rather than using a GA. Because of the time intensive nature of this method, it is referred to in this thesis as the ‘Brute Force’ method.

Using the Brute Force method, the optimal engine configuration is known beyond a doubt. Thus the results of this test were compared to the results output by Steele-Flight to confirm that the GA used in Steele-Flight had picked out the best engine combinations. Also, the results output by the Brute Force method are presented in this thesis to show the trends in engine performance and gain insight into the reasons behind the choices made in Steele-Flight.

Due to the time intensive nature of this method, not every combination of engines was actually run using the Brute Force method. Instead, only enough to prove the validity of Steele-Flight is shown.

Chapter 6- Results

6.1 Results of Trajectory Optimization Validation

In order to verify Steele-Flight's ability as a trajectory optimizer, it was compared to the validated trajectory optimizer known as ASTOS. Both programs were provided the same mission, flight constraints, engine models and vehicle model. The comparison of the outputs of each program is documented below.

The ASTOS initial guess started with a baseline payload of 7000kg and upper and lower bounds on the fuel mass of {0-90}Mg for the first stage and {0-85}Mg for the second stage, thus the same were used for Steele-Flight's GA.

The results of the Steele-Flight GA initial guess run can be seen in Figure 18 plotted with the initial guess results from ASTOS. The black lines represent Steele-Flight GA results, while the red lines represent the ASTOS initial guess.

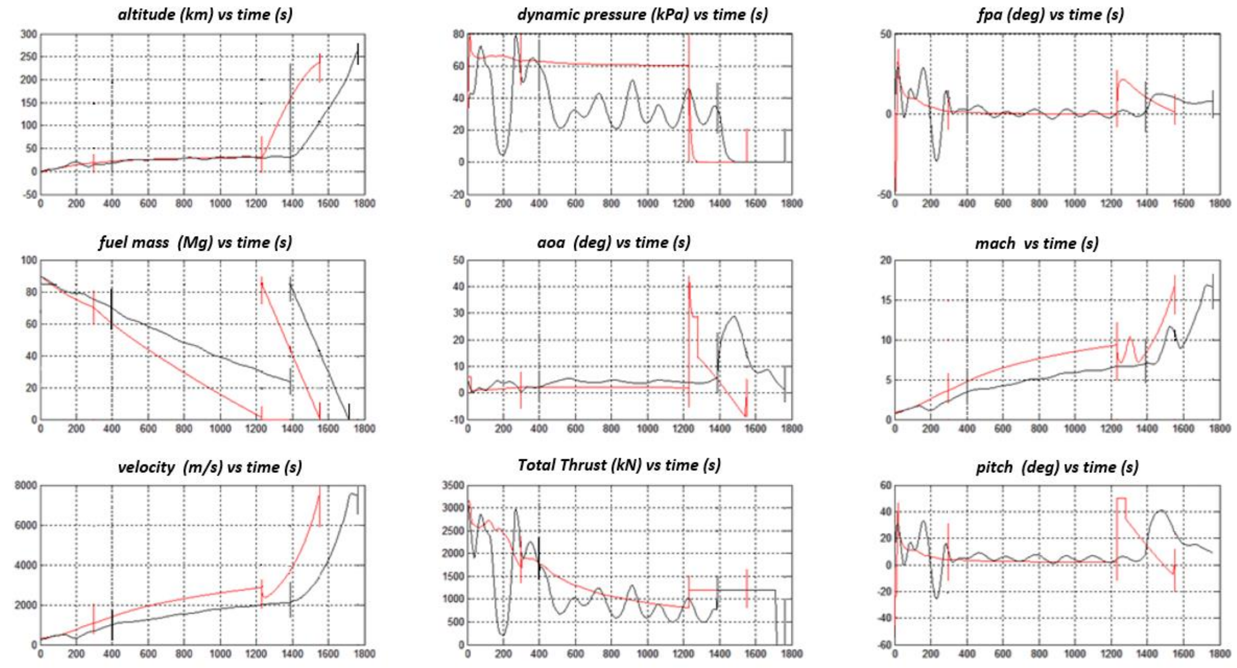

Figure 18. The results of Steele-Flight's initial guess trajectory (black) plotted next to the ASTOS initial guess trajectory (red). The vertical lines represent the phase end points. In the case of the graph for fuel mass, the fuel mass for both stages is shown.

After these initial guesses were obtained, NLP optimization studies could be performed on each initial guess to maximize the payload.

ASTOS used the optimizer known as CAMTOS [32] and used the default settings for this optimizer which were set up for the example problem. These settings included an optimization tolerance of 1.0e-5, a Constraint tolerance of 1.0e-6, an ODE tolerance 1.0e-4 and a Relative Integration error of 1.0e-8 (for an explanation of these tolerances the reader is directed to [33]). GPOPS-II was then set to the same tolerances.

The optimization was subject to the following path constraints: a dynamic pressure of {20-60} kPa during phase 1 and 2; an acceleration upper limit of 34.5m/s^2 during phase 1 and 2; and a Mach number upper limit of 7 in phase 2. Also, end point constraints were applied to the end of each phase: the Mach number of phase 1 was to be 3.5; the final Mach number of phase 2 was to be 7, the final dynamic pressure of phase 1 was to be between {20-60}kpa; the final orbit was to have perigee altitude of 70km and an apogee altitude of 450km. After optimization it was seen that the desired orbit perigee and apogee altitudes were met with the final altitude reaching the required 74km. Since the orbit was along the equatorial plane, the orbit inclination was 0. ASTOS yielded a payload 9830kg and an initial stage 1 fuel weight of 70Mg. The fuel mass of stage 2 stayed at 85Mg, thus yielding an initial GTOW of 296,320kg.

The Steele-Flight GA initial guess, when optimized in GPOPS-II using similar variable bounds and the same constraints (except for acceleration constraint as this was deemed unnecessary since the acceleration never exceeded 15m/s^2) yielded a final payload of 9855kg and an initial fuel mass in stage 1 of 67Mg. Just as with ASTOS, the fuel mass of stage 2 remained at 85Mg, thus yielding a GTOW of 293,345kg. The results of both tests are plotted alongside one another below. The black line indicates the Steele-Flight output, while the red line indicates the ASTOS output. The straight vertical lines indicate the phase separation points.

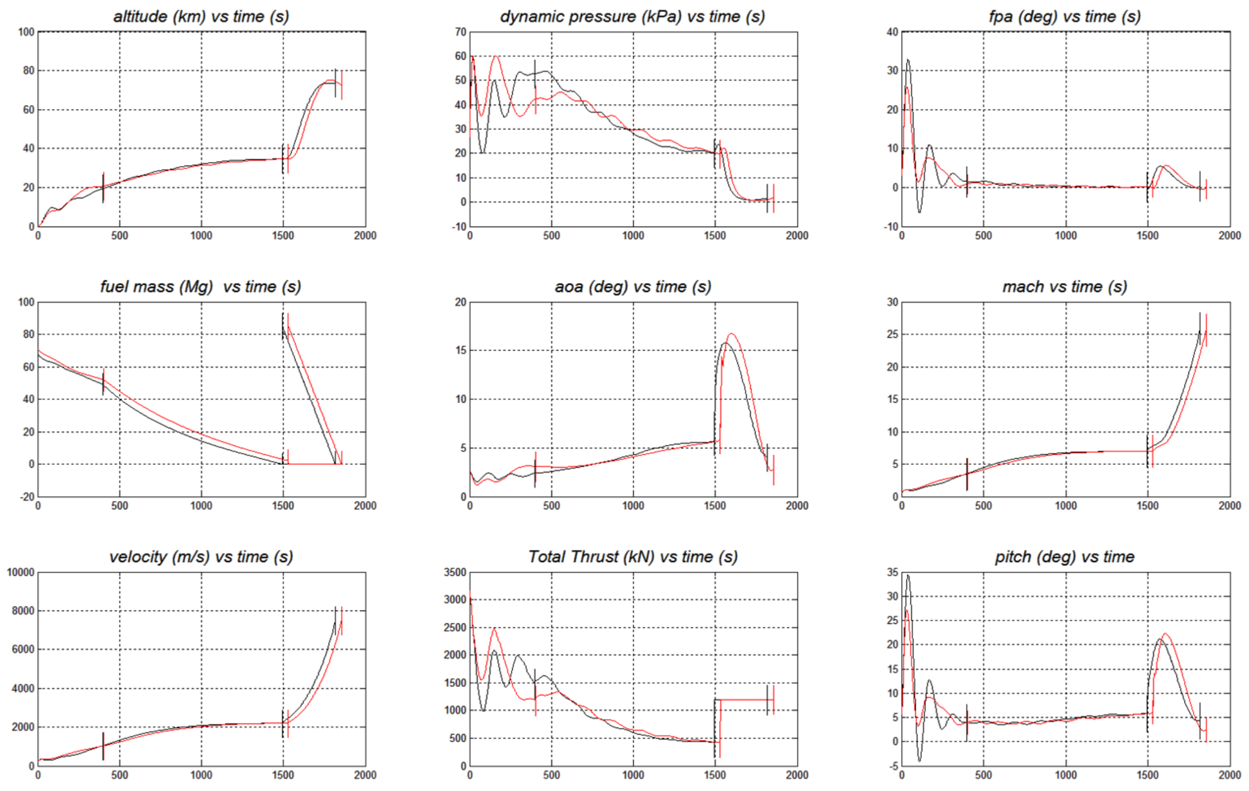

Figure 19. The results of Steele-Flight's final optimization output (black) alongside the ASTOS final optimization output.

The results show that the payload output from GPOPS-II was only 0.25% higher than the ASTOS output. The initial fuel mass for the first stage output by GPOPS-II was 95.7% of the initial fuel mass of the ASTOS output. The GTOW output by GPOPS-II was only 98.99% of that output by ASTOS. As can clearly be seen in the graphs above, the phase-times line up extremely well, with the largest deviation being only 3% at the end of phase 2.

It is noticeable that there are some oscillations in the first phase that differ between the two optimizations. Such small differences should be expected since the two optimizers use different initial guesses.

6.2 Results of Engine Selection Validation Study

The ability of Steele-Flight to choose the optimum engine configurations to minimize an objective function were tested and validated by comparing the output configurations generated by Steele-Flight with the results of a Brute Force method.

This Brute Force method was an exhaustive search of the design space which was performed by the author manually selecting every possible engine configuration and plugging them into GPOPS-II to find their optimal trajectories. In this manner, nearly every possible solution to the problem was found and thus the true optimal engine configuration was found beyond a doubt. When Steele-Flight chose the same engine configuration, its ability to choose optimal engine configurations from a broad selection of possible configurations was confirmed.

Before the results of Steele-Flight are shown in subsection 6.2.2, the outputs of the Brute Force method are examined to explain why the optimum solution found by both methods is the optimum solution and to examine the reasoning behind the choices made by Steele-Flight.

6.2.1 –Results of the 'Brute Force' Method

In the end, the Brute Force method revealed that the best engine configurations used 5 turbojets, 5 ramjets, 5 scramjets and 1 SSME engine and the second best configuration used 4 turbofans, 5 ramjets and 5 scramjets and 1SSME. These results can be verified in Appendix C.5. However, since the Brute Force method finds the most accurate trajectory (relative to the tolerances set in GPOPS-II) for every engine combination, the results of these trajectory optimization runs can be examined to see the trends in performance created by the variation in the engine configurations. Thus a number of trade studies were performed using data from the Brute Force method, the results of which, provide justification for the outputs of Steele-Flight. These studies are documented below.

<u>Study on the performance effects of varying turbine engines and second stage rockets.</u>

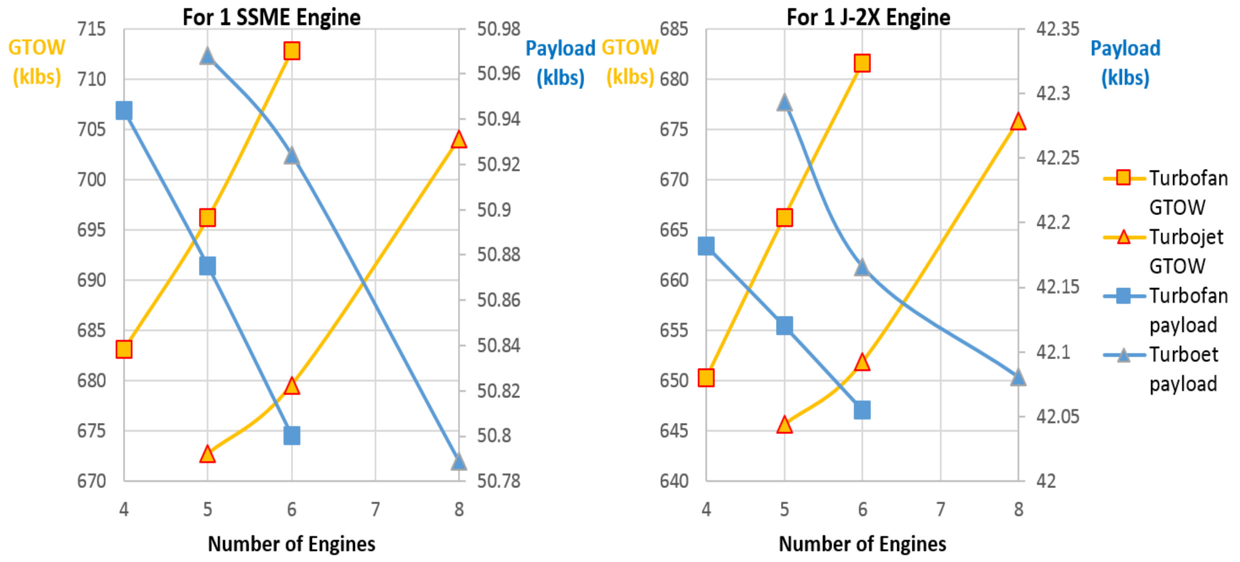

Figure 20. Trends for changes in payload and GTOW for increasing the number of turbine engines. The left chart displays the trends when an SSME engine is used for the second stage, while the right chart uses a J-2X engine.

The results from a number of tests that varied the number of turbine engines, the model used for turbine engines and the model of second stage rockets while keeping the number DMSJ engines constrained to 5 can be seen above in Figure 20. Results including one rocket engine are shown. No test that used two second stage rocket engines converged due to the acceleration constraints on the final phase. Analysis of these results yielded the following conclusions:

1. The J-2X engine is simply not competitive with the SSME engine when it comes to maximizing payload. The SSME engine has both a higher thrust and a higher Isp than the J-2X engine. Besides this, the SSME engine allows the second stage of the vehicle to carry more propellant (See the charts in Appendix C.1). Because the J-2X engine has a lower O/F mixture ratio, it requires a greater amount of hydrogen for each kg of oxygen burned. Thus the proportions of the LOX and LH2 stored reach the volume limit with a lower propellant mass than that reached when the SSME is used. This would end up being the case regardless of what the first stage engines were because the first stage only contributes to the payload by accelerating as much second stage mass as possible to the maximum altitude and Mach number that the scramjet can attain. Beyond this limit, it can do no more.
2. The performance of the turbojets and the turbofans is so close that the true determining factor between the two of them would be the GTOW and thus the amount of fuel required. This ultimately comes down to the fact that the turbojet has a higher Isp.
3. The trends in this graph indicate that fewer turbine engines actually cause an increase in total payload. The reason for this lies in the amount of time allocated to the phases. Given enough time to accelerate the mass, less turbine engines can do the same job as more turbine engines and thus the DMSJ phases don’t have the weight of unneeded additional turbine engines weighing them down. Thus more payload can be carried by the first stage. Details of how this conclusion was arrived at can be seen in Appendix C.2.

Study on the effects of varying the number of ramjets and scramjets.

After the above tests were performed, further tests were performed to measure the sensitivity to the number of ramjets and scramjets used. The results are seen below in Table 3:

Active Turbine Engines	Active Ramjets	Active Scramjets	Active 2nd Stage Rockets	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)
4 Turbofans	5 Ramjets	5 Scramjets	1 SSME	683,080	296,180	50,944
4 Turbofans	4 Ramjets	4 Scramjets	1 SSME	688,510	295,580	50,349
4 Turbofans	4 Ramjets	5 Scramjets	1 SSME	682,530	296,170	50,938
4 Turbofans	5 Ramjets	4 Scramjets	1 SSME	692,700	295,580	50,341
5 Turbojets	5 Ramjets	5 Scramjets	1 SSME	672,720	296,200	50,968
5 Turbojets	4 Ramjets	4 Scramjets	1 SSME	681,190	295,590	50,354
5 Turbojets	4 Ramjets	5 Scramjets	1 SSME	675,480	296,210	50,970
5 Turbojets	5 Ramjets	4 Scramjets	1 SSME	684,660	295,620	50,389

Table 3. Results from tests varying the number of ramjets and scramjets. The highlighted engine configurations correspond to those that were eventually chosen by Steele-Flight.

From analyzing the table, the following observations have been made:

1. The sensitivity of GTOW and payload to decreasing the number of ramjets is insignificant in comparison to its sensitivity to decreasing the number of scramjets. This makes sense, since the scramjet is the final engine in the first stage, therefore, what it can lift to the maximum staging conditions is far more important than what the ramjets can lift since the scramjets are on the final phase of the first stage trajectory.
2. The most interesting results to take away from the above table is the fact that decreasing the number of ramjets to 4, while keeping the same number of scramjets, creates very minor changes in the overall performance when compared to using 5 ramjets. The trends also differ between the turbofan configurations and turbojet configurations. For the turbojet configuration, the graphs of which can be seen in Appendix C.3, it can be noticed that the 4 ramjets yield a slightly higher payload, but at the cost of a significantly higher GTOW. This results because the 4 ramjets run longer than the 5 ramjets, requiring more fuel and only making it to a slightly lower altitude than the 5 ramjets did. Thus the 4 ramjet configuration has a GTOW 2,760lbs higher than the 5 ramjet configuration. The results however, are slightly different for the case for the 4 turbofans, which yield a lower GTOW for the 4 ramjets but also a lower payload which causes the objective function (defined in Equations 14 and 15) to favor the 5 ramjet configuration. The fact that differences in GTOW and payload are relatively small, highlights the ability of the trajectory optimizer to compensate for changes in the engine configuration.

<u>Study into the effects of using ducted rockets in the first two phases.</u>

More tests were performed to test the merits of using ducted rockets. None of these tests compared well with the tests using pure Turbine Engines for the first stage. Tests that used ducted rockets in both phases 1 and 2, did not converge, because they required so much propellant that they exceed the maximum fuel volume limits of the first stage. The tests that used a mix of ducted rockets and turbine engines in phases 1 and 2 did converge and obtained payload values comparable to configurations that used pure turbine engines. However, the pure Turbine engine configurations had much lower GTOW values due to the lack of need for stored oxygen.

Due to the sheer number of possible engine combinations that could result from using ducted rockets with turbine engines in the first stage, validation cases were run after the Steele-Flight results were obtained. The ducted rocket configurations that were obtained were validated by testing the engine configuration designs chosen by Steele-Flight and testing those configurations that were similar to those chosen by Steele-Flight. The data resulting from these tests can be seen in Appendix C.4.

The outputs of every single test performed with the brute force method can be seen in Appendix C.5 with the corresponding result of the objective function (also used in Steele-Flight and defined in Equations 14 and 15) that was used to rank each engine configuration. The results from all of these tests yield one very interesting point. All of the configurations cannot get past a maximum payload reaching 51,000lbm. Thus, it is reasonable to believe that this is the maximum payload that the second stage can get to orbit given the engine options available for the second stage. How much more payload the first stage could have carried if the second stage could lift more, is the subject for a future test with more options for the second stage propulsion systems.

6.2.2 –Results of Steele-Flight

The Steele-Flight program was run with the objective of determining the engine configuration that could deliver the most payload to orbit with the lowest amount of required propellant relative to rival configurations. The test used the same initial guess and constraints as the 'Brute Force' method and was set to run for 20 generations using the clans and chiefs method described in chapter 4. Using this method, Steele-Flight would output 5 unique optima (the chiefs), ordered according to which chief performed the best. Each chief would be the best representative of their clan (recall that a clan is a unique combination of engine types, thus all engine configurations using turbojets in phases 1 and 2 and SSME rockets in phase 5 are part of a single clan). Thus, while Steele-Flight's objective is to find the single best engine configuration for the provided vehicle and mission, the clans and chiefs model allows Steele-Flight to find additional options and represents a more thorough search of the design space. The GA portion of the hybrid method was run using 88 parallel processors to increase the computation speed.

After the GA ran the full 20 generations and taking a full 24hours to do so, it selected 5 chiefs representing the 5 best unique engine configurations. The results are seen in Table 4:

Rank	Phase 1 Engines	Phase 2 Engines	Phase 3 Engines	Phase 4 Engines	Phase 5 Engines	Payload (lbm)	GTOW (lbm)	Objective Function
1st place	5 Turbojet	5 Turbojet	5 Ramjet	5 Scramjet	1 SSME	50,968	672,720	-30.4903
2nd place	4 Turbofan	4 Turbofan	5 Ramjet	5 Scramjet	1 SSME	50,944	683,080	-30.3698
3rd place	4 Turbofan	5 Ducted Rocket	3 Ramjet	5 Scramjet	1 SSME	50,943	746,430	-29.7142
4th place	1 Ducted Rocket	4 Turbofan	4 Ramjet	5 Scramjet	1 SSME	50,943	764,660	-29.5577
5th place	5 Ducted Rocket	6 Turbojet	4 Ramjet	5 Scramjet	1 SSME	50,924	775,990	-29.4579

Table 4. The output of Steele-Flight in order of rank depending on the minimum objective function.

Just as with the Brute Force method, the top two engine configurations chosen by Steele-Flight are the configurations using pure Turbine engines, a single SSME engine in the second stage and all 5 DMSJ engines. These configurations lined up exactly with the outputs of the Brute force method which indicated that the top configurations were the configurations using 5 turbojets and the 4 turbofans.

The other 3 chiefs were all different mixes of the Turbine engines and the ducted rockets and were included in the study for the purpose of further exploration of the design space. What is interesting about these configurations is that, while each of them is close to the local optimum for their unique clan, none of these chiefs are actually at the local optimums as can be seen in Appendix C.4. This is, however, expected due to the nature of GAs. The top two clans naturally dominate the population in terms of reproduction in later generations. Since the ducted rocket configurations could not compete with the pure Turbine engine configurations, the later configurations began to dominate early on in the generations and thus reproduced the most. As a result, in the 20th and final generation, 13.6% of the population of engine combinations generated by the GA belonged to clan 1 and 65.9% belonged to clan2, while only 4.5% belonged to clan 3, 3.4% belonged to clan 4 and 1.1% belonged to clan 5. The remaining 11.3% of the population was composed of unique clans that did not make it into the top 5. Due to these proportions in the population, the genes of clans 3-5 were less likely to be involved in reproduction and thus less likely to improve. These results do not detract from the main objective of Steele-Flight which was to find the single optimum configuration given the defined mission, objective function and variable parameters, which it was able to accomplish.

Thus Steele-Flight was confirmed to be able to determine optimal engine configurations for given vehicle models and mission constraints by comparing its results with the results of the Brute Force method. Beyond this, Steele-Flight also identify the second best clan available from the population with its corresponding chief. However, given the behavior of GAs, the amount of optimums that can be found is dependent on the size of the population and on the number of generations allowed to run.

Chapter 7- Summary, Future Improvements and Conclusions

7.1 Summary of Results and Conclusions

In this thesis the method to find engine configurations that maximized payload for a conceptual TSTO RLV using trajectory optimization techniques was implemented through a code known as Steele-Flight. Steele-Flight is meant to be a conceptual level tool that can guide RLV designers in their decision making for engine configurations to be used on future RLV designs. While useful for basic trajectory optimization, Steele-Flight's primary purpose is to find the optimal choice of engine configuration and optimal trajectory with respect to a given objective function for a fixed vehicle design and mission when provided a pool of different engine models to select from.

The methods employed by Steele-Flight require the use of a Genetic Algorithm to simultaneously find an initial guess for the engine configuration and an initial guess for the trajectory which is later plugged into the NLP based optimization code known as GPOPS-II to perform a more accurate trajectory optimization with respect to the objective function than was done by the GA. After the initial guess is found, the two optimizers work together in tandem with the GA manipulating the engine variables and GPOPS-II optimizing the trajectory according to the currently selected engine configuration. The output of this process is then run through GPOPS-II once more, this time incorporating a mesh refinement algorithm to find a more accurate final trajectory to go with the final selected engine configuration.

Steele-Flight's ability as a trajectory optimizer was verified by its comparison to the trajectory optimizer known as ASTOS. When provided the same input data, mission constraints and optimizer constraints, Steele-Flight was able to find a comparable trajectory to that found by ASTOS, where the differences in the outputs varied only slightly.

Steele-Flight's method to find engine selections to maximize payload delivered to orbit was validated by comparison with a 'Brute force' technique that tested out nearly every configuration that was comparable to the Steele-Flight output. This technique was the only method that could verify the outputs of Steele-Flight beyond a doubt. However, while Steele-Flight's top two outputs were confirmed by the Brute force method, the 3rd, 4th and 5th place configurations varied slightly from their local optimums. This deviation from the local optimum in every clan is to be expected due to the basic nature of GAs and shows the need for larger populations if more than a single optimum is sought out.

7.2 Future Improvements

The amount of possible improvements to Steele-Flight are vast, thus only a few immediate issues will be addressed here.

7.2.1 –Parallel engine operation

As it currently stands, operation of different engine types occurs in sequence. Only one type of engine is allowed to be operating in each phase of the trajectory, thus transition between two different types of engine between phases must be assumed to occur instantaneously. This is an unrealistic assumption that in the future could be improved upon if Steele-Flight could model two different types of engines operating in parallel. Also, many current ELVs have multiple engines working together, such as when the Space

Shuttle uses its main engines in tandem with solid rocket boosters to achieve lift off. In its current state, Steele-Flight would not be able to model this. Thus, the ability to simulate two different types of engines would greatly expand the amount of scenarios that Steele-Flight can handle.

7.2.2 –Vertical Launches

Due to the dynamic flight model currently used by Steele-Flight, vertical launches cannot be modeled and thus only horizontal takeoff RLVs can currently be considered. The addition of another dynamic flight model that could tolerate vertical launches would greatly broaden the number of cases that Steele-Flight can simulate.

7.2.3 –Vehicle Design optimization (how to use excess volume)

While Steele-Flight currently focuses on finding the best engine configurations for a fixed vehicle design and size, the ability to increase or decrease the size of the vehicle would expand the amount of possible engine combinations to be explored. Also, being able to include multiple possible vehicle designs would further improve exploration of the design space.

7.2.4 –Oxygen Collection

Oxygen collection for RBCC engines is a concept that could greatly increase the effectiveness of RBCC engines and the overall payload capacity of an RLV. Future versions of Steele-Flight could greatly benefit from being able to model this currently researched technology.

References

1. Kurzke, J., *GasTurb11*. 2007.
2. Fischer, D., P. Gath, and K. Well, *ASTOS Model Library Version 7.2.0.* 2006-2014.
3. Dietrich, K., *Advanced two-stage launch vehicle concepts (SANGER)*, in *26th Joint Propulsion Conference*. 1990, American Institute of Aeronautics and Astronautics.
4. Sachs, G. and J. Drexler, *Optimal ascent of a Horus/Saenger type space vehicle*, in *Astrodynamics Conference*. 1988, American Institute of Aeronautics and Astronautics.
5. Eklund, D., *Quicksat: A Two Stage to Orbit Reusable Launch Vehicle Utilizing Air Breathing Propulsion for Responsive Space Access*, in *Space 2004 Conference and Exhibit*. 2004, American Institute of Aeronautics and Astronautics.
6. Weingertner, S., *SAENGER - The reference concept of the German Hypersonics Technology Program*, in *5th International Aerospace Planes and Hypersonics Technologies Conference*. 1993, American Institute of Aeronautics and Astronautics.
7. Olds, J.R. and J.E. Bradford, *SCCREAM: A Conceptual Rocket-Based Combined-Cycle Engine Performance Analysis Tool.* Journal of Propulsion and Power, 2001. **17**(2): p. 333-339.
8. Marc, B. and F. Milton, *Two-Stage-to-Orbit Reusable Launch Vehicle Propulsion Performance Study*, in *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. 2004, American Institute of Aeronautics and Astronautics.
9. Paul, G., M. Joseph, and P. Michael, *System Design of a Reusable, Horizontal Take-Off/Horizontal Landing Two Stage to Orbit Vehicle*, in *46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. 2010, American Institute of Aeronautics and Astronautics.
10. Ernst, H. and K. Dietrich, *Saenger - The German aerospace vehicle program*, in *National Aerospace Plane Conference*. 1989, American Institute of Aeronautics and Astronautics.
11. Escher, W.J.D. and E. Society of Automotive, *The synerjet engine : airbreathing/rocket combined-cycle propulsion for tomorrow's space transports : selected papers, 1963-1996*. 1997, Warrendale, Pa.: Society of Automotive Engineers.
12. Hendrick, P., et al., *Saenger-type T.S.T.O. using in-flight LOX collection*, in *33rd Joint Propulsion Conference and Exhibit*. 1997, American Institute of Aeronautics and Astronautics.
13. Joseph, H., F. Milton, and E. Dean, *TSTO Reusable Launch Vehicles Using Airbreathing Propulsion*, in *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. 2006, American Institute of Aeronautics and Astronautics.
14. Bayley, D.J., *Design Optimization of Space Launch Vehicles using a Genetic Algorithm*. 2007, Auburn University AL. p. 196.
15. GmbH, A.S., *ASTOS 7 Advanced Launcher Tutorial. Version 7.2.0.* 2014.
16. Weiland, C., ed. *Aerodynamic Data of Space Vehicles*. 2014, Springer-Verlag: Berline Heidelberg.
17. Patterson, M.A. and A.V. Rao, *GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming.* ACM Trans. Math. Softw., 2014. **41**(1): p. 1-37.

18. Bonnie J. McBride, S.G., *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications II. Users Manual and Program Description.* NASA Reference Publication, 1996(1311).
19. Kurzke, J., *GasTurb 11: Design and Off-Design Performance of Gas Turbines.* 2007: p. 50.
20. Bradford, J.E., *A technique for rapid prediction of aftbody nozzle performance for hypersonic launch vehicle design*. 2001, ProQuest, UMI Dissertations Publishing.
21. John, B. and O. John, *Improvements and enhancements to SCCREAM, a conceptual RBCC engine analysis tool*, in *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. 1998, American Institute of Aeronautics and Astronautics.
22. John, O., et al., *SCCREAM (Simulated Combined-Cycle Rocket Engine Analysis Module) - A conceptual RBCC engine design tool*, in *33rd Joint Propulsion Conference and Exhibit*. 1997, American Institute of Aeronautics and Astronautics.
23. Inc., S.E., *RedTop-Lite manual.* 2011.
24. Tewari, A., *Atmospheric and space flight dynamics : modeling and simulation with MATLAB and Simulink*. 2007, Boston: Birkhäuser.
25. Bryson, A.E. and Y.-C. Ho, *Applied optimal control : optimization, estimation, and control*. 1975, Washington; New York: Hemisphere Pub. Corp. ; Distributed by Halsted Press.
26. Betts, J.T., *Practical methods for optimal control using nonlinear programming*. 2001, Philadelphia, PA: Society for Industrial and Applied Mathematics.
27. Osyczka, A., *Evolutionary algorithms for single and multicriteria design optimization*. 2002, Heidelberg; New York: Physica-Verlag.
28. Shippey, B.M., *Trajectory optimization using collocation and evolutionary programming for constrained nonlinear dynamical systems*. 2008, ProQuest, UMI Dissertations Publishing.
29. *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization.* SIAM Review, 2005. **47**(1): p. 99-131.
30. Gill, P.E., W. Murray, and M.H. Wright, *Practical optimization*. 1981, London; New York: Academic Press.
31. Goldberg, D.E., *Genetic algorithms in search, optimization, and machine learning*. 1989, Reading, Mass.: Addison-Wesley Pub. Co.
32. Gath, P.F., *CAMTOS - a software suite combining direct and indirect trajectory optimization methods*. 2002. 172 S.
33. Fischer, D., P. Gath, and K. Well, *ASTOS 7 User Manual. Version 7.2.0.* 2006-2014.

Appendix A: Input variables and constraints for Engine Selection Validation test

This Appendix documents all the numerical quantities defining the Engine Selection Validation test.

Variables	Phase 1 bounds	Phase 2 bounds	Phase 3 bounds	Phase 4 bounds	Phase 5 bounds
AOA(°)	{0 6}	{0 6}	{2 6}	{2 6}	None
Pitch(°)	None	None	None	None	{-10 80}
Phase Time span (s)	{50 400}	{50 500}	{50 700}	{50 900}	{50 900}
Engine Type	Turbine Engines, Ducted Rocket	Turbine Engines, Ducted Rocket	Ramjet	Scramjet	Rocket
Engine setting (for initial guess only)	None	None	None	None	Max throttle; low throttle

Table 5. Variable bounds for all variables assigned by phase. The bounds on the Engine controls varied depending on which engine was selected and thus is not included here.

Variable	Stage 1	Stage 2
Stage Fuel (lbm)	{44,092.45 220,462.3}	{44,092.45 187,392.9}
Payload (lbm)	None	{8,818.49 70,547.92}
Amount of turbines on Stage	Jets {5 6 8} Fans{4 5 6}	None
Amount of ramjets on Stage	{4 5}	None
Amount of scramjets on Stage	{4 5}	None
Amount of rockets on Stage	{2 5}	{1 2}
Choice of turbine model	Turbojet; Turbofan	None
Choice of rocket model	Ducted Rocket	SSME; J-2X

Table 6. Variable bounds of all variables assigned by Stage.

Appendix B: Engine models for Engine Selection Validation test

Engine Model	BPR	Burner Temp (R)	LPC PR	HPC PR	Afterburner Temp (R)	Weight (lbs)	Radius (ft)	Thrust (lbf)	Isp (s)
Turbofan	.15	3330	4	7	4320	13,500	6.819	127,881	4739
Turbojet	0	3330	4	7	4320	9601	4.494	89,974	4578

Table 7. Design point data for the turbine engines.

Engine Model	Fuel	Mixture ratio	Weight (lbs)	Diameter (in)	Thrust (klb)	Isp (s)
SSME	LOX/LH2	6:1	7,775	96	512	452
J-2X	LOX/LH2	5.5:1	5,450	120	294	448

Table 8. Vacuum data for pure rocket engines.

Appendix C: Results of Engine Selection Validation test

C.1 –Results of Turbine Engine tests using J-2X and SSME Rocket engines

This section details the comparison of the J-2X rocket engine and the SSME rocket engine being used on the second stage of the Saenger RLV, while the first stage is powered by turbine engines and 5 DMSJ engines.

Turbines	Rockets	GTOW (lbm)	2nd stage GTOW (lbm)	Payload (lbm)
5 Turbojets	1 SSME	672,720	296,200	50,968
4 Turbofans	1 SSME	683,080	296,180	50,944
5 Turbojets	1 J-2X	645,660	275,730	42,160
4 Turbofans	1 J-2X	650,250	275,750	42,182

Table 9. Data comparing performance of J-2X rocket engine and SSME rocket engine.

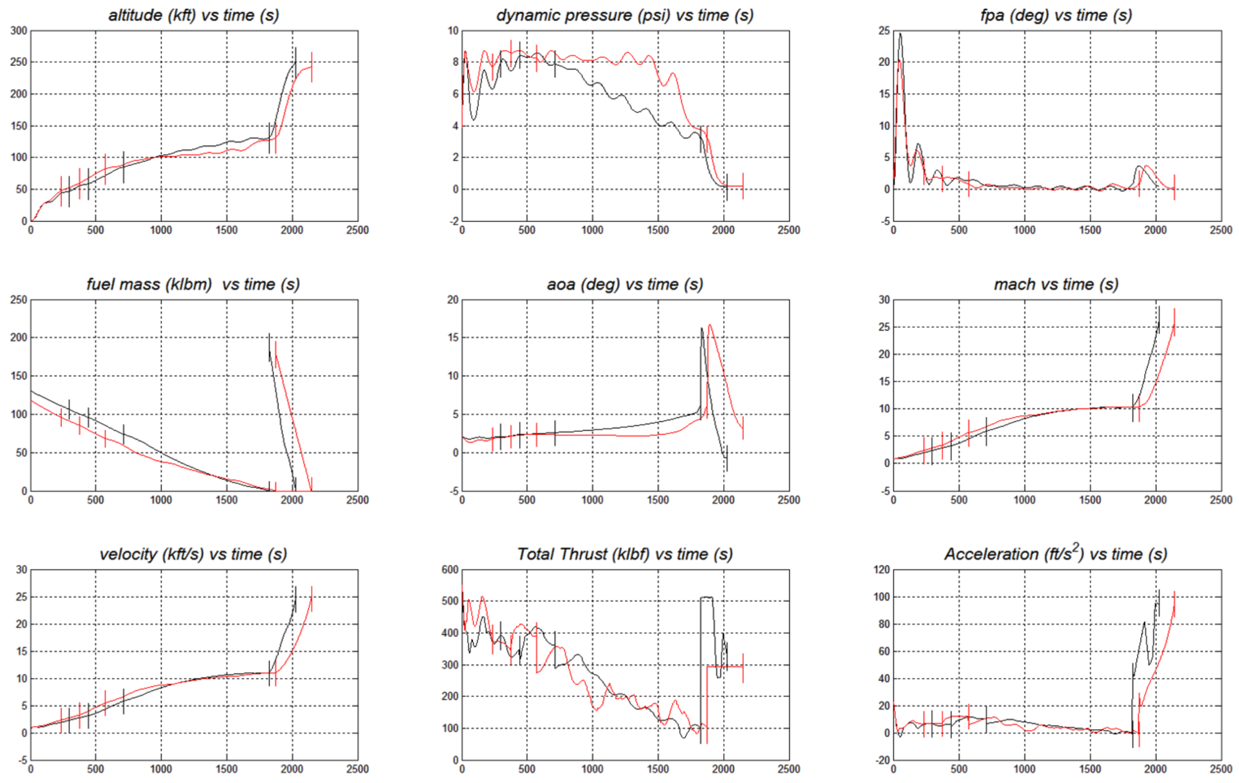

Figure 21. Chart of results using 4 turbofans. Black lines represent the run using the SSME engine. The red lines represent the run using the J-2X engines. The vertical lines are the phase boundaries. It can be seen that the dynamic pressure stays within the boundary constraints of {2.9 8.7} psi imposed on phases 1-4. The acceleration stays within the imposed boundary of 98.42ft/s^2. The vehicle achieves a final altitude above the required 242,782.2ft.

C.2 –Results of varying the number of Turbine engines

This section addresses the question as to why decreasing amounts of turbine engines in the first two phases causes an increase in payload? To examine this, only the run of 6, 5 and 4 turbofans are compared. Each is run with all 5 DMSJ engines active and 1 SSME engine.

Active Turbine Engines	Final Time (s)	First stage engine mass (lbm)	First stage fuel mass (lbm)	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)
6 Turbofans	2082.0	105,830	134,030	712,820	296,040	50,800
5 Turbofans	2051.7	92,329	130,830	696,200	296,110	50,875
4 Turbofans	2028.8	78,827	131,150	683,080	296,180	50,944

Table 10. Comparing the effects of varying the number of turbine engines.

In a test of 6 turbofans versus 4 turbofans, the 6 turbofans naturally had more thrust than the 4 turbofans in the first two phases and thus finished the first two phases faster because they reached the maximum on turbofan performance faster.

The 4 turbofans were active longer and were able to achieve the same final conditions as that achieved by the 6 turbofans. Without the extra weight of 2 turbofans (around 27,001lbs!) being carried by the first stage, the 4 turbofans were able to achieve a lower GTOW, thus allowing more payload to be carried to orbit due to reduced engine mass. However, even though a lower GTOW was used to get to the maximum staging condition (131.2kft and Mach 10.3), the payload increase was relatively small (only 99.2lbs). This is due to the fact that the maximum upper limit on payload is entirely dependent on the Thrust and Isp of the second stage rocket.

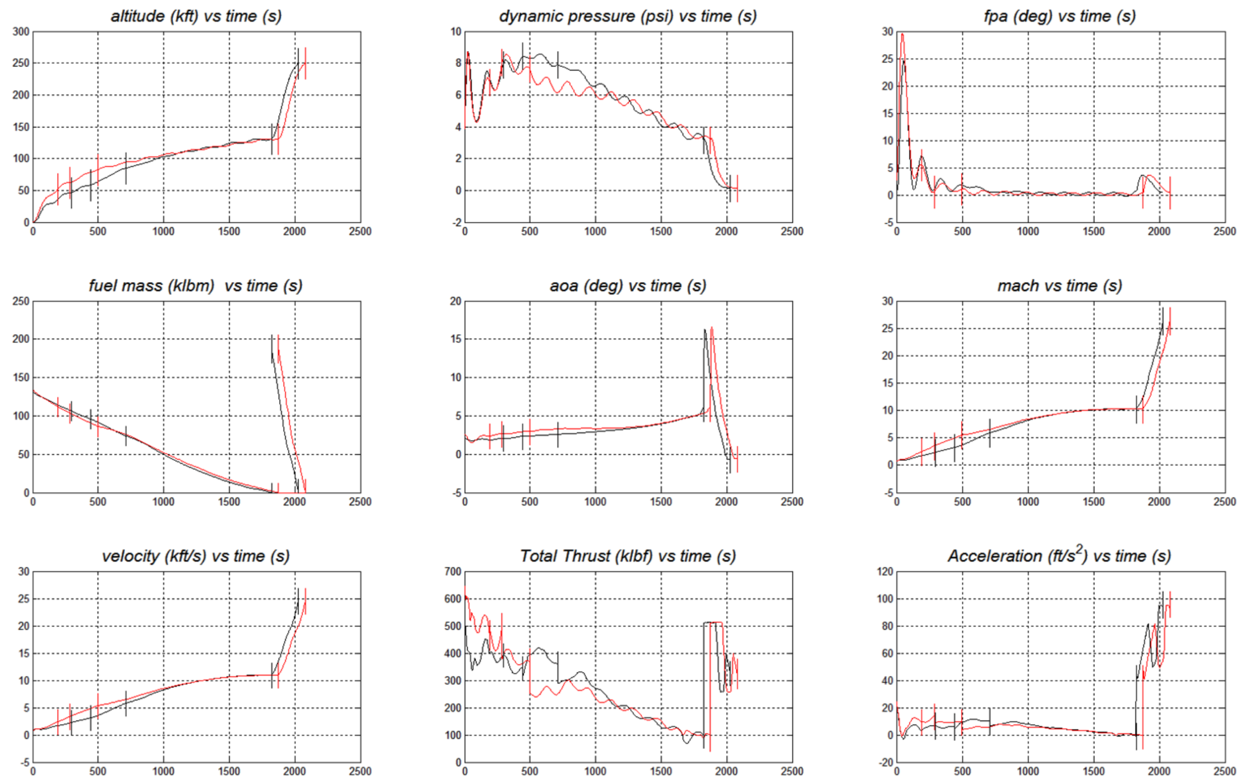

Figure 22. Chart of results using 1 SSME. Black lines represent the run using 4 turbofans. The red lines represent the run using 6 turbofans. It can be seen that the dynamic pressure stays within the boundary constraints of {2.9 8.7} psi imposed on phases 1-4. The acceleration stays within the imposed boundary of 98.42ft/s^2. The vehicle achieves a final altitude above the required 242,782.2ft.

C.3 –Results of varying the number of ramjet engines

This appendix addresses the question that arose in tests that varied the number of ramjets and the number of scramjets. The question is, why did decreasing the number of ramjets, while maintaining the same number of scramjets have little effect on payload?

Active Turbine Engines	Active Ramjets	Active Scramjets	Active 2nd Stage Rockets	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)
5 Turbojets	5 Ramjets	5 Scramjets	1 SSME	672,720	296,200	50,968
5 Turbojets	4 Ramjets	5 Scramjets	1 SSME	675,480	296,210	50,970

Table 11. Comparison to find effects of varying the number of ramjets.

From the table above it can be noticed that decreasing the number of ramjets to 4, while keeping the same number of scramjets results in roughly the same payload as with 5 ramjets. The GTOW, however increases by 2,760lbs. It appears that the time span on the 4 ramjet phase is increased and the 4 ramjets appear to make it to a slightly lower altitude than the 5 ramjets did. Also, the 5 ramjets have a slightly higher acceleration while simultaneously increasing altitude. Thus the 5 ramjets simply take a shorter, more energy intensive path to endpoints that have a slightly higher altitude. This path appears to be the more efficient of the two because the 4 ramjets require more fuel for their longer run. In the end, this additional fuel ends up driving the GTOW up higher than when the 5 ramjets are used. This is because the ramjets and scramjets are the same engines running at different modes, thus decreasing to 4 ramjets, while still having 5 scramjets means that there are still 5 of these engines, only 1 is turned off during ramjet mode. Therefore, there is no reduction in engine mass.

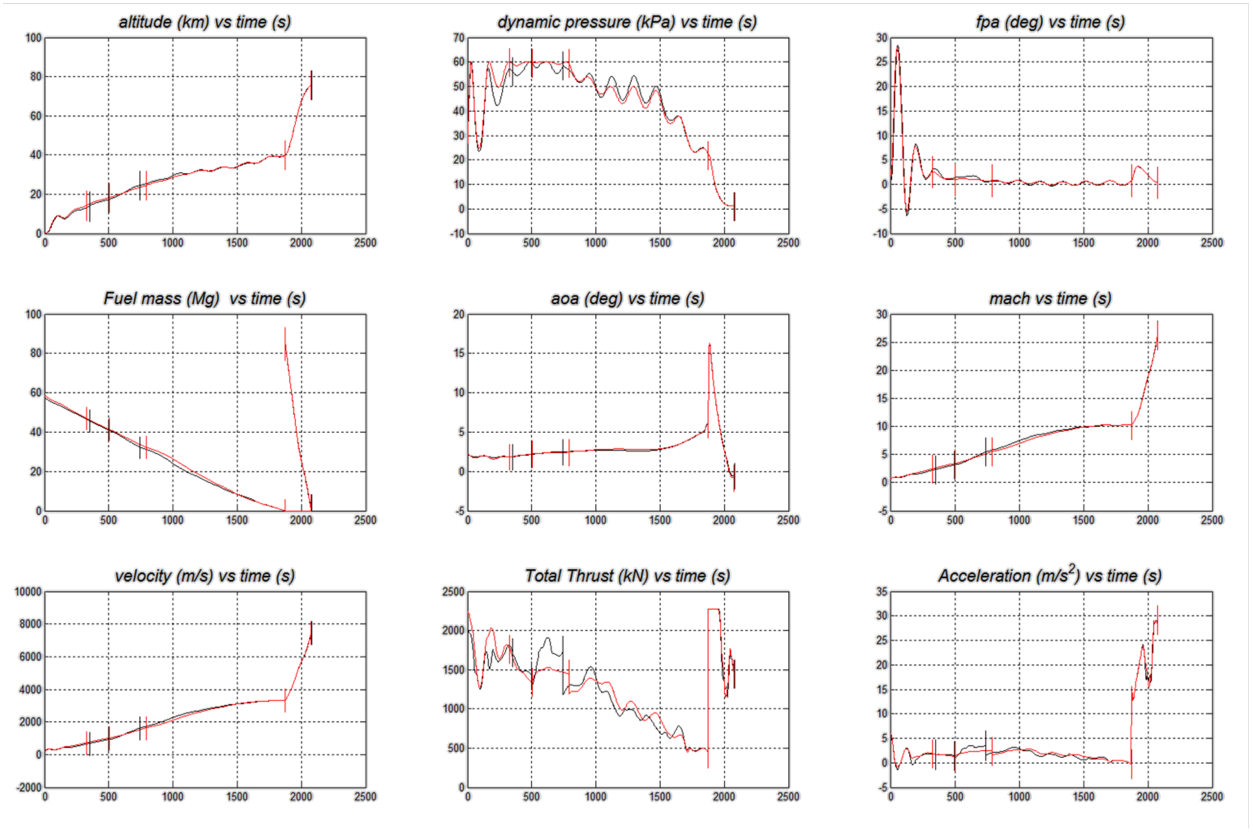

Figure 23. Chart of results using 5 turbojets and 1 SSME. Black lines represent the run using 5 ramjets. The red lines represent the run using 4 ramjets. It can be seen that the dynamic pressure stays within the boundary constraints of {20 60} kPa imposed on phases 1-4. The acceleration stays within the imposed boundary of 30m/s^2. The vehicle achieves a final altitude just above the required 74km.

C.4 –Data for Ducted Rocket tests

All the configurations using ducted rockets used 5 scramjets and 1 SSME rocket, since the previous tests already established that these are always required for the best output. The results of the objective function based on Equations 14 and 15 is provided and the configurations selected by Steele-Flight are highlighted in yellow. It should be noted that the values plugged into the Objective function were in metric units.

Phase 1 Engine	Phase 2 Engine	Phase 3 Engine	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)	Objective
4 Turbofans	5 Ducted Rockets	3 Ramjets	746,430	296,180	50,943	-29.7142
4 Turbofans	5 Ducted Rockets	4 Ramjets	727,880	296,170	50,940	-29.9007
4 Turbofans	4 Ducted Rockets	3 Ramjets	753,930	296,180	50,946	-29.6573
4 Turbofans	5 Ducted Rockets	5 Ramjets	721,080	296,180	50,943	-29.969
5 Turbofans	4 Ducted Rockets	4 Ramjets	760,250	296,100	50,869	-29.5577

Table 12. Outputs of configurations similar to the 3rd place chief

Phase 1 Engine	Phase 2 Engine	Phase 3 Engine	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)	Objective
1 Ducted Rocket	4 Turbofans	4 Ramjets	764,660	296,180	50,943	-29.5577
2 Ducted Rockets	4 Turbofans	4 Ramjets	771,660	296,170	50,939	-29.5048
2 Ducted Rockets	5 Turbofans	4 Ramjets	758,770	296,110	50,871	-29.5747
2 Ducted Rockets	5 Turbofans	5 Ramjets	754,920	296,110	50,871	-29.6116
1 Ducted Rockets	4 Turbofans	5 Ramjets	766,500	296,170	50,936	-29.5359
1 Ducted Rocket	4 Turbofans	3 Ramjets	767,600	296,180	50,944	-29.5090
5 Ducted Rocket	6 Turbofans	4 Ramjets	791,890	296,040	50,805	-29.2536

Table 13. Outputs of configurations similar to the 4th place chief

Phase 1 Engine	Phase 2 Engine	Phase 3 Engine	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	Payload (lbm)	Objective
4 Ducted Rockets	5 Turbojets	4 Ramjets	766,410	296,210	50,970	-29.5595
5 Ducted Rockets	5 Turbojets	4 Ramjets	766,430	296,210	50,977	-29.5655
5 Ducted Rockets	6 Turbojets	4 Ramjets	775,990	296,160	50,924	-29.4579
4 Ducted Rockets	6 Turbojets	4 Ramjets	775,980	296,160	50,921	-29.452
5 Ducted Rockets	6 Turbojets	5 Ramjets	775,220	296,160	50,927	-22.4481

Table 14. Output of configurations similar to the 5th place chief

C.5 –Complete Engine validation Data table

All data obtained from the brute force method with the Objective function based on Equations 14 and 15. It should be noted that the values plugged into the Objective function were in metric units. The configurations chosen by Steele-Flight are highlighted in yellow.

Phase 1 Engine	Phase 2 Engine	Phase 3 Engine	Phase 4 Engine	Phase 5 Engine	Total GTOW (lbm)	GTOW 2nd Stage (lbm)	PL (lbm)	Objective
5 TJ	5 TJ	5 RJ	5 SJ	1 SSME	672,720	296,200	50,968	-30.4903
6 TJ	6 TJ	5 RJ	5 SJ	1 SSME	679,540	296,160	50,924	-30.3779
8 TJ	8 TJ	5 RJ	5 SJ	1 SSME	704,010	296,020	50,789	-29.9812
5 TJ	5 TJ	4 RJ	5 SJ	1 SSME	675,480	296,210	50,970	-30.452
5 TJ	5 TJ	4 RJ	4 SJ	1 SSME	681,190	295,590	50,354	-29.9510
5 TJ	5 TJ	5 RJ	4 SJ	1 SSME	684,660	295,620	50,389	-29.9681
5 TJ	5 TJ	5 RJ	5 SJ	1 J-2X	645,660	275,730	42,160	-25.4821
6 TJ	6 TJ	5 RJ	5 SJ	1 J-2X	651,890	275,730	42,166	-25.3866
8 TJ	8 TJ	5 RJ	5 SJ	1 J-2X	675,850	275,650	42,081	-25.0727
4 TF	4 TF	5 RJ	5 SJ	1 SSME	683,080	296,180	50,944	-30.3698
5 TF	5 TF	5 RJ	5 SJ	1 SSME	696,200	296,110	50,875	-30.1737
6 TF	6 TF	5 RJ	5 SJ	1 SSME	712,820	296,040	50,800	-29.9509
4 TF	4 TF	4 RJ	5 SJ	1 SSME	682,530	296,170	50,938	-30.3557
4 TF	4 TF	4 RJ	4 SJ	1 SSME	688,510	295,580	50,349	-29.8964
4 TF	4 TF	5 RJ	4 SJ	1 SSME	692,700	295,580	50,341	-29.8652
4 TF	4 TF	5 RJ	5 SJ	1 J-2X	650,250	275,750	42,182	-25.4131

5 TF	5 TF	5 RJ	5 SJ	1 J-2X	666,260	275,690	42,120	-25.2285
6 TF	6 TF	5 RJ	5 SJ	1 J-2X	681,580	275,620	42,055	-25.0483
4 TF	5 DR	3 RJ	5 SJ	1 SSME	746,430	296,180	50,943	-29.7142
4 TF	5 DR	4 RJ	5 SJ	1 SSME	727,880	296,170	50,940	-29.9007
4 TF	5 DR	5 RJ	5 SJ	1 SSME	721,080	296,180	50,943	-29.969
4 TF	4 DR	3 RJ	5 SJ	1 SSME	753,930	296,180	50,946	-29.6573
5 TF	4 DR	4 RJ	5 SJ	1 SSME	760,250	296,100	50,869	-29.5577
1 DR	4 TF	4 RJ	5 SJ	1 SSME	764,660	296,180	50,943	-29.5577
2 DR	4 TF	4 RJ	5 SJ	1 SSME	771,660	296,170	50,939	-29.5048
2 DR	5 TF	4 RJ	5 SJ	1 SSME	758,770	296,110	50,871	-29.5747
2 DR	5 TF	5 RJ	5 SJ	1 SSME	754,920	296,110	50,871	-29.6116
1 DR	4 TF	5 RJ	5 SJ	1 SSME	766,500	296,170	50,936	-29.5359
1 DR	4 TF	3 RJ	5 SJ	1 SSME	767,600	296,180	50,944	-29.5090
5 DR	6 TF	4 RJ	5 SJ	1 SSME	791,890	296,040	50,805	-29.2536
5 DR	6 TJ	4 RJ	5 SJ	1 SSME	775,990	296,160	50,924	-29.4579
4 DR	5 TJ	4 RJ	5 SJ	1 SSME	766,410	296,210	50,970	-29.5595
5 DR	5 TJ	4 RJ	5 SJ	1 SSME	766,430	296,210	50,977	-29.5655
4 DR	6 TJ	4 RJ	5 SJ	1 SSME	775,980	296,160	50,921	-29.452

Table 15. Outputs of all tests conducted with the Brute Force method.