Combined Optimization of Trajectory and Design for Dual Propulsion Spacecraft

AE5810 Thesis Space

Domas M. Syaifoel

Delft University of Technology

TU Delft

Combined Optimization of Trajectory and Design for Dual Propulsion Spacecraft

AE5810 Thesis Space

by

Domas M. Syaifoel

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 14 December 2020.

Student number:	4436024	
Project duration:	November 2019 – December 2020	
Thesis committee:	Dr. A. Cervone,	TU Delft, supervisor
	Dr. J. Guo	TU Delft, committee chair
	ir. R. Noomen	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Following the trend of miniaturization and standardization of satellite design, as well as recent successes in the use of cubesats in interplanetary missions, a cubesat design capable of reaching another planet fully independently may lead to significant cost reductions for future missions. While efficient low thrust propulsion systems exist, Earth escape using low thrust only leads to significant incident radiation doses when crossing the Van Allen belts. As such, this report aims to present the design of a dual thrust cubesat, i.e. one which employs both high thrust and low thrust propulsion systems, such that the transfer time and the number of van Allen belt crossings is within requirements, while a final orbit around Mars remains attainable.

In doing so, this report explores the theory behind low thrust trajectory optimization, and attempts to combine this with a general optimization scheme including both high thrust phases as well as the optimization of the spacecraft system design. Implementation of such a scheme has not been shown to be useful: issues in finding a good initial guess prevent convergence in many cases, and the proposed plan to alleviate the encountered issues requires an iterative approach between system design and trajectory design. As such, the proposed scheme has not been found to have any benefit over using existing trajectory design tools in an iterative way.

A design approach is presented using the `pykep` trajectory optimizer developed at the European Space Agency (ESA). Iteration between this tool, high thrust calculations, and system design budget calculations allows for optimizing towards a feasible design that meets the requirements, starting from an arbitrary initial guess. When a list of commercial off the shelf (COTS) components is available following the CubeSat standard, it is possible to quickly generate a feasible design without significant prior work.

This report presents a full COTS design for a cubesat capable of reaching Mars independently from a common piggyback launch orbit. The design in question is a 12 unit cubesat with a dry mass of 11.4 kg and a wet mass of 14.8 kg. Launched into a geostationary transfer orbit, it can reach escape velocity though 9 Van Allen belt passes, and reach Mars in less than 5 years.

Due to the constrained scope of the design, further work is needed to verify that the momentum dumping budget, link budget, and thermal budget are indeed sufficient. It is further recommended to improve on the work in this thesis, by finding an alternative integrated optimization scheme, or by including and automating more features within `pykep`. Lastly, recommendations are given for integration, testing, and launch of the proposed spacecraft design.

Contents

Nomenclature

Abbreviations

ADC	Attitude determination and control subsystem
AU	Astronomical unit
BVP	Boundary value problem
CAD	Computer aided design
COM	Communication subsystem
COTS	Commercial off the shelf
CPT	Chemical thruster propellant subsystem
CTH	Chemical thruster subsystem
CTR	Structural subsystem
EPR	Electric thruster propellant subsystem
EPS	Electric power subsystem
ESA	European Space Agency
ETH	Electric thruster subsystem
GTO	Geostationary transfer orbit
I2C	Interintegrated circuit
LEO	Low Earth orbit
MBH	Monotonic basin hopping
MPPT	Maximum power point tracker
NASA	National Aeronautics and Space Administration
OBC	On board computer subsystem
OCP	Optimal control problem
ODE	Ordinary differential equation
PCA	Patched conic approximation
PMP	Pontryagin's maximum principle
RAD	Radiation shielding subsystem
SOI	Sphere of influence
SOL	Solar array subsystem
SQP	Sequential quadratic programming
SSO	Sun synchronous orbit

THR	Thermal control subsystem
TID	Total ionizing dose
TRL	Technology readiness level

Latin Symbols

a	Semi major axis
C	Cost function
c	Charged capacity
e	Eccentricity
F	Thrust
f	Arbitrary function
f	Slow equinoctial element
g	Slow equinoctial element
g_0	Standard surface gravitational acceleration
H	Hamiltonian
h	Slow equinoctial element
h	Throttle control setting
i	Inclination
I_{sp}	Specific impulse
k	Slow equinoctial element
k	Switching function
L	Fast equinoctial element
M	Mean anomaly
m	Mass
n	Numerator
p	Power
p	Slow equinoctial element
r	Radius
t	Time
u	Velocity component
V	Speed
v	Velocity component
w	Velocity component
x	Position component
y	Position component

z Position component

$\boldsymbol{a}$ Acceleration vector

$\boldsymbol{q}$ State vector

$\boldsymbol{r}$ Position vector

$\boldsymbol{s}$ Velocity vector

Greek Symbols

α Pitch angle

β Yaw angle

Δ Increment

λ Costate

μ Standard gravitational parameter

ν True anomaly

Ω Longitude of ascending node

ω Argument of periapsis

ϕ Position angle

π Ratio between circle circumference and diameter

θ Position angle

$\boldsymbol{\alpha}$ Dynamic control vector

$\boldsymbol{\lambda}$ Costate vector

Other

$(\dot{})$ First derivative with respect to time

$(^T)$ Transposed

$(_c)$ Chemical

$(_e)$ Electric

$(_f)$ Final

$(_g)$ Gravitational

$(_i)$ Initial

$(_j)$ Iterated

$(_n)$ Normal

$(_p)$ Payload

$(_r)$ Radial

$(_s)$ Solar

$(_s)$	Structural
$(_t)$	Tangential
∇	Gradient
∂	Partial derivative

List of Figures

List of Tables

1

Introduction

In 2008, Delfi-C^3 was launched, the first nanosatellite developed in the Netherlands, as designed and operated by the Delft University of Technology. As a nanosat, the spacecraft has a considerably small size: it consists of 3 units of the CubeSat form factor, which are cubic blocks of 10 by 10 by 10 cm.

The CubeSat standard is one of the steps in the ongoing miniaturization and standardization of spacecraft and spacecraft components. This comes with significant savings in costs: the Delfi line of cubesats shows it is feasible for an academic institution to design, build, and arrange the launch of a satellite. These savings are due to several factors. The size of the completed spacecraft allows for inexpensive launches, as several similar sized spacecraft are launched at once, sharing the expense. Standardization of the form factor allows for low expenses for integration in the launch vehicle, and deployment from the vehicle. Standardization of the components allows for spacecraft designers to select and combine existing components, rather than to design all components from scratch. These components are commercially available, have been individually tested to commercial standards, and are ready for integration immediately after acquisition. This is usually designated as commercial off the shelf (COTS) components. The use of COTS components for cubesat missions further reduces design, integration, and test costs.

The trend of miniaturization continues. After Delfi-C^3, the TU Delft developed and launched Delfi-n3Xt, and is currently developing Delfi-PQ satellite according to the PocketQube standard: cubes of 5 cm length. In the meantime, the study society at the Aerospace Engineering faculty of the TU Delft, VSV 'Leonardo da Vinci', is working on its own cubesat, dubbed Da Vinci, in cooperation with the suppliers of cubesat components.

With spacecraft becoming smaller and more affordable, the space flight industry is looking at miniature satellites for more prestigious missions. In 2018, NASA demonstrated the first use of cubesats beyond Earth orbit with the Mars Cube One mission. These cubesats, however, relied on the propulsion system of the launch vehicle to reach Mars; the cubesat's internal propulsion systems were only used for trajectory corrections and attitude control.

In this report, it is investigated whether it is feasible to design a miniature satellite in the same way as the Delfi line, as an academic demonstration, using standardized components, and launched into an Earth orbit; which can reach Mars by itself.

While other interplanetary targets could be considered with a lower ΔV requirements, such as the Moon or certain asteroids, Mars is considered as the target of this study. As reasons for this, it must be noted that Mars is a significant target for scientific research, including the aforementioned NASA mission. Furthermore, a mission to Mars would be considered to be a prestigious achievement in the public eye, especially when compared to current cubesat missions. Lastly, Mars is the target with the lowest ΔV requirement that can be called an interplanetary mission, contrasted with other bodies in the solar system.

This leads to the main research question of this thesis:

What is a feasible design for a standalone cubesat mission to Mars?

The design of interplanetary cubesat missions is currently subject to research. However, no literature has been found to present a cubesat design that extensively uses COTS components. As such, the goal of this report is to investigate the feasibility of COTS designs for interplanetary missions, with a significant reduction in expected cost.

Instead, literature has been found that emphasizes the use of 'dual thrust' for such a mission, such as Mani et al. [36], with dual thrust referring the combination of high thrust and low thrust propulsion systems. The reason for this is intuitive. The use of low thrust propulsion has led to significant savings in propellant mass, which is especially important for miniature spacecraft. These solutions are typically electric propulsion systems which accelerate propellant to very high velocities, reducing the required propellant mass to reach the same impulse.

However, the use of low thrust has several drawbacks. The difference in thrust leads to a longer transfer time. This may lead to an unacceptable mission duration, or increase the requirements on radiation tolerance, due to prolonged exposure. This is especially the case when considering Earth escape using low thrust: each time the spacecraft crosses the Van Allen radiation belt, it receives a radiation dose orders of magnitude larger than during the rest of the transfer. As such, Mani et al. [36] suggest to use high thrust components to reduce the time spent in highly radiative zones.

In order to do this, it is necessary to optimize the trajectory and systems design of the spacecraft together, in order to reach a feasible design. This leads to the first research subquestion.

> *What is a useful framework for optimizing in tandem the trajectory and systems design of a dual thrust spacecraft?*

When such a theoretical framework is found, there are a number of options to develop a practical implementation. While these options will be further discussed in chapter 3, this subquestion is listed here as well.

> *Which implementation is useful for combined optimization of a dual thrust system?*

Lastly, optimizing high thrust trajectories is computationally simple [26], but optimization of low thrust trajectories is mathematically involved, and currently a popular research subject. As such, it is expected that the optimization of the low thrust part of the trajectory will require the most effort, both in programming effort and computational effort. In order to address this, a major part of the literature study was dedicated to researching methods for optimizing low thrust trajectories. This leads to the third research subquestion.

> *What is a useful framework for optimizing low thrust trajectories?*

As such, this report aims to answer these questions in reverse order. First, it is explored how low thrust trajectories can be optimized. Then, it is attempted to combine low thrust optimization with high thrust phases and systems design into a single framework. Lastly, this framework is used to design the mission to Mars.

This report is organized as follows. First, chapter 2 summarizes the literature study, in order to gain an understanding of low thrust trajectory optimization, dual thrust trajectories, and combined optimization of trajectories and systems. Then, chapter 3 defines the design problem, including the objectives of the mission and specific requirements, and the intended design process as conceived from the literature study. Then, chapter 4 and chapter 5 describe the theoretical and practical aspects of this design process. The resultant design is presented in chapter 6, after which the report concludes with a discussion in chapter 7.

2

Literature Study

This chapter summarizes the literature study, submitted previously as Syaifoel [61], using the same layout as the literature study itself. As such follows a discussion on the motivation of this research, on considerations regarding systems and trajectory design, on the use of dual propulsion, and on practical aspects regarding numerical optimization and verification & validation.

2.1. Motivation

The motivation of this study derives from miniaturization and standardization of spacecraft components, with many COTS products following the CubeSat standard [32]. Due to the limits in mass and size, the use of low thrust propulsion systems is necessary for cubesats to reach similar distances to larger spacecraft.

This includes the possibility of interplanetary missions using cubesats, although this is expected to require a high degree of optimization. Specifically, this alludes to optimizing both the systems design and the spacecraft trajectory: this way, the trajectory is not optimized for a single set design, nor is the design optimized for a single trajectory; but this is done in tandem. This is termed coupled optimization [36].

Furthermore, the low thrust performance of these systems leads to long transfer times, which may by itself lead to unacceptable costs, as well as considerable radiative doses throughout the transfer and especially during Van Allen belt crossings. As such, it is recommended to combine low thrust systems with high thrust systems to trade off the advantages and disadvantages of both: this is termed dual thrust [37].

Overall, the optimization problem involves both the high thrust trajectory, low thrust trajectory, and the system design itself. Of these aspects, optimization of high thrust trajectories is considered to be simple, but low thrust trajectory is conceptually complex and computationally expensive [26]. As will be discussed further, optimization of a low thrust trajectory requires either significant computational resources, and depending on the approach, a more advanced knowledge of dynamics than high thrust trajectory calculations. Because of this, the design of a low thrust trajectory has been found to not always converge, depending on the initial guess. This hinders the implementation of an automated optimization scheme. Due to the relative difficulty of low thrust optimization, a major part of the literature study is dedicated to this problem.

2.2. Systems Design

Systems design is considered a major part of the BSc and Space Flight MSc track at the Aerospace Faculty of the Delft University of Technology. As such, it is not fully explained here.

However, it is necessary to look at the aspects which lend itself to optimization. For example, Leonard [33] describes systems engineering as an iterative process, in which the design is continually adapted to the requirements in an iteration loop. This approach is not always used. Spangelo et al. [58] perform a coupled optimization of a low thrust trajectory and the systems design, but do so in a waterfall approach: first the trajectory is optimized, then the spacecraft is optimized based on the resulting trajectory.

In order to optimize the systems design, it is necessary to define a parameter to optimize for. Often, this is considered as overall mission cost, which at an early stage in the design process is based on estimates, as seen in Mosher et al. [42], which incidentally stressed the need for COTS components to reduce design, integration, and test costs. To make this estimation accurate, it is necessary to rely on a large data set, as seen in approaches by Bearden [4], Aas et al. [2], and Aas et al. [1]. However, as the cost estimations in this thesis

are not expected to be as accurate as necessary, it is instead possible to take another metric which correlates well with overall mission cost, such as system mass.

With the objective well defined, it becomes possible to numerically solve a system design problem using an evolutionary approach, as shown in Mosher [40] and Mosher [41].

Coupled optimization is seen in literature as an extension of numerical system design. The specific approach differ, as expected, based on the method of optimizing the low thrust trajectory. Petukhov et al. [47] use the indirect method, based on Pontryagin's maximum principle (PMP) to calculate the trajectory, but optimize the design at a very high level by include some design variables as static parameters. Spangelo et al. [58] and Spangelo and Longmier [57] use instead Edelbaum's approximation , a less complex approach, which is only valid for circular orbits. Lastly, [14] uses an evolutionary neural network approach to optimize a solar sail spacecraft with each trajectory. These approaches are discussed further below.

2.3. Trajectory Design

In all found literature, trajectory design is based on Keplerian dynamics, with a single gravitational mass and with thrust as the sole disturbing force. As such, aerodynamic, solar, and tidal disturbances are neglected. Furthermore, all undisturbed orbits are Keplerian orbits in which only a single state variable, an anomaly, changes over time. For interplanetary missions, the patched conic approximation (PCA) is used, in which only the planet's gravity is considered until the spacecraft leaves the planet's sphere of influence (SOI), after which only the Sun's gravity is considered.

Like system design optimization, trajectory optimization requires a well defined objective. Often, ΔV is minimized, equivalent to minimizing propellant mass; alternatively, travel time is minimized.

Trajectory optimization is equivalent to solving and optimizing a boundary value problem (BVP), which is an ordinary differential equation (ODE) over time with constraints on both end points. Depending on the mission type, the number of constraints at each boundary differs: a rendezvous problem has more constraints than an orbit to orbit problem [48].

For high thrust trajectories, optimization is considered "mathematically relatively straightforward" [26]. Specifically, it can be considered as an instance of Lambert's problem, which can be solved analytically using for example Kriz [31].

2.3.1. Edelbaum's Approximation

For low thrust trajectories between two circular orbits, which is a common use case for Earth orbiting spacecraft, Edelbaum's approximation holds. This entails that thrust is at maximum throttle at all time, and the thrust component in radial direction is zero. This leaves one control variable, the yaw angle β.

Kechichian's algorithm gives an analytic solution to this problem, which is presented in Kechichian [23], Kechichian [24], and Kluever [28]. Casalino and Colasurdo [9] expand Edelbaum's approximation for variable mass, specific impulse, and multiple revolutions. Kechichian's algorithm is summarized as follows.

1. Given gravitational parameter μ, and initial $(_i)$ and final $(_f)$ semi major axes a, calculate circular speeds V.
$$V = \sqrt{\mu/a} \tag{2.1}$$

2. Given the total inclination change Δi, calculate initial yaw angle β_i.
$$\tan\beta_i = \frac{\sin(\Delta i\pi/2)}{V_i/V_f - \cos(\Delta i\pi/2)} \tag{2.2}$$

3. Find the total ΔV.
$$\Delta V = V_i\cos\beta_i - \frac{V_i\sin\beta_i}{\tan\left(\Delta i\pi/2+\beta_i\right)} \tag{2.3}$$

4. The transfer time t_f depends solely on the thrust F.
$$t_f = \Delta V/F \tag{2.4}$$

5. At any time t, find the instantaneous optimal yaw angle β, velocity V and inclination i.
$$\beta(t) = \arctan\frac{V_i\sin\beta_i}{V_i\cos\beta_i - Ft} \tag{2.5}$$

$$V(t) = \sqrt{V_i^2 - 2V_i F t \cos\beta_i + F^2 t^2} \tag{2.6}$$

$$i(t) = i_i + \frac{2}{\pi} \arctan \frac{Ft - V_i \cos\beta_i}{V_i \sin\beta_i} + 1 - \frac{2\beta_i}{\pi} \tag{2.7}$$

For optimizing noncircular low thrust trajectories, two general approaches exist: the direct and indirect method. In the direct method, the trajectory is discretized first, and then each discrete step is optimized with continuity constraints. In the direct method, the trajectory is optimized as a whole. The direct method is less accurate and less conceptually complex than the indirect method, with accuracy dependent on the step size. To summarize, the direct method discretizes, then optimizes; while the indirect method optimizes, then discretizes.

2.3.2. Direct Method

The direct method is presented in Sims and Flanagan [54] and Sims et al. [55]. Here, thrust in the trajectory is at each step modeled as a single impulse; in other words, the low thrust problem is tackled as many high thrust problems. Overall, the trajectory can therefore directly be solved as a nonlinear optimization problem, constrained at both endpoints and constrained to be continuous between each step.

The direct method is executed as follows.

- Determine an initial and final state $\boldsymbol{q}_0$ and $\boldsymbol{q}_n$ given a number of time steps n.
- For each time step, generate an initial guess for the state vector at the midpoint of this time step, and propagate from this state backwards to the beginning of the time step.
- For each time step, generate an initial guess for the dynamic control vector, $\boldsymbol{\alpha}$. This vector determines the impulse at the midpoint of the timestep, which changes the state vector after the impulse. Integrate from this new state vector to the end of the time step.
- This leads to a constrained optimization problem. The constraints are such that discrepancies between time steps are within a certain tolerance. The problem is optimized for minimum time, minimum mass, or any other metric.
- This problem is then solved by a generic constrained optimization algorithm.

2.3.3. Indirect Method

The indirect method is based on PMP. Instead of optimizing the objective directly, it instead aims to optimize a parameter called the Hamiltonian, which is related to but distinct from the parameter of the same name in Hamiltonian mechanics. In optimal control theory, the Hamiltonian at any time depends on the performance index, which in turn is based on the optimization objective at this time [22].

The practical use of the indirect method is shown most explicitly by Mengali and Quarta [39]. Much of the approach is done analytically, including derivation of the dynamics, and the derivatives of the costates, and the optimal values of the control variables. When this is not possible, a numerical scheme can also be used [49]. Lastly, convenient implementations in vectors and matrices are presented in Betts [6].

The indirect method is summarized as follows.

1. For each state, introduce a costate. Thus, in addition to the state vector $\boldsymbol{q}$, there is a costate vector $\boldsymbol{\lambda}$.
2. Determine the derivative of each state from the dynamics, $\dot{\boldsymbol{q}}$.
3. At any time, the Hamiltonian H is found by summation of the products of state derivative and costate, and can be appended with an arbitrary cost function C.

$$H = \boldsymbol{\lambda}^T \cdot \boldsymbol{q} - C \tag{2.8}$$

4. PMP states that, at any time, an optimal control law maximizes H. Such control laws are found by analytic inspection of H, where possible, or by a numerical scheme alternatively. These laws depend on the states and costates, the exact values of which are not yet known.
5. Determine the derivatives of the costates from the Hamiltonian: $\dot{\boldsymbol{\lambda}} = -\nabla_{\boldsymbol{q}} H$

6. Given an initial state and final state, there are multiple ways to solve the resultant boundary value problem. The simplest single shooting method generates an initial guess of the costates for a boundary, as well as an initial guess for the total transfer time, then integrates towards the other boundary. Other methods include multiple shooting, collocation, and dynamic programming.

7. This yields an constrained optimization problem, with a single tolerance constraint at the remaining boundary. The optimization objective can be total transfer time, initial mass, or any other metric.

When the problem involves discontinuous thrust, one of the control variables is the throttle. From optimizing the Hamiltonian, it is clear that at any point in time the throttle should either be fully closed or fully open. This is denoted as 'bang bang' control, and controlled by a switching function. Due to the discontinuous nature of the switching function, this problem is difficult to integrate accurately [5].

There are a few extensions to the indirect method. Firstly, PMP is also applicable using Edelbaum's approximation [7]. This is accurate for circular orbits only, as such the thrust direction is constant [18]. Citing these simplifications, Casalino [8] allows for a variable I_{sp}, which may be useful implementing dual thrust. For the parts where electrical thrust is considered, eclipses significantly affect electrical power. To address eclipses, Kluever [28] presents an adaptation of Edelbaum's approximation, using a method from Neta and Vallado [43] to determine eclipse times. Furthermore, Whiffen and Sims [67] and Sauer [52] present control methods based on PMP for reaching multiple targets.

2.3.4. Evolutionary Neurocontrol

Lastly, Ohndorf [44], citing Dachwald [13], considers evolutionary neurocontrol as an alternative optimization method. Here, the control variables are linked to the state through a neural network, and the spacecraft's performance is simulated. This is optimized using a evolutionary approach: based on each simulation's fitness, a new generation of neural weights is generated which are evaluated in turn; through iteration this converges to an optimal trajectory.

This approach is interesting, because it requires no prior knowledge of the system whatsoever. Instead, the neural network learns the appropriate control output throughout iterations. This means that this method is applicable to any control optimization problem, independent of the dynamics involved and without any need to derive the dynamics beyond the basic state derivatives. The spacecraft design parameters are simply additional to the neural weight parameters, so combined optimization is not more complex than trajectory optimization, nor is dual thrust optimization more complex than single thrust problems.

Considering the novelty of this approach, as well as the requirements for a good understanding of artificial intelligence design and for considerable computational power, it has not been chosen to explore this method further in this thesis.

2.4. Dual Propulsion Systems

A dual propulsion system combines a high thrust system with a low thrust system. What constitutes a high or low thrust system depends entirely on its implementation in relation to the spacecraft and the mission geometry. For this report, the most important distinction is whether the thrust to weight ratio of any implementation is sufficient that a useful burn time is short enough to reasonably model as instantaneous. For this report, this is considered sufficient if the burn time is less than 1% of the orbital period, which for a Low Earth Orbit (LEO) of 500 km altitude would mean a burn time of less than a minute. For cubesat with masses of tens of kilograms, significant orbit changes can be realised with short burns if the thrust is in the order of Newtons.

Such thrust levels are available for systems generally based on chemical reactions: solid propellant with mixed oxidizer and fuel, hybrid propellant with liquid oxidizer and solid fuel, bipropellant with liquid oxidizer and liquid fuel, and monopropellant with a single liquid propellant activated by a catalyst. These systems have typical I_{sp} values in the order of 100 s.

For propulsion systems based on electrical power, their thrust is generally much lower, in the order of milliNewtons, with typically a higher I_{sp} in the order of 1000 s. These include ion thrusters, radio frequency thrusters, electrospray thrusters, pulsed plasma thrusters, and vacuum arc thrusters.

Cold gas thrusters and resistojets are considered to have a relatively low thrust as well, but also have a typically low I_{sp}.

For integration in a miniaturized system, Pallichadath et al. [46] note several aspects that are more critical due to this smaller form factor: specifically heat dissipation, volume constraints, and power constraints.

These miniaturized propulsion systems are often reliant on micro electromechanical systems (MEMS) technology, a process that is more reminiscent of microchip lithography than mechanical assembly. Such implementations are listed in Hitt et al. [20] and Silva et al. [53], the latter of which also presents aggregated performance ranges. Lastly, Mani et al. [35] present an overview of electric propulsion applicable to cubesats, as well as a table of useful comparable properties of different electric propulsion types. An in depth thruster performance model is presented, citing Grondein et al. [19] and Chabert et al. [12].

Mission design using dual thrust is considered in literature as well: Kluever [27] and Kluever and Pierson [29] consider using the chemical propulsion first, and electrical propulsion second. Furthermore, the distinction is stressed between solar and nuclear power, which affects the trajectory design. Alternatively, Oleson et al. [45] consider dual systems, where the electric thrust is only used for station keeping. More recently, Mani et al. [37], Mani et al. [36], and Mani et al. [38] present a full design of an interplanetary cubesat mission, using dual thrust. This design considers chemical thrust only for Earth escape, and electrical thrust for the interplanetary transfer. Mani et al. explicitly stress the need for a "combined propulsion system-hybrid trajectory design framework" as a global optimization procedure for interplanetary missions, and expect that such a model will have major consequences for cubesat use in these missions. This is the main inspiration for this thesis.

Some developments are discussed that are beyond the scope of the thesis. Rovey et al. [50] and Donius and Rovey [16] present preliminary designs in which the low thrust and high thrust systems are combined into a single thruster, and research into staging for miniaturized low thrust systems is currently underway [30]. These developments could lead to additional cost savings. Furthermore, Kluever [27] and Kluever and Pierson [29] specifically address three body dynamics for lunar missions, citing a "hybrid method" of Edelbaum's approximation and the indirect method from Dixon and Biggs [15]. Lastly, Mengali and Quarta [39] use a constant radial thrust as a simplification of dual thrust systems, which is applicable for a design combining a thruster and a solar sail. This has the benefit of greatly simplifying the dynamics, as the solar sail only effectively reduces gravitational effects, but is considered to simplistic to be useful for this thesis.

2.5. Numerical Aspects

An overview of a appropriate numerical methods is presented by Kim [26] citing Betts [6]. Overall, the found literature recommends using a global heuristic method as a rough first pass, followed by a local optimization method for finer control. Kluever notes that sequential quadratic programming (SQP) is an appropriate numerical method, citing Dixon and Biggs [15]. Kim recommends simulated annealing and agrees with Kluever that SQP is an appropriate local optimization method.

On the other hand, Dachwald [13] and Mosher [40] prefer to use evolutionary/genetic algorithms, which has benefits when combining it with an evolutionary systems design approach.

Lastly, a wide array of alternative numerical optimization schemes is available, which are not further discussed here. However, many implementations are freely available and modifiable. These include `scipy`[1], a general library for scientific programming; `pykep` [21], a low thrust trajectory optimizer developed at the European Space Agency (ESA); `pygmo`, the related optimization library; and `gekko` [3], an optimal control problem solver developed at Brigham Young University. Common optimization algorithms included in these projects are sequential least squares programming, included in `scipy` as `slsqp`, and `ipopt`[2].

2.6. Verification & Validation

Verification and validation of the design is typically done through comparison with previous results; with either simpler of similar implementations.

For verification, Jiang et al. [22] and Dachwald [13] compare their resultant trajectory with trajectories determined previously in older literature. For the systems engineering aspect, Mengali and Quarta [39] inspect the slope of a certain PMP parameter, such that the solution is at least locally optimal. For a newly developed optimization model, most authors, including Mosher, Sims, and Whiffen, individually verify their model against a previously developed model.

For the thesis, verification must be done through comparison with existing literature with both trajectory and systems design.

Mosher et al. [42] use a database of flown missions to validate the system design result, and evaluates the practical use of the model for the design of new missions, to validate the model itself. The resulting

[1] `https://www.scipy.org`
[2] `https://coin-or.github.io/Ipopt/`

component requirements can also be compared to existing components [35].

For the trajectory, Ohndorf [44] justifies the use of the integration scheme and the optimization scheme, and validates the trajectory against a large number of examples from literature.

For the thesis, the trajectory results must be compared with existing missions. Furthermore, the system design must be compared with existing spacecraft; this may be difficult to perform at significantly high level of detail, as few similar existing designs exist. Lastly, the designed components must be compared with available components; as CubeSat components are highly standardized, comparisons can be made within very similar contexts, which is considered useful.

2.7. Conclusion

The literature study provides useful starting point to develop a combined optimization framework for trajectory design and systems design for a dual thrust system.

It provides an understanding of several low thrust trajectory optimization methods, including Edelbaum's approximation, the direct method, the indirect method (using PMP), and evolutionary neurocontrol. For these, the indirect method has been subject to research for including systems design capabilities. Additionally, evolutionary neurocontrol is a promising alternative, but only in the early stages of development.

It is concluded that the indirect method can be adapted to be used in dual thrust scenarios, by adding a second thrust factor to the dynamics; and that system design can be implemented, by setting its parameters as static control parameters.

3

Design Problem

This chapter defines the problem, to which the design presented in chapter 6 should ultimately be a solution. First, the intent of the design is discussed. From this, general objectives are derived, as well as specific requirements. Second, the design process is described, as determined to be applicable from the literature study. From the literature study's analysis on available numerical solvers, there are two general options in consideration.

3.1. Intended Achievement

As discussed in the introduction, the ultimate goal of this thesis is to advance the use of standardized and miniaturized spacecraft technology towards self-contained interplanetary missions, in the same way it is already possible for missions in Earth orbit.

To achieve this, it is considered to set an example mission: in turn, it is expected that all relevant aspects will be encountered naturally. Specifically, this focuses on dual thrust, optimization of the trajectory, and specific implementations of the spacecraft itself.

3.2. Design Objectives

From this intent, it is possible to determine some objectives of the design, which are listed below. Considering the design problem as an optimization problem, these objectives form a general notion of optimality of the design.

These objectives are not to be confused with requirements, i.e. the strictly defined criteria for a design to be considered feasible. Instead, objectives are determined as an overall metric, towards which the design will pushed as long as the requirements are met.

In other words, if the design problem is considered a constrained optimization problem, the requirements form the constraints, and the objectives form in some way the evaluation function.

The objectives are as follows. It is considered that the more the spacecraft adheres to these objectives, the more likely it is that the goal of this thesis is reached.

- OBJ-1: The spacecraft shall be optimized towards minimal cost.
- OBJ-2: The spacecraft shall be optimized towards minimal mass.
- OBJ-3: The spacecraft shall consist as much as possible of COTS components.
- OBJ-4: The spacecraft shall consist as much as possible of components with a high technology readiness level (TRL).
- OBJ-5: The trajectory shall be optimized towards a minimal total radiation dose throughout the mission.

3.3. Design Requirements

From the objectives, the top level requirements are derived. These are considered to be 'SMART', i.e. specific, measurable, achievable, realistic, and time bound. The design is considered feasible if and only if it meets these requirements, independent of possible improvements towards the objectives. On the other hand, by pushing an infeasible design in the direction of the objectives, it is more likely that the design becomes feasible.

- REQ-1: The spacecraft shall be launched as a piggyback option or secondary payload.

  Rationale: This allows for minimization of launch costs.

  - REQ-1.1: The spacecraft shall adhere to the CubeSat standard.

    Rationale: This is required for low cost piggyback launches, and allows for minimization of design, acquisition, integration, and test costs.

  - REQ-1.2: The spacecraft shall have a volume of at most 16 U.

    Rationale: This is the maximum launch option as given by many launch providers available for a request online, and a common size in literature for larger cubesats.

  - REQ-1.3: The spacecraft shall have a mass of at most 21.3 kg

    Rationale: This is the guideline maximum mass of a 16 U cubesat, according to the standard. While more massive cubesats exist, adhering more strictly tot his standards will ensure compatibility with launch and deployment providers in terms of mass.

  - REQ-1.4: The spacecraft shall be launched into an initial orbit for which piggyback options exist.

    Rationale: This allows for minimization of launch costs.

- REQ-2: The spacecraft shall allow a payload of 1 U volume

  Rationale: This is a common cubesat payload size, and sufficient for e.g. a single optical imager [10].

- REQ-3: The spacecraft shall allow a payload of 1.3 kg

  Rationale: This is the guideline maximum mass of a cubesat unit, and sufficient for e.g. a single optical imager [10].

- REQ-4: The spacecraft shall allow a payload to draw 1 W of operating power when not performing a maneuver

  Rationale: this is sufficient to power, for example, a single optical imager [10].

- REQ-5: The spacecraft shall reach a final orbit around Mars.

  Rationale: An interplanetary mission is the goal of this case study. No restriction is given on the actual final orbit, due to its dependence on the payload, which is beyond the scope of this study. While actual final orbit affects the ΔV requirements, it is considered that any orbit around Mars is acceptable. This means that different final orbits may be chosen to minimize the required ΔV.

- REQ-6: The spacecraft shall reach its final orbit in at most 5 years.

  Rationale: This is considered long enough to make a transfer possible, but short enough to account for degradation of components. Additionally, it a common time span in similar feasibility studies.

- REQ-7: The spacecraft be launched in the year 2030 at the latest.

  Rationale: A window is required for trajectory optimization; this is considered sufficiently close to lead to a relevant design, as well as sufficiently ahead to allow enough time for development.

- REQ-8: The spacecraft shall not use hydrazine thrusters.

  Rationale: This is a common sustainability constraint for new spacecraft concepts.

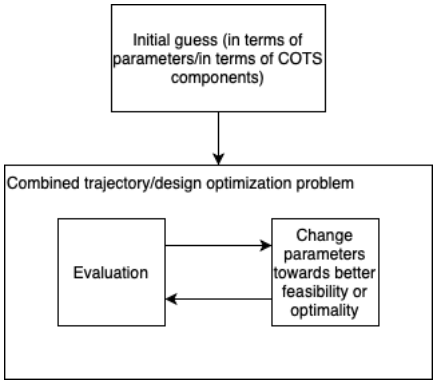

Figure 3.1: Overview of the overall process.

3.4. Derived Requirements

Typically, further requirements are derived from the top level requirements, such as specifics on propulsion or trajectory. However, in this report, these requirements are subject to optimization themselves. For each design iteration, specific requirements may change: for example, if an iteration has an overall lower mass, a propulsion system with a lower thrust may suddenly be a feasible design option. It is expected that these requirement will change rapidly on iterations, and are interdependent between the many components. As such, they will not be listed in advance.

Due to the nature of the proposed optimization procedure, only the few presented objectives and requirements are necessary to begin the design procedure; this is less than is typical for a systems design problem.

3.5. Scope

It is necessary to consolidate the scope of the design, in order to limit the level of detail for such a preliminary study. The main point of research lies in combination of system design and trajectory, as such the astrodynamics will be considered fully.

Secondly, the propulsion system will be a main focus point. This includes: high thrust: type, propellant, thrust, I_{sp}, power; low thrust: type, propellant, thrust, I_{sp}, power; propellant tanks; thermal requirements; overall mass; and overall volume. Similarly, the power system is expected to be critically important for the low thrust propulsion system, and will be considered fully. This includes: solar power input, eclipses, degradation, power throughput, charged capacity, mass, and volume. Lastly, radiation avoidance is a major point of optimization for the trajectory and the design. As such will be included the tolerances and degradation of individual components.

For the remaining aspects of the design, the level of detail will be limited such that only the budgets will be considered. For attitude determination & control, this includes estimations for required pointing accuracy, power usage, mass, and volume. For the payload, power usage, mass, and volume are set as requirements. For communication, a high level link budget is considered sufficient, in addition to power usage, mass and volume. Thermal control and command and data handling are considered only qualitatively based on estimates.

This level of detail is considered sufficient, as any lower level considerations would deal with detailed design of individual components. The focus on this study is on the optimization aspect of general system design and trajectory design, and restricting the scope will allow for more time allocated in this area.

3.6. Design Process Overview

In general, the design process is considered as solving a constrained optimization problem. The design requirements form the constraints, so a design is feasible if it meats all constraints. Within the feasibility space, the design is optimized towards the design objectives: minimal cost with metric with some uncertainty.

As such, the design process considers an “initial guess”, which is unusual for system engineering designs. Then, from this initial guess, the design iterates first towards the feasibility space, and then within the feasibility space towards an optimal point.

This means that the overall process can be depicted simplistically, as in Figure 3.1.

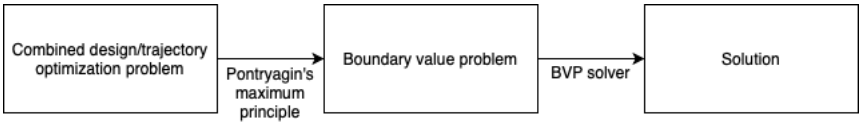

Figure 3.2: Overview of the implementation from scratch, using the indirect approach and an existing BVP solver.

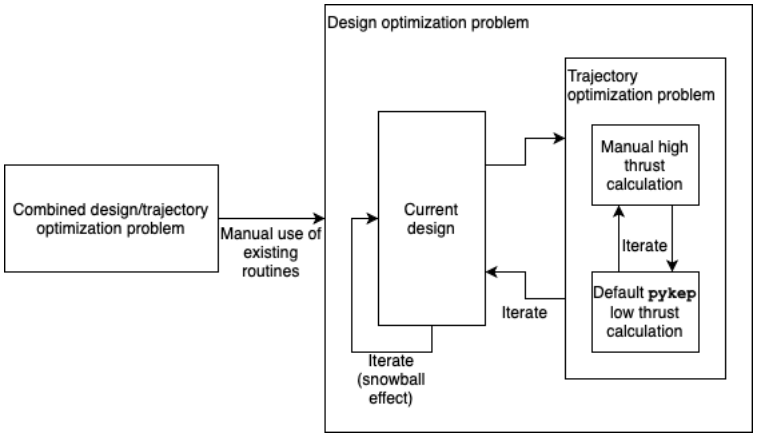

Figure 3.3: Overview of the implementation using `pykep` as is.

From the literature study, the number of global options for implementing the design was narrowed down to two, each using freely available and modifiable software.

The first option is to write a new implementation of the indirect method from scratch, using static variables to encode system design parameters. This can be done in two ways, either using a boundary value problem solver, such as included in the `scipy` package, or using a generic solver for optimal control problems (OCPs), such as `gekko`.

The second option is to use `pykep`, a full fledged low thrust trajectory optimization package. Here too, there are two ways to use `pykep`: first by performing system design iterations manually, and running `pykep` for iterations, secondly by implementing an automated system design optimization object in `pykep`.

This leads to the aforementioned research question, *"Which implementation is useful for combined optimization of a dual thrust system?"*, which will be answered during the implementation phase.

3.7. Using a BVP or OCP Solver

The preferred implementation uses the theory as presented in the literature study. The spacecraft trajectory is optimized using the indirect method, in which the direction and level of thrust are encoded as dynamic control parameters. The system design parameters, such as thrust, I_{sp}, power, and structural masses, are encoded as static control parameters. As the Hamiltonian is defined to include the cost function as required, the entire problem is optimized towards minimal cost, including the trajectory design and system design. This process is depicted in Figure 3.2.

An alternative implementation uses `gekko`, which derives the optimal control laws automatically, but is theoretically equivalent.

3.8. Using a Trajectory Optimizer

Currently, `pykep`'s implementation only supports single low thrust systems, and has no support for varying system design parameters within an optimization problem. As such, `pykep` can be used as is, if used with manual iterations, and manually calculating high thrust sections of the trajectory. This method does not require writing a trajectory optimizer from scratch, but this also means it is not an automated solution. This method is depicted in Figure 3.3.

Figure 3.3 requires additional explanation. Overall, the combined problem is split into two nested problems: a design optimization problem, with internally a trajectory optimization problem. This latter setup is tackled as follows. From the current design, the trajectory is optimized, implemented within `pykep`, with additional code for high thrust optimization. Firstly, `pykep`'s low thrust optimization process is an iterative

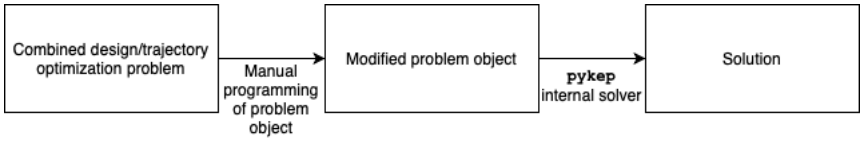

Figure 3.4: Overview of the implementation using a custom `pykep` problem object.

one. The result of this is iterated with the high thrust calculation to converge towards a feasible dual thrust trajectory: the low thrust trajectory affects the high thrust trajectory and vice versa. The resultant complete trajectory is in turn iterated with the system design: a trajectory change affects the design, and the design affects the trajectory. Lastly, the system design itself must be iterated as well: a change is one component affects the other components as well, this is known as the snowball effect. It is clear that this method uses many nested iterations, which is its most important drawback.

Alternatively, it should be possible to adapt `pykep`, to automatically optimize two legs for a trajectory, e.g. an escape trajectory and an interplanetary trajectory. This is possible, because `pykep` already has the possibility to calculate multi-leg trajectories for the same spacecraft.

Internally, `pykep` works with problem objects as defined by the `pygmo` module, developed concurrently by ESA. The specific `trajopt` objects describe low thrust trajectory optimization problems. It may be possible to instantiate a new `pygmo` problem object using a similar definition to the `trajopt` objects, such that the decision vector includes for example the optimal time of switching, and the propellant masses. This option is depicted in Figure 3.4.

3.9. Conclusions

With the design problem defined, it is explicit what can be considered an acceptable design, and what constitutes a 'better' design within the acceptable design range.

While the scope of the design is limited compared to a full system design exercise, this is necessary to focus on the optimization aspect of general system design and trajectory design. In turn, this will ultimately result in several recommendations in improving the design further under a larger scope.

Lastly, it should be noted this report considers this design problem, as well as design problems in general, to be a constrained optimization problem which can be solved as such. This view follows the Systems Engineering discipline in placing significant importance of the requirements and objectives of a design problem, and may be a useful attitude as a solution based approach.

4

Theoretical Approach

Independent of the choice of implementation, it is necessary to present the theoretical aspects of a combined dual thrust optimization problem. For an implementation from scratch, it is necessary to completely describe the dynamics and optimal control laws. For the implementation using `pykep`, this has been done internally, but an understanding is still necessary to use `pykep`: either to adapt the problem object or to iterate the results manually.

This chapter describes the theoretical approach for coupled optimization of a dual thrust system, for any implementation without loss of generality.

4.1. State

A spacecraft state vector $\boldsymbol{q}$ for a dual propulsion spacecraft includes 6 components related to position and velocity. In terms of Kepler elements, there are 5 slow components and one fast component; in terms of other coordinate systems, there are 3 position components and 3 velocity components. It must be noted that the derivations here are made in terms of coordinate systems, but it is applicable to Kepler elements as well.

With coordinate systems, the state vector is divided in the position vector $\boldsymbol{r}$, velocity vector $\boldsymbol{s}$, the chemical propellant mass m_c, electrical propellant mass m_e, and the current charged capacity of the electrical power system (EPS), c_e. All are variable in time t.

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{s} \\ m_c \\ m_e \\ c_e \end{bmatrix} \tag{4.1}$$

It must be noted that battery charge is only relevant when eclipses are considered: otherwise, the power usage of the electrical thruster can be assumed to equal the power output of the solar panels. This is not necessarily a realistic assumption, however, it can be considered that a large fraction of the incident power is used by the thruster, and any other power draws can be considered as losses in efficiency without further complicating the dynamics.

Using the first derivative with respect to time, the dynamics are given as a differential equation.

$$\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{a} \\ \dot{m}_c \\ \dot{m}_e \\ \dot{c}_e \end{bmatrix} \tag{4.2}$$

4.2. Accelerations

The state derivatives are expanded one by one. First, the acceleration is due to gravity $\boldsymbol{a}_g$, and due to the chemical and electric propulsion systems.

$$\boldsymbol{a} = \boldsymbol{a}_g + \boldsymbol{a}_c + \boldsymbol{a}_e \tag{4.3}$$

Gravity is determined from Newton's law, over planets ($_j$).

$$\boldsymbol{a}_g = \sum_j \frac{\mu_j(\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \tag{4.4}$$

For a single planet, with the origin at the planet's center of mass, this reduces to.

$$\boldsymbol{a}_g = -\frac{\mu_j \boldsymbol{r}}{||\boldsymbol{r}||^3} \tag{4.5}$$

For both thrusts exist dynamic control vectors $\boldsymbol{\alpha}_c$ and $\boldsymbol{\alpha}_e$, the control throttles h_c, and h_e the maximum thrust F_c and F_e. For both, the acceleration depends on the instantaneous mass, which includes the unchanging structural mass m_s and payload mass m_p. The structural mass is considered a function of the static control variables, the initial state, and the maximum used capacity; the payload mass is considered given.

$$\boldsymbol{a}_c = \frac{\boldsymbol{\alpha}_c F_c}{m_c + m_e + m_s + m_p} \tag{4.6}$$

$$\boldsymbol{a}_e = \frac{\boldsymbol{\alpha}_e F_e}{m_c + m_e + m_s + m_p} \tag{4.7}$$

$$||\boldsymbol{\alpha}_c|| \leq 1 \tag{4.8}$$

$$||\boldsymbol{\alpha}_e|| \leq 1 \tag{4.9}$$

4.3. State Changes

For the mass flows, the specific impulse I_{sp} is scaled with the standard Earth surface gravity g_0.

$$\dot{m}_c = -\frac{||\boldsymbol{\alpha}_c|| F_c}{g_0 I_{sp_c}} \tag{4.10}$$

$$\dot{m}_e = -\frac{||\boldsymbol{\alpha}_e|| F_e}{g_0 I_{sp_e}} \tag{4.11}$$

$$0 \leq m_c \tag{4.12}$$

$$0 \leq m_e \tag{4.13}$$

The change in charged capacity is dependent on the power usage and power generation, the latter of which is considered to be inversely proportional to the square of the distance to the sun, denoted with subscript ($_s$). The power generation is scaled with respect to the standard solar flux at 1 AU. The maximum capacity $c_{e,m}$ depends on the maximum required use.

$$\dot{c}_e = -||\boldsymbol{\alpha}_c|| p_e + p_s \left(\frac{1AU}{||\boldsymbol{r}_s - \boldsymbol{r}||} \right)^2 \tag{4.14}$$

$$0 \leq c_e \leq c_{e,m} \tag{4.15}$$

4.4. Static Control

A static control vector $\boldsymbol{w}$ is introduced to contain all variables of optimization which do not depend on time.

$$\boldsymbol{w} = \begin{bmatrix} F_c \\ F_e \\ I_{sp_c} \\ I_{sp_e} \\ p_e \\ p_s \\ t_f \end{bmatrix} \tag{4.16}$$

The last variable is the transfer time t_f.

Finally, the cost function is considered to be the sum of functions of all static control variables.

$$C(\boldsymbol{w}) = f(F_c, I_{sp_c}) + f(F_e, I_{sp_e}, p_e) + f(p_s, c_{e,m}) + f(t_f) \tag{4.17}$$

In addition, a cost should be added for the time spent in high-radiation zones zones, which can be determined from the given trajectory.

$$C_{+\text{rad}}(\boldsymbol{w}) = C(\boldsymbol{w}) + f(\boldsymbol{r}) \tag{4.18}$$

The use of this cost function depends on the implementation. For an automated approach, the cost function is used as the target for optimization.

4.5. Hamiltonian

It is considered that this general problem can be solved using PMP [26]. PMP requires construction of the Hamiltonian H, containing the derivative of the state, a costate vector $\boldsymbol{\lambda}$, and the cost function.

$$H = \boldsymbol{\lambda}^T \cdot \dot{\boldsymbol{q}} + C(\boldsymbol{w}) \tag{4.19}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_r \\ \boldsymbol{\lambda}_s \\ \lambda_{m_c} \\ \lambda_{m_e} \\ \lambda_{c_e} \end{bmatrix} \tag{4.20}$$

$$H = \boldsymbol{\lambda}_r \cdot \boldsymbol{s} + \boldsymbol{\lambda}_s \cdot \left(\sum_j \frac{\mu_j(\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} + \frac{\boldsymbol{\alpha}_c F_c + \boldsymbol{\alpha}_e F_e}{m_c + m_e + m_s + m_p} \right) \tag{4.21}$$

$$- \lambda_{m_c} \frac{||\boldsymbol{\alpha}_c|| F_c}{g_0 I_{sp_c}} - \lambda_{m_e} \frac{||\boldsymbol{\alpha}_e|| F_e}{g_0 I_{sp_e}} + \lambda_{c_e} \left(-||\boldsymbol{\alpha}_e|| p_e + p_s \left(\frac{1\,AU}{||\boldsymbol{r}_s - \boldsymbol{r}||} \right)^2 \right) \tag{4.22}$$

$$+ C(\boldsymbol{w}) \tag{4.23}$$

4.6. Optimal Dynamic Control

Based on the theory of optimal control, H must be maximized. This means the direction of $\boldsymbol{\alpha}_j$ can be restricted. From analytical inspection, it is clear that H is maximised if $\boldsymbol{\lambda}_s$ and $\boldsymbol{\alpha}_c$ are in the same direction. The same holds for $\boldsymbol{\alpha}_e$. Therefore, we find the optimal direction for both $\boldsymbol{\alpha}_c$ and $\boldsymbol{\alpha}_e$.

$$\boldsymbol{\alpha} = \frac{\boldsymbol{\alpha}_c}{||\boldsymbol{\alpha}_c||} = \frac{\boldsymbol{\alpha}_e}{||\boldsymbol{\alpha}_e||} = \frac{\boldsymbol{\lambda}_s}{||\boldsymbol{\lambda}_s||} \tag{4.24}$$

It can be concluded from optimal control theory, that it will never be necessary to simultaneously fire both thrusters in different directions. Instead, it is useful to introduce throttle settings h for both thrusters, and use a general thrust direction unit vector $\boldsymbol{\alpha}$.

$$\boldsymbol{\alpha}_j = \boldsymbol{\alpha} h_j \tag{4.25}$$

This simplifies H. Therefore, $H_{\boldsymbol{\alpha}}$, the part of H dependent on $\boldsymbol{\alpha}$, is shown.

$$\begin{aligned} H_{\boldsymbol{\alpha}} &= \boldsymbol{\lambda}_s \frac{\boldsymbol{\alpha}(h_c F_c + h_e F_e)}{m_c + m_e + m_s + m_p} = \frac{\boldsymbol{\lambda}_s \cdot \boldsymbol{\lambda}_s}{||\boldsymbol{\lambda}_s||} \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} \\ &= \frac{||\boldsymbol{\lambda}_s||^2}{||\boldsymbol{\lambda}_s||} \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} = ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} \end{aligned}$$

For h_j, H must be maximized as as well. This requires maximization of H_{h_j}, i.e. the parts of H dependent on h_j.

$$H_{h_j} = ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} - \lambda_{m_c} \frac{h_c F_c}{g_0 I_{sp_c}} - \lambda_{m_e} \frac{h_e F_e}{g_0 I_{sp_e}} - \lambda_{c_e} h_e p_e \tag{4.26}$$

$$H_{h_c} = ||\boldsymbol{\lambda}_s|| \frac{h_c F_c}{m_c + m_e + m_s + m_p} - \lambda_{m_c} \frac{h_c F_c}{g_0 I_{sp_c}} \tag{4.27}$$

$$H_{h_e} = ||\boldsymbol{\lambda}_s|| \frac{h_e F_e}{m_c + m_e + m_s + m_p} - \lambda_{m_e} \frac{h_e F_e}{g_0 I_{sp_e}} - \lambda_{c_e} h_e p_e \tag{4.28}$$

It is useful to introduce 'switching functions' k_j, such that $H_{h_j} = k_j h_j$.

$$k_c = ||\boldsymbol{\lambda}_s|| \frac{F_c}{m_c + m_e + m_s + m_p} - \lambda_{m_c} \frac{F_c}{g_0 I_{sp_c}} \tag{4.29}$$

$$k_e = ||\boldsymbol{\lambda}_s|| \frac{F_e}{m_c + m_e + m_s + m_p} - \lambda_{m_e} \frac{F_e}{g_0 I_{sp_e}} - \lambda_{c_e} p_e \tag{4.30}$$

It appears that the switching functions for the two thrusters are different, so the thrusters will not necessarily fire together. This shows that there may be a theoretical advantage to having two different thrusters.

The optimal value for h_c and h_e depends on k_c and k_e, respectively.

$$h = \begin{cases} 1, k > 0 \\ 0, k < 0 \end{cases} \tag{4.31}$$

Note that the value of h_j is irrelevant for $k_j = 0$.

If eclipses are considered and the spacecraft is in eclipse, it must be confirmed that $h_e = 0$ if $p_e = 0$, because the electrical thruster cannot fire if the battery is empty. If the thruster does fire in this case, an additional constraint has to be added to the k_e switching function.

All in all, H is slightly simplified.

$$\begin{aligned} H = \boldsymbol{\lambda}_r \cdot \boldsymbol{s} + \boldsymbol{\lambda}_s \cdot \left(\sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) + ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} \\ - \lambda_{m_c} \frac{h_c F_c}{g_0 I_{sp_c}} - \lambda_{m_e} \frac{h_e F_e}{g_0 I_{sp_e}} - \lambda_{c_e} h_e p_e + \lambda_{c_e} p_s \left(\frac{1\,AU}{||\boldsymbol{r}_s - \boldsymbol{r}||} \right)^2 \\ + C(\boldsymbol{w}) \end{aligned}$$

4.7. Costates

Although the costates themselves cannot be directly expressed, the derivatives of the costates are found using the gradient with respect to the state [26].

4.7.1. Derivation for Earth Orbits

A derivation within a single SOI is significantly less complex.

$$H = \boldsymbol{\lambda}_r \cdot \boldsymbol{s} + \boldsymbol{\lambda}_s \cdot \left(-\frac{\mu \boldsymbol{r}}{||\boldsymbol{r}||^3} \right) + ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p}$$
$$- \lambda_{m_c} \frac{h_c F_c}{g_0 I_{sp_c}} - \lambda_{m_e} \frac{h_e F_e}{g_0 I_{sp_e}} - \lambda_{c_e} h_e p_e + \lambda_{c_e} p_s + C(\boldsymbol{w})$$

$$\dot{\boldsymbol{\lambda}}_r = -\nabla_{\boldsymbol{r}} H = -\nabla_{\boldsymbol{r}} \left(\boldsymbol{\lambda}_s \cdot \left(-\frac{\mu \boldsymbol{r}}{||\boldsymbol{r}||^3} \right) \right) = \mu \boldsymbol{\lambda}_s \cdot \nabla_{\boldsymbol{r}} \left(\frac{\boldsymbol{r}}{||\boldsymbol{r}||^3} \right) \tag{4.32}$$

$$= \mu \boldsymbol{\lambda}_s \cdot \nabla_{\boldsymbol{r}} \frac{||\boldsymbol{r}||^3 - \boldsymbol{r} \cdot \nabla_{\boldsymbol{r}} ||\boldsymbol{r}||^3}{||\boldsymbol{r}||^6} = \mu \boldsymbol{\lambda}_s \cdot \nabla_{\boldsymbol{r}} \frac{||\boldsymbol{r}||^3 - \boldsymbol{r} \cdot \left(3||\boldsymbol{r}||^2 \frac{\boldsymbol{r}}{||\boldsymbol{r}||} \right)}{||\boldsymbol{r}||^6} \tag{4.33}$$

$$= \mu \boldsymbol{\lambda}_s \cdot \nabla_{\boldsymbol{r}} \frac{||\boldsymbol{r}||^3 - \boldsymbol{r} \cdot (3||\boldsymbol{r}||\boldsymbol{r})}{||\boldsymbol{r}||^6} \tag{4.34}$$

$$= \mu \boldsymbol{\lambda}_s \cdot \nabla_{\boldsymbol{r}} \frac{||\boldsymbol{r}||^3 - 3||\boldsymbol{r}||^3}{||\boldsymbol{r}||^6} \tag{4.35}$$

$$= -2\mu \boldsymbol{\lambda}_s ||\boldsymbol{r}||^3 \tag{4.36}$$

This aligns with the findings of Jiang et al. [22].

4.7.2. Derivation for Interplanetary Trajectories

A derivation for a true multi-body simulation is more complex.

$$\dot{\boldsymbol{\lambda}}_r = -\nabla_{\boldsymbol{r}} H = -\nabla_{\boldsymbol{r}} \left(\boldsymbol{\lambda}_s \cdot \left(\sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) + \lambda_{c_e} p_s \left(\frac{1\,AU}{||\boldsymbol{r}_s - \boldsymbol{r}||} \right)^2 \right) \tag{4.37}$$

(Reorder last term.)

$$= -\nabla_{\boldsymbol{r}} \left(\boldsymbol{\lambda}_s \cdot \sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1\,AU)^2 \nabla_{\boldsymbol{r}} \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}||^2} \tag{4.38}$$

(Multiplication rule in first term.)

$$= -\nabla_{\boldsymbol{r}} \boldsymbol{\lambda}_s \cdot \left(\sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \boldsymbol{\lambda}_s \cdot \left(\nabla_{\boldsymbol{r}} \sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1\,AU)^2 \nabla_{\boldsymbol{r}} \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}||^2} \tag{4.39}$$

(Reduce first term.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\nabla_{\boldsymbol{r}} \sum_j \frac{\mu_j (\boldsymbol{r}_j - \boldsymbol{r})}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1\,AU)^2 \nabla_{\boldsymbol{r}} \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}||^2} \tag{4.40}$$

(Reorder first term.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \nabla_{\boldsymbol{r}} \frac{\boldsymbol{r}_j - \boldsymbol{r}}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1\,AU)^2 \nabla_{\boldsymbol{r}} \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}||^2} \tag{4.41}$$

(Chain rule.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \nabla_{\boldsymbol{r}} (\boldsymbol{r}_j - \boldsymbol{r}) \nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} \frac{\boldsymbol{r}_j - \boldsymbol{r}}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1\,AU)^2 \nabla_{\boldsymbol{r}} (\boldsymbol{r}_s - \boldsymbol{r}) \nabla_{(\boldsymbol{r}_s - \boldsymbol{r})} ||\boldsymbol{r}_s - \boldsymbol{r}|| \nabla_{||\boldsymbol{r}_s - \boldsymbol{r}||} \frac{1}{||\boldsymbol{r}_s - \boldsymbol{r}||^2} \tag{4.42}$$

(Evaluate last term.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \nabla_{\boldsymbol{r}} (\boldsymbol{r}_j - \boldsymbol{r}) \nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} \frac{\boldsymbol{r}_j - \boldsymbol{r}}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - \lambda_{c_e} p_s (1AU)^2 \cdot -1 \cdot \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||} \frac{-2}{||\boldsymbol{r}_s - \boldsymbol{r}||^3} \tag{4.43}$$

(Simplify last term.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \nabla_{\boldsymbol{r}} (\boldsymbol{r}_j - \boldsymbol{r}) \nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} \frac{\boldsymbol{r}_j - \boldsymbol{r}}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.44}$$

(Evaluate first term.)

$$= -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \cdot -1 \cdot \frac{||\boldsymbol{r}_j - \boldsymbol{r}||^3 - (\boldsymbol{r}_j - \boldsymbol{r}) \nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} ||\boldsymbol{r}_j - \boldsymbol{r}||^3}{||\boldsymbol{r}_j - \boldsymbol{r}||^6} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.45}$$

(Evaluate last derivative.)

$$\nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} ||\boldsymbol{r}_j - \boldsymbol{r}||^3 = \nabla_{(\boldsymbol{r}_j - \boldsymbol{r})} ||\boldsymbol{r}_j - \boldsymbol{r}|| \cdot 3||\boldsymbol{r}_j - \boldsymbol{r}||^2 = \frac{\boldsymbol{r}_j - \boldsymbol{r}}{||\boldsymbol{r}_j - \boldsymbol{r}||} \cdot 3||\boldsymbol{r}_j - \boldsymbol{r}||^2 = 3(\boldsymbol{r}_j - \boldsymbol{r})||\boldsymbol{r}_j - \boldsymbol{r}|| \tag{4.46}$$

(Substitute.)

$$= \dot{\boldsymbol{\lambda}}_r = -\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \cdot -1 \cdot \frac{||\boldsymbol{r}_j - \boldsymbol{r}||^3 - (\boldsymbol{r}_j - \boldsymbol{r}) \cdot 3(\boldsymbol{r}_j - \boldsymbol{r})||\boldsymbol{r}_j - \boldsymbol{r}||}{||\boldsymbol{r}_j - \boldsymbol{r}||^6} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.47}$$

(Simplify.)

$$= \boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \frac{||\boldsymbol{r}_j - \boldsymbol{r}||^3 - (\boldsymbol{r}_j - \boldsymbol{r}) \cdot 3(\boldsymbol{r}_j - \boldsymbol{r})||\boldsymbol{r}_j - \boldsymbol{r}||}{||\boldsymbol{r}_j - \boldsymbol{r}||^6} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.48}$$

$$= \boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \frac{||\boldsymbol{r}_j - \boldsymbol{r}||^3 - 3||\boldsymbol{r}_j - \boldsymbol{r}||^3}{||\boldsymbol{r}_j - \boldsymbol{r}||^6} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.49}$$

$$= \boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \frac{-2||\boldsymbol{r}_j - \boldsymbol{r}||^3}{||\boldsymbol{r}_j - \boldsymbol{r}||^6} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.50}$$

$$\boldsymbol{\lambda}_{\boldsymbol{r}} = -2\boldsymbol{\lambda}_s \cdot \left(\sum_j \mu_j \frac{1}{||\boldsymbol{r}_j - \boldsymbol{r}||^3} \right) - 2\lambda_{c_e} p_s (1AU)^2 \frac{\boldsymbol{r}_s - \boldsymbol{r}}{||\boldsymbol{r}_s - \boldsymbol{r}||^4} \tag{4.51}$$

4.8. Other Costates

The remaining costates are simpler to derive, regardless of the complexity of the dynamics.

$$\dot{\boldsymbol{\lambda}}_s = -\nabla_{\boldsymbol{s}} H = -\boldsymbol{\lambda}_{\boldsymbol{r}} \tag{4.52}$$

$$\dot{\lambda}_{m_c} = -\frac{\partial H}{\partial m_c} = -\frac{\partial}{\partial m_c} \left(||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} \right) = ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{(m_c + m_e + m_s + m_p)^2} \tag{4.53}$$

$$\dot{\lambda}_{m_e} = -\frac{\partial H}{\partial m_e} = -\frac{\partial}{\partial m_e} \left(||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{m_c + m_e + m_s + m_p} \right) = ||\boldsymbol{\lambda}_s|| \frac{h_c F_c + h_e F_e}{(m_c + m_e + m_s + m_p)^2} \tag{4.54}$$

The above shows that the costates for the propellant masses are identical. This reduces the number of costates by one.

$$\dot{\lambda}_m = \dot{\lambda}_{m_c} = \dot{\lambda}_{m_e} \tag{4.55}$$

$$\dot{\lambda}_{c_e} = -\frac{\partial H}{\partial c_e} = 0 \tag{4.56}$$

As such, λ_{c_e} is always zero, and can be ignored in all equations it appears in. This may mean that charged capacity is not a useful parameter to include, although this may not be possible to simply remove this parameter, depending on the implementation.

4.9. Conclusions

From the theory alone, it is concluded that the thruster never need to fire in different directions simultaneously, and that λ_{c_e} may need to be reworked depending on the implementation.

The determination of the costates concludes the theoretical aspects of optimal control. Each expression is programmable within a model, but the usage differs depending on implementation and problem. This includes choices of dimensionality, coordinate system, number of thrusters; as well as other considerations such as solar distance, eclipses, degradation, tidal forces, and drag.

5

Implementation

This chapter describes the implementation attempts, to adapt the theory as determined in the previous chapter to a design process as described in chapter 3.

5.1. Implementation using a BVP Solver

Building upon the theory in the previous chapter, it is thought possible to solve any optimization problem, including combined optimization of dual thrust trajectories and systems design. By transforming the problem into a boundary value problem, one should only need an algorithm to solve a generic BVP, such as `scipy`'s `solve_bvp` function. It is considered out of the scope of this thesis to develop a BVP algorithm from scratch.

Such a process would work as follows.

1. Given: payload mass, mass function from parameters (thrust, I_{sp}, ...), cost functions from parameters (thrust, I_{sp}, ...), cost functions from trajectory (time, radiation, ...), dynamics functions.
2. Derive the equations algebraically.
   (a) Multiply the dynamics by a time scaling factor.
   (b) Derive the Hamiltonian.
   (c) Find the analytical expression for optimal thrust as done in section 4.6. If the dynamics are complicated such that an analytic expressions cannot be found, a numerical method can be used.
   (d) Simplify the Hamiltonian using the found expression optimal thrust, or, if this was not found, its numerical approximation.
   (e) Find the optimal thrust level, and its applicable switching function.
   (f) Find the derivatives of the costates.
   (g) Find the derivatives of H with respect to the static control parameters (time scaling, spacecraft parameters, ...). If this is infeasible, a numerical method can be used.
3. Set up the boundary value problem numerically.
   (a) Write the state and costate derivatives as a function of the states and costates.
   (b) Write the boundary condition function; outputs zero when conditions are met.
   (c) Set the derivatives of H with respect to static parameters to zero at boundary conditions. If a numerical method is used, set the static parameters such that H is maximized instead.
4. Generate an initial guess.
5. Run the BVP solving algorithm.

5.1.1. Automation of Derivations

While the derivations in the previous chapter are suitable for the general problem definition, generating the costates anew for a specific problem is tedious. As such, it is possible to automate this process using an symbolic programming language such as `sympy`[1].

A short introduction to `sympy` suffices. As a symbolic programming language, `sympy` allows manipulation of algebraic variables in the same way that one would do so by hand. While a great array of functionality is available, three specific functions are sufficient for this case. Firstly, symbols are instantiated using, for example, `r = symbols('r')`, such that `r` can take any real or complex value. This is comparable with `matlab` syntax. Secondly, variables can be assigned a function of these symbols. For example, `H = r + ...` assigns to `H` a function of `r`. In this case, variables can be substituted. For example, `H = H.subs(r, sqrt(x**2 + y**2))`, would replace any `r` in `H` with an equivalent expression. Lastly, `diff(H, r)` would return the derivative of `H` with respect to `r` as a symbolic equation.

A simple example for a optimal control problem is given below. Note that this example uses a polar coordinate system in two dimensions; a discussion on the use of coordinate systems can be found in subsection 5.1.6.

1. Define states, costates, constants, and control variables, and define the equations of motion.

```
from sympy import symbols, sin, cos, atan2, diff

r, rd, thd, m, lr, lrd, lthd, lm = symbols('r rd thd m lr lrd lthd lm')
a, h, f, isp, g0, mu = symbols('a h f isp g0 mu')

dr = rd
drd = -mu/r**2 + h*sin(a)*f/m - r*thd**2
dthd = r*h*cos(a)*f/m + 2*rd*thd
dm = -h*f/isp/g0
```

2. Define the Hamiltonian.

```
H = dr*lr + drd*lrd + dthd*lthd + dm*lm - h
print('H =', H)

>>> H = -f*h*lm/(g0*isp) + lr*rd
+ lrd*(f*h*sin(a)/m - mu/r**2 - r*thd**2)
+ lthd*(f*h*r*cos(a)/m + 2*rd*thd) - h
```

3. Inspect the Hamiltonian for the thrust angle control variable, in this case a. It is clear that $H_a = (l_u \sin a + l_v \cos a) \cdot \ldots$, so H is maximized if $a = \arctan l_v / l_u$.

```
H = H.subs(a, atan2(lthd, lrd))
print('H =', H)

>>> H = -f*h*lm/(g0*isp) + lr*rd
+ lrd*(f*h*lthd/(m*sqrt(lrd**2 + lthd**2)) - mu/r**2 - r*thd**2)
+ lthd*(f*h*lrd*r/(m*sqrt(lrd**2 + lthd**2)) + 2*rd*thd) - h
```

4. Find the switching function by taking the derivative of H with respect to h.

```
print('dHdh =', diff(H, h))

>>> dHdh = f*lrd*lthd*r/(m*sqrt(lrd**2 + lthd**2))
+ f*lrd*lthd/(m*sqrt(lrd**2 + lthd**2)) - f*lm/(g0*isp) - 1
```

[1] `https://www.sympy.org`

5. Find the derivatives of the costates by taking minus the derivatives of H with respect to the respective state.

```
print('dlr =', -diff(H, r))
print('dlrd =', -diff(H, rd))
print('dlthd =', -diff(H, thd))
print('dlm =', -diff(H, m))

>>> dlr = -dt*f*h*lrd*lthd/(m*sqrt(lrd**2 + lthd**2))
- dt*lrd*(2*mu/r**3 - thd**2)
dlrd = -dt*lr - 2*dt*lthd*thd
dlthd = 2*dt*lrd*r*thd - 2*dt*lthd*rd
dlm = dt*f*h*lrd*lthd*r/(m**2*sqrt(lrd**2 + lthd**2))
+ dt*f*h*lrd*lthd/(m**2*sqrt(lrd**2 + lthd**2))
```

This yields all the algebraic equations required to solve the optimal control problem.

5.1.2. Cost Function

The implementation of the cost function is dependent on the use case. In general, the cost function can be implemented simply as part of the Hamiltonian, i.e. the overall cost function is subtracted from H.

The more complex the cost function is, the more difficult it becomes to find an algebraic expression for the parameters derived above. As such, at this stage only a simple cost function have been considered, containing a weighted sum of only the transfer time and spacecraft mass. Once this approach is shown to work, it can be improved by adding a more elaborate cost function.

However, it already becomes apparent at this stage that finding a useful cost function is not necessarily trivial. While simple cost functions such as dry mass or travel time can be simply defined, combining costs requires finding, for example reasonable weights for a weighted average. In general, it can be considered that such a weighted average can be set up using actual costs based on previous missions, for example an estimate of cost per day of transfer time and a cost per kg of dry mass. At this stage of mission design, these estimates can be very inaccurate, meaning this optimization procedure yields a false optimum. This discussion will be continued when considering initial guesses.

5.1.3. Differential Equation Function

The second step entails substituting the derived equations into a format that the BVP solver can handle. This requires a short introduction of the `solve_bvp` algorithm in `scipy`.

The call signature is `solve_bvp(fde, fbc, t, q, p)` with `p` and other parameters optional. The function `fde` is the differential equation function. This function takes the time span `t`, state vector `q` including costates, and outputs the derivatives of the states. Additionally and optionally, it takes the static parameter `p`. The majority of the work lies in writing out the required functions; a call of `solve_bvp` then simply returns the solutions of the BVP.

An example without the static parameter would be the case for a boundary value problem where the final time is known. The derivatives of the costates and the switching function can then simply be copied in.

1. Unpack the state vector `q`.

```
from numpy import sqrt, sin, cos, atan2, where

def fde(t, q):
  r, rd, thd, m, lr, lrd, lthd, lm = q
```

2. Implement the angle control a and throttle control h as calculated previously, as function of the states and costates. This means h is set to 1 if the switching functions is positive, and to 0 otherwise. Note that the `where` clause contains the switching function; it sets the throttle to 1 if the function is positive, and to 0 otherwise.

```
  a = atan2(lthd, lrd)
```

```
h = where(f*lrd*lthd*r/(m*sqrt(lrd**2 + lthd**2))
+ df*lrd*lthd/(m*sqrt(lrd**2 + lthd**2))
- f*lm/(g0*isp) - 1 > 0, 1, 0)
```

3. Write out the derivatives of the states, as defined previously, and of the costates, as calculated previously.

```
dr = rd
drd = -mu/r**2 + h*sin(a)*f/m - r*thd**2
dthd = r*h*cos(a)*f/m + 2*rd*thd
dm = -h*f/isp/g0

dlr = -f*h*lrd*lthd/(m*sqrt(lrd**2 + lthd**2))
- lrd*(2*mu/r**3 - thd**2)
dlrd = -lr - 2*lthd*thd
dlthd = 2*lrd*r*thd - 2*dt*lthd*rd
dlm = f*h*lrd*lthd*r/(m**2*sqrt(lrd**2 + lthd**2))
+ f*h*lrd*lthd/(m**2*sqrt(lrd**2 + lthd**2))

return [dr, drd, dthd, dm, dlr, dlrd, dlthd, dlm]
```

The static parameter vector `p` can be used for any unknown variables that do not change over time. Most importantly, this can be used to change the boundary value problem into a free time scenario. In this case, one of the variables in `p` is dt, a scaling factor for the time step. If we present the boundary value problem as ranging from $t = 0$ up to $t = 1$, and multiply each derivative with dt, we have implicitly defined a free final time of dt.

It must be noted that `solve_bvp` can freely change the parameters in `p` to any value, including zero and negative values. In order to prevent any numerical issues, there are two important considerations. First, a good initial guess must be given for `p` as well as for the states and costates. This is the most major drawback of indirect methods. Secondly, one can restrict dt to strictly positive numbers above 1, for example by setting $dt = \sqrt{p_0^2 + 1}$, which is positive and differentiable everywhere.

1. Unpack the state vector `q`.

```
from numpy import sqrt, sin, cos, atan2, where

def fde(t, q):
  r, rd, thd, m, lr, lrd, lthd, lm = q
  dt = sqrt(p[0]**2 + 1)
```

2. Implement the angle control a and throttle control h as calculated previously, as function of the states and costates. The difference here is that all terms in h are multiplied by dt.

```
a = atan2(lthd, lrd)
h = where(dt*f*lrd*lthd*r/(m*sqrt(lrd**2 + lthd**2))
+ dt*df*lrd*lthd/(m*sqrt(lrd**2 + lthd**2))
- dt*f*lm/(g0*isp) - 1 > 0, 1, 0)
```

3. Write out the derivatives of the states, as defined previously, and of the costates, as calculated previously. Again, all derivatives are now multiplied by dt.

Additionally, it was intended that the static optimization variables, such as thrust and I_{sp}, could also be encoded in `p` in a similar manner.

5.1.4. Boundary Condition Function

The `solve_bvp` routine requires a boundary condition function, `fbc`, which takes the calculated states at the initial and final time, and returns for each the difference between the calculated state and the required state. Here, the number of degrees of freedom becomes important.

When considering the simplest 2D case with states x, y, u, v, the boundary condition function should return $2 \cdot 4 = 8$ values: one extra costate for each state. If any of these states is free, for example the final value of u is free, then instead set λ_u to zero.

Additionally, when including dt as a static variable, the boundary condition function needs to return an additional value, or the problem would be underconstrained. For time scaling, this value is well defined: the function needs to return the complete Hamiltonian at the final time, which must be set to zero.

Some examples are given.

- A fixed time BVP, with initial states $(...0)$, final states $(...f)$, and target states $(...t)$. The u component is free at both boundaries: as such the costate is returned to be set to zero, rather than the state itself.

```
def fbc(q0, qf):
  x0, y0, u0, v0, lx0, ly0, lu0, lv0 = q0
  xf, yf, uf, vf, lxf, lyf, luf, lvf = qf
  return [x0 - x0t,
          y0 - y0t,
          lu0,
          v0 - v0t,
          xf - xft,
          yf - yft,
          luf,
          vf - vft]
```

- A free time BVP. It is necessary to calculate the throttle control at the final state in order to determine the Hamiltonian.

```
def fbc(q0, qf, p):
  x0, y0, u0, v0, lx0, ly0, lu0, lv0 = q0
  x, y, u, v, m, lx, ly, lu, lv, lm = qf
  dt = sqrt(p[0]**2 + 1)

  h = where(dt*lu*ly/sqrt(lv**2 + ly**2)
  + dt*lv**2/sqrt(lv**2 + ly**2) - 1 > 0, 1, 0)

  H = dt*h*lu*ly/sqrt(lv**2 + ly**2)
  + dt*h*lv**2/sqrt(lv**2 + ly**2) + dt*lx*u + dt*ly*v - h

  return [x0 - x0t,
          y0 - y0t,
          lu0,
          v0 - v0t,
          x - xft,
          y - yft,
          lu,
          v - vft,
          H]
```

5.1.5. Elimination of Variables

At this stage, it may become apparent that variables are unneeded. For example, considering a 2D transfer problem from one circular orbit to another. When using a Cartesian coordinate system, there are four variables x, y, u, v.

All four variables are relevant for the derivatives: the derivatives of x and y are u and w, and the derivatives of u and w depend on gravity, which depends on position.

However, for the boundary conditions, some issues arise. It is not relevant at which point in the orbit the transfer starts and ends, just that the shape of the initial and final orbit is as required. Therefore, there are only 3 boundary conditions, while there are 4 variables.

1. Radius $r = \sqrt{x^2 + y^2}$ equals required radius.
2. Speed equals circular orbital speed $\sqrt{\mu / r}$.
3. Velocity direction is perpendicular to radius vector.

There is also no costate directly related to the position in the orbit, which could be set to zero, so the system cannot be solved using a BVP solver.

Instead, it is possible to use a polar coordinate system with state $r, \theta, \dot{r}, \dot{\theta}$. When implementing the dynamics, it becomes clear that the inclusion of θ as a state is problematic. First, θ is free at both boundary conditions, which can be solved by setting λ_θ to zero instead. However, we see that $\dot{\lambda}_\theta$ is also zero, meaning that λ_θ is zero everywhere.

The reason for this is that θ is not relevant anywhere in the dynamics. This may not be initially obvious from looking at the dynamics equations, as θ does appear, but it makes sense intuitively: θ is irrelevant for the dynamics dependent on gravity, and irrelevant for the dynamics dependent on control. If we were to include, for example, lunar perturbations, which is dependent on θ, then θ would appear in the dynamics and $\dot{\lambda}_\theta$ would not be zero everywhere. This issue appears similarly for λ_{c_e}, using the general derivation in chapter 4.

Additionally, the reason that the system becomes numerically unsolvable, even though θ may just be 'ignored', is that the inclusion of a zero-everywhere vector leads to singular Jacobian, such that the direction in which to converge is undefined.

To conclude, including θ makes this problem numerically unsolvable, even though encoding the full position of a spacecraft seems logical when coming from a Cartesian coordinate system.

5.1.6. Initial Guess and Coordinate Systems

With the derivatives and boundary conditions implemented, `solve_bvp`, like all boundary value problem solvers, needs an initial guess to start. Depending on the accuracy of this initial guess, the solver may take longer to converge, or fail to converge at all.

The difficulty of determining a sufficiently close initial guess depends on the problem and on the coordinate system used.

1. Cartesian

   A Cartesian coordinate system is simple to implement in both 2D and 3D. From here on, the variables added for 3D systems are enclosed in parentheses.

   $$\boldsymbol{r} = \begin{bmatrix} x \\ y \\ (z) \end{bmatrix} \tag{5.1}$$

   $$\boldsymbol{s} = \begin{bmatrix} u \\ v \\ (w) \end{bmatrix} \tag{5.2}$$

   The Cartesian coordinate system is especially difficult to obtain a good initial guess without much experience with designing trajectories. The most important reason for this is that Cartesian coordinates vary wildly throughout the transfer. The values switch signs often, and it is not possible to place upper and lower bounds on individual state variables: for example, an absolute $|x|$ coordinate within the radius of Earth is fine if $|y|$ is outside the radius of Earth, but not otherwise.

   Cartesian coordinates are not necessarily unbounded: each state variable can be made dimensionless by dividing by a certain constant, e.g. dividing positional values by the radius of Earth, such that the expected possible values lie within a previously determinable range. However, this does not affect the difficulty of finding a good initial guess.

2. Polar/Spherical

   A spherical coordinate system can be implemented in 3D, or as a polar system in 2D.

$$\boldsymbol{r} = \begin{bmatrix} r \\ \theta \\ (\phi) \end{bmatrix} \tag{5.3}$$

$$\boldsymbol{s} = \begin{bmatrix} \dot{r} \\ \dot{\theta} \\ (\dot{\phi}) \end{bmatrix} \tag{5.4}$$

   In a two dimensional transfer problem, a polar system is more suitable for finding an initial guess. For example for circular-to-circular transfers, r is always positive and easily bounded; as shown θ may be eliminated, $\dot{r}$ goes smoothly from zero to positive to zero, and $\dot{\theta}$ is always positive and easily bounded.

   In this case, a good initial guess would be a linearly increasing r, a half-period sinusoidal $\dot{r}$, and a linearly decreasing $\dot{\theta}$.

   However, once the orbital position becomes important, θ's periodicity may become an issue. Either θ must be bound between 0 and 2π, becoming discontinuous, or the boundary condition function must acknowledge periodicity.

   Furthermore, the dynamics become more complicated. This is not an issue by itself, but the more complicated the dynamics are, the more complicated the derivatives of the costates become. While these are derivable automatically, it may become very difficult to determine an optimal control law for the control vector from visual inspection of the Hamiltonian.

   For a polar coordinate system, the dynamics are as follows[2].

$$\dot{\boldsymbol{q}}_{2D} = \begin{bmatrix} dr/dt \\ d\theta/dt \\ d\dot{r}/dt \\ d\dot{\theta}/dt \end{bmatrix} = \begin{bmatrix} \dot{r} \\ r\dot{\theta} \\ \ddot{r} - r\dot{\theta}^2 \\ 2\dot{r}\dot{\theta} + r\ddot{\theta} \end{bmatrix} \tag{5.5}$$

   Here, $\ddot{r}$ depends on gravity and thrust, and $\ddot{\theta}$ depends on thrust only.

   For 3D implementations, the last variable ϕ remains bounded, but becomes more difficult to present an initial guess for. Furthermore, in 3 dimensions in general, it is beneficial to determine the optimal control law in terms of velocity costate vector as shown in section 2 e.g. $\boldsymbol{\alpha}_{3D} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T = \begin{bmatrix} a_r & a_t & a_n \end{bmatrix}^T$. Here $(_r)$, $(_t)$ and $(_n)$ stand for radial, tangential and normal, respectively, which is customary for disturbance vectors in a spherical coordinate system. The control vector in terms of trigonometric functions would become

$$\boldsymbol{\alpha}_{3D} = \begin{bmatrix} h\sin\alpha \\ h\cos\alpha\cos\beta \\ h\cos\alpha\sin\beta \end{bmatrix}$$

   for which the optimal control law would be more difficult to decipher.

3. Kepler Elements

   Alternatively, the dynamics state of $\boldsymbol{r}$ and $\boldsymbol{s}$ can instead be given in terms of the Kepler elements. The distinction between position variables and velocity variables is lost, and the gravitational influence is no longer considered part of the dynamics, but implicitly included in the coordinate system.

   The Kepler elements are semi major axis a, eccentricity e, inclination i, longitude of ascending node Ω, argument of periapsis ω, and any anomaly, e.g. true anomaly ν.

[2] `http://pioneer.netserv.chula.ac.th/~anopdana/211/24rtheta.pdf`

$$\boldsymbol{q} = \begin{bmatrix} a \\ e \\ (i) \\ (\Omega) \\ \omega \\ \nu \end{bmatrix} \tag{5.6}$$

In this system, all states but ν are constant throughout time if thrust and external disturbances are zero. If thrust and disturbances are considered, all vary slowly throughout time. Furthermore, a and e are bounded and positive, the other variables are bounded. This further reduces the difficulty of finding a good initial guess.

Furthermore, this system is useful when considering a transfer between orbits, where the actual position in the initial and final orbit is irrelevant. An orbit is fully described by taking all variables except ν. In a 2D case between elliptical orbits, and in any 3D case, the anomaly at any point in time is relevant during the transfer, so ν cannot be eliminated; instead, ν is free at each boundary so λ_ν is set to zero.

However, ν is a periodic variable with the same issues as before. Furthermore, the variables become singular when the orbit approaches circularity or equatoriality, which were previously considered simplifications. Because of this, the dynamics are usually immediately given in terms of equinoctial elements in literature.

Simplified dynamics are available, however, which are used to derive Edelbaum's approximation [25]. In this formulation, only circular orbits are considered, as well as only tangential and out-of-plane thrust; radial thrust is considered safe to exclude. These dynamics are as follows, derived from the mean anomaly M.

$$V = \sqrt{\mu / a} \tag{5.7}$$

$$\begin{bmatrix} \dot{a} \\ \dot{i} \\ \dot{\Omega} \\ \dot{\omega + M} \end{bmatrix} = \begin{bmatrix} 2a\Delta_t / V \\ \Delta_h \cos(\omega + M) / V \\ \Delta_h \sin(\omega + M) / V / \sin(i) \\ V / a - \Delta_h \sin(\omega + M) / V / \tan(i) \end{bmatrix} \tag{5.8}$$

Note here that Δ indicates the acceleration in radial $(_r)$, normal $(_n)$, or tangential $(_t)$ direction, such that $\boldsymbol{\alpha} = \begin{bmatrix} \Delta_r & \Delta_t & \Delta_n \end{bmatrix}^T$.

4. Equinoctial Elements

Equinoctial elements are a variation of parameters on the Kepler elements, in the same way that Rodriguez parameters extend upon Euler angles for rotation. Using equinoctial elements, a singularity only occurs at circular, equatorial, retrograde orbits, which is sufficient for almost all applications. That is to say, singularities still exist, but have been 'postponed' to a state that is rarely reached, at the cost of more complex dynamics equations [25].

$$\boldsymbol{q} = \begin{bmatrix} p \\ f \\ g \\ h \\ k \\ L \end{bmatrix} = \begin{bmatrix} a(1 - e^2) \\ e\cos\omega + \Omega \\ e\sin\omega + \Omega \\ \tan i/2 \cos\Omega \\ \tan i/2 \sin\Omega \\ \Omega + \omega + \nu \end{bmatrix} \tag{5.9}$$

This system is relevant only for 3D orbits. Like Kepler elements, all but one are constant for unperturbed motion, in this case L, and all vary slowly through perturbed motion.

However, the dynamics become very involved [25].

$$s^2 = 1 + h^2 + k^2 \tag{5.10}$$
$$w = 1 + f\cos L + g\sin L \tag{5.11}$$
$$r = p/w \tag{5.12}$$
$$q = \sqrt{p/\mu} \tag{5.13}$$

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{p} \\ \dot{f} \\ \dot{g} \\ \dot{h} \\ \dot{k} \\ \dot{L} \end{bmatrix} = \begin{bmatrix} 2pq\Delta_t/w \\ q(\Delta_r \sin L + ((w+1)\cos L + f)\Delta_t/w - (h\sin L - k\cos L)g\Delta_n/w) \\ q(-\Delta_r \cos L + ((w+1)\sin L + g)\Delta_t/w + (h\sin L - k\cos L)g\Delta_n/w) \\ qs^2\Delta_n/2/w \cdot \cos L \\ qs^2\Delta_n/2/w \cdot \sin L \\ w^2\sqrt{\mu p}/p^2 + q/w \cdot (h\sin L - k\cos L)\Delta_n \end{bmatrix} \tag{5.14}$$

Finding the optimal control law with respect to becomes a very involved process. Relevant matrices of costate derivatives are presented in *Applied Numerical Astrodynamics* by A. Kechichian (2018) [25], however these are all derived for specific use cases. It is not thought feasible to automatically derive the costate equations for an arbitrary cost function.

5. Quaternions

   The use case for equinoctial elements is immediately reminiscent of the Rodriguez parameters in spacecraft attitude control. As such, it is of interest to see if quaternion representations are applicable for orbital mechanics as well. It appears there are a few, and only very recent, papers on this, e.g. Sapunkov and Chelnokov [51] and Libraro [34].

   It is expected that the use of quaternions would significantly ease the process of finding a good initial guess. While the individual components do switch sign, their implementation would be nonsingular, continuous, non-periodic, and bounded between -1 and 1. In fact, if the implementation of an angular position is similar to the implementation of attitude, the allowed magnitude of any state would be 1, restricting the possible values to the unit hypersphere.

   Although quaternions would make deriving nonsingular dynamics much simpler, using it would probably require work far beyond the scope of this research.

5.1.7. Initial Guess for Costate Values

The generation of initial guesses for the costates is more troublesome still than for the states. In most cases, not even the boundary conditions of the costates are known. They are only known when the state itself is free; in which case it is also possible that this state may be eliminated.

It appears that the only consistent strategy is to use the costates of previously calculated results for similar trajectories. However, these are not available when redoing a calculation with different values for e.g. thrust or I_{sp}, for which the trajectory changes drastically. In this case, it is better to converge to the desired trajectory iteratively, i.e. slowly changing the static parameters from a known case to the desired case, recalculating the trajectory at every step. Because of this, it is no longer feasible to consider a joint optimization scheme as a single boundary value problem.

An alternative approach reduces the solving procedure to single-shooting algorithm, which removes the need for a algorithm such as `solve_bvp`.

In this case, we consider the initial values of the costates as optimization parameters, along with static variables such as such as expected transfer time.

Using the initial states and costates, and knowing the optimal control law depending on the current states and costates, the system is integrated until the final time, and the difference between the required final state and actual final state is returned. This means that the problem is now posed as an (iterative) optimization problem with the initial values of the costates and the transfer time as unknowns.

As such, the costate values only have to be guessed at the initial time, rather than throughout the transfer, reducing the potential degrees of freedom. However, the initial guesses still need to be guessed accurately enough for the system to converge.

5.1.8. Further Issues

It was considered that static control variables, such as values for thrust and I_{sp} could be included in the parameter `p` . However, this leads to several problems.

Firstly, any value in `p` is by design allowed to vary without bounds. While it is possible to restrict values to certain ranges, as was done in the case of dt, it is not clear if adding such boundaries reduces the program's chance of convergence.

Secondly, for each static control variable, `solve_bvp` requires one and only one additional boundary condition. There are multiple expected possibilities, and it has not been possible to confirm, which, if any, is correct. For example, the derivative of the Hamiltonian w.r.t. the static control variable must be zero: this has not been verified to be a sufficient boundary condition. Furthermore, the second derivative of the Hamiltonian w.r.t. the static control variable must be positive: this is not implementable as a single-valued condition. It has also not been verified if it is sufficient to implement this boundary condition at the final boundary, as was the case for dt, or if in this case the boundary should instead be set at the initial time, or at both.

Thirdly, and most importantly, this method remains very sensitive to the initial guess. This is dependent on the choice of coordinate system, as shown above, but remains an issue for any coordinate system.

Fourthly, an integrated approach would be very sensitive to the objective function, which is based entirely on estimates at the phase of design where such a system is used. This means that any potential design would be very sensitive to parameters with a low confidence in accuracy.

5.1.9. Conclusion

The most important issue lies with the need for a good initial guess, such that an automated approach cannot be guaranteed to be stable, and will need manual intervention, so the advantage of an automated approach is lost.

The need for an initial guess can be solved by an iterative approach, which again removes the advantage of having a single combined optimization problem.

In conclusion, for any automated system using boundary value problem solvers, it has not been possible to present a solution with benefits over an iterative scheme with existing low thrust optimization methods.

5.2. Implementation using an OCP Solver

The OCP solver `gekko` is one of the many freely available dynamic system solvers. Like the rest, it works by transforming the dynamic system into an optimization problem, which is then solved internally; in `gekko`'s case using `ipopt`.

It is considered that the use of `gekko` is theoretically equivalent to the use of `scipy`, while reducing the chance of practical issues. That is to say, if the implementation fails in `gekko` as well, it would support the conclusions on the theoretical nature of the encountered issues, rather than e.g. programming mistakes.

The conceived use of `gekko` is as follows.

1. Given: payload mass, mass function from parameters (thrust, I_{sp}, ...), cost functions from parameters (thrust, I_{sp}, ...), cost functions from trajectory (time, radiation, ...), dynamics functions.
2. Set the states as State Variables.
3. Set the dynamic controls as Control Variables.
4. Set the static controls as Fixed Variables.
5. Set the dynamics as Equations.
6. Set the boundary conditions as Equations.
7. Set the cost function as Objective.
8. Run the OCP solver.

5.2.1. System Definition

It is simply possible to implement variables as gekko objects such as State Variables (e.g. velocity), Control Variables (e.g. thrust level), or Fixed Variables (e.g. I_{sp}), without assigning specific values, and they will be handled as such. State variables will be integrated over time, control variable will be varied within defined bounds, and fixed variables will be changed to meet the objective, but will be constant in time.

An introduction to gekko's syntax is given using some examples.

```
from gekko import GEKKO
model = GEKKO()
a = model.CV(lb=0, ub=2*model.pi) # With lower and upper bound
x = model.SV()
dt = model.FV(lb=1, ub=10000)
dt.STATUS = 1 # Allows varying
```

The dynamics are simply implemented as equations. Derivations of e.g. costates, where necessary, are done internally using sympy as well. For example:

```
model.Equation(x.dt() == u) # dt means derivative
model.Equation(u.dt() == -x*mu/(x**2 + y**2)**1.5 + h*model.cos(a)*f/m)
```

Boundary conditions are set using either a fix constraint or an equation.

```
model.fix(x, 0, x0)
model.Equation(y*[1, 0, 0, 0, 0] == y0)
```

5.2.2. Issues

Similar to the implementation is scipy, this implementation returns several issues.

There are two implementation issues with the boundary conditions, namely that the fix constraints lead to warnings that the constraint fails, while the equation constraints lead to the problem being overconstrained. However, in either case, the simulation does run.

Because of this, gekko fails to converge at all. The main issue with intermediate solvers such as gekko lies in their use as black box solvers. This means that, whenever something is wrong, the user needs to guess how the errors relate to the input. While it is not feasible to determine how these issues occur in this case, it is logical to assume that it is the same as in the scipy implementation. It is considered that this is, again, due to the lack of a good initial guess, such that the problem is initialized randomly, and the chance of convergence is random as well. Because of this, the use of gekko has no computational advantage over deriving and solving the BVP using other methods.

The use of gekko is no longer considered useful for this study, due to its black box nature and its equivalence in functionality to a BVP solver as discussed above.

5.3. Implementation using a Trajectory Optimizer

ESA's pykep is a full solver for trajectory optimization problems. In that sense, pykep is even more so a black box than a general optimal control problem solver. However, pykep is used in practice for the preliminary design of space missions, in contrast to most other available software.

Defining and solving an low thrust transfer problem can be done in few lines of code. However, pykep's documentation is incomplete and outdated, so it may be useful to present an overview of its usage, as derived from example code and experimentation.

In general, use of pykep follows.

1. Set up a trajopt trajectory optimization object.
2. Load this as a pygmo problem object. The pygmo project is ESA's system for generic optimization, and used as backend in pykep.
3. Set up the optimization algorithm, and determine the optimization parameters.
4. Run the algorithm.

The simplest working case has been found to be as follows.

```
import pykep as pk
import pygmo as pg
traj = pk.trajopt.direct_pl2pl()
grad = pk.examples.add_gradient(traj, with_grad=True)
prob = pg.problem(grad)
pop = pg.population(prob, 1)
algo = pk.examples.algo_factory('slsqp')
pop = algo.evolve(pop)
print(prob.feasibility_x(pop.champion_x))
```

This contains several steps.

1. A `trajopt` object is instantiated. This is essentially a function, taking parameters such as departure time and initial and final states, and returning values such as feasibility.
2. Many solvers for nonlinearly constrained problems require the problem to present a gradient, including `slsqp` and `ipopt`. This can be done as a finite difference scheme using the `add_gradient` function from the example module. Other solvers, such as `nlopt` or heuristic methods do not use the gradient.
3. The function, with gradient, is converted to a `pygmo` problem object.
4. A population of solutions is generated randomly. For deterministic solvers, the population size can be 1, while for heuristic solvers this can be as many as required.
5. An algorithm must be instantiated. The simplest way to do this is using the `algo_factory` function from the example module. Using `slsqp` is sufficient for Earth to Mars transfers or few-revolution orbits in general. For many revolutions, this is not sufficient; in which case one can use a heuristic method. For example, one can use differential evolution, or use `slsqp` within a monotonic basin hopping (MBH) scheme [68][11]. Both options are activated as shown below.

   ```
   algo2 = pg.algorithm(pg.algorithms.de())
   algo3 = pg.algorithm(pg.algorithms.mbh(
       algo=pk.examples.algo_factory('slsqp')
   ))
   ```

6. Finally the problem is solved by evolving the population. The found optimum is not always a feasible trajectory; for functions such as `slsqp`, this depends on the initial guess; for heuristic methods, this is random. The more revolutions, the less likely a feasible optimum is found.

5.3.1. Custom Problem Object

As is, `pykep` has a number of problem objects available for the optimization of single thrust trajectories. It is conceivable to write a custom problem objects for dual thrust propulsion, coupled optimization of systems design, or both. However, there are a number of expected issues.

Firstly, while `pykep` does allow for customized optimization problems, each existing problem is built around direct and indirect transcription of trajectory 'legs': for each of these legs, `pykep` internally allows only a single value for thrust and I_{sp}; and the number of legs cannot be changed during optimization.

Secondly, `pykep` currently has no support for optimizing trajectories up to escape velocity, with is especially relevant for the cubesat design in question. As such, a possible design trajectory would be limited to separating the thrust, for example, high thrust for an escape trajectory, and low thrust for the interplanetary transfer.

Thirdly, `pykep` has no support for cost functions depending on trajectory, which would be especially relevant to consider the transfer through radiative zones. To make this approach useful, it would be necessary to add this functionality, but generating a radiative model from scratch is beyond the scope of this research.

As such, writing a useful problem object may require altering the internal workings of the program. Changing the internal workings of `pykep`, while allowed under its license, would be beyond the scope of this thesis, considering the size of the program and the lack of documentation. Lastly, even if these issues are avoided, `pykep` remains fully dependent on the accuracy of an initial guess, so using `pykep` in an automated scheme would remain as unstable as the implementation attempts presented before.

5.3.2. Manual Iteration

One way to avoid the aforementioned issues is to use the assumption mentioned before, which is an assumption which may be necessary to include even when implementing a custom problem object, namely that high thrust is used for Earth escape, and low thrust is used for the interplanetary transfer. This approach assumes that the high thrust system must be used up completely first; and the low thrust system afterwards.

Such an approach would work as according to Figure 3.3.

1. Given an iteration (or for the first iteration, a guess) for a high level systems design.
2. Use existing `pykep` routines to find an optimal Earth to Mars trajectory.
3. Using this trajectory, find an optimal high thrust trajectory for Earth escape, using high thrust calculations, which is simply implementable.
4. Iterate between the two trajectories until convergence.
5. Using the new total trajectory, iterate the systems design until convergence. For example, this can be a change in propellant mass or radiation shielding.
6. Iterate the systems design for the given trajectory until it converges. This is necessary because, e.g. a required increase in propellant mass causes a further increase in propellant mass, due to the larger structural mass. This is often dubbed the "snowball efect".
7. With the new systems design, restart from 2. to find a new trajectory.

This entire sequence is repeated until, as is expected, the difference between iterations becomes increasingly smaller. The number of iterations depends on the required accuracy; for this study, it was opted not to optimize further beyond the used accuracy of the mass budget, which is in the order of grams.

At this point, the design can either be accepted, or be changed. For example, it may be considered that a different design option is preferable after all. In this case, the design option can simply be replaced, and the process can be restarted; after which the design will iterate towards a new optimum, which will be either better or worse. If iterations are fast enough, these changes can quickly be evaluated or reversed.

Lastly, it must be considered that the level of detail can be increased between iterations. This allows for a very high level initial guess, which is easier to implement. Increasing the level of detail midway allows multiple people to work on different aspects, in the spirit of concurrent design.

5.4. Conclusion of Implementation Attempts

It has not been proven possible to present an automated solution to jointly optimize dual thrust trajectories and systems, which does not rely on manual guessing or iteration.

It is still possible that such an approach exists, either using methods that have not been discussed, or after considering the discussed methods more carefully. However, with the current knowledge and resources on this topic, it is not thought to be feasible to continue working on these attempts.

As such the attempts to find an integrated approach will be abandoned. A manual approach using existing features in `pykep` will be used instead.

This answer the research question posed last: *currently, the most useful implementation relies on manual iteration using existing* `pykep` *functionality.*

6

Resultant Design

This chapter describes the design at various iterations of the design process. It starts with considerations for an initial guess, as well as a general overview of the design options. Secondly, a number of milestones are shown. Lastly, the final considered iteration is presented in detail.

6.1. Overview

The manual implementation using `pykep` is an iterative approach. Similar to the initially proposed implementation in `scipy`, it starts from an initial guess, after which the trajectory and systems design are continuously iterated.

The initial guess must first be generated. This is done based on preliminary design option choices, however, because it is expected that these options change, it is not necessary to set these design options as final. The initial guess is described below.

After this, the first milestone lies in finding a feasible trajectory using the on the initial guess. Once this has been found, further trajectory iterations are generated faster and more consistently.

Then, the next milestone is reached when a feasible combined design is found, in which both the design budget and trajectory meet the performance requirements.

A next milestone is considered for a combined design, using COTS components. At this point, the requirements are met, and the design problem is considered solved. However, the design can still be iterated further. Additionally, design options can be switched and iterated to investigate their effect on the design.

For this thesis, it is considered sufficient to optimize one step further beyond the first design that meets all requirements. This entails performing one design option switch, if required, when all requirements are met, and then iterating until convergence.

6.2. Initial Guess

For the initial guess, there are four major decisions on design options. Note that this design option being chosen at this stage does not mean that this option is kept throughout the design process: it is only used as an initial guess, and while it is likely that it will remain the chosen solution, it may change if the requirements cannot be met.

The decisions that are made in advance, as initial guess, are as follows:

1. The type of thrusters for the low and high thrust system respectively.
2. The choice of launcher, and thus initial parking orbit, which significantly affects the ΔV requirements.
3. The choice of actuation for the attitude determination and control (ADC) system.
4. The choice of how the solar array will be placed upon the spacecraft.

Smaller decisions will be detailed when relevant. However, as mentioned before, derived requirements are expected to change rapidly, such that such a trade off may apply to only a single specific iteration.

	F [N]	I_{sp} [s]	Reignitable	TRL
Solid	50	250	✗	9
Liquid monopropellant	1	250	✓	7
Liquid bipropellant	1	290	✓	4

Table 6.1: Typical performance for the high thrust options under consideration.

	F [N]	I_{sp} [s]	TRL
Cold gas	0.1	70	9
Resistojet	0.03	100	-
- Electric	-	-	4
- Solar thermal	-	-	3
Solar sail	-	-	3
Ion thruster	0.005	3000	7
Hall thruster	0.05	1800	6
Electrospray	0.0015	1800	5
Pulsed plasma	0.0005	1000	5
Vacuum arc	0.000002	3000	4
Radio frequency	0.001	1000	4

Table 6.2: Typical performance for the low thrust options under consideration.

Note that, for the trajectory, the initial guess is randomly selected by `pykep`, from a range between pre-programmed lower and upper bounds. Within this range, the trajectory optimization solver is deemed likely to converge by the authors of `pykep`. However, the choice remains fully random within this range and may still fail to converge.

6.2.1. High Thrust

High thrust options are taken from Mani et al. [36], Silva et al. [53], and Casini et al. [10], as summarised in Syaifoel [61].

General options are in Table 6.1. Note that for COTS components, other variables such as mass, volume, etc. may vary drastically from manufacturer to manufacturer. The TRL relates to miniaturized systems only.

Each of these options is eligible be considered for an iteration: a considerable array of COTS options exist. Hydrazine options are not considered, nor widely available as COTS cubesat components. Solid propellant thrusters are still under consideration, even though they are not reignitable: a single burn may be sufficient if thrust and I_{sp} or propellant mass are large enough. For the initial consideration, liquid bipropellant is considered leading, because of the highest possible I_{sp}.

6.2.2. Low Thrust

These options are taken from Mani et al. [36], Silva et al. [53], and Casini et al. [10], and as summarised in Syaifoel [61], as well. Typical values are presented in Table 6.2. Note that, again, these are typical values considered for generating an initial idea; the performance of actual existing COTS components varies drastically.

Not all options are represented in the available COTS components. Very low TRL components are not available as technology demonstrations either, as such they will not be considered further.

At this stage, the highest I_{sp} is considered leading; this may change later. As such, ion thrusters are considered for now.

6.2.3. Launch Orbit

For launch orbits, only the options are considered for which piggybacking is commonly available. Expected launch costs are also considered, which can be estimated albeit without verification; as such the selection is based mainly on ΔV requirements. From commercial availability online, it appears that most launch options target Low Earth Orbit (LEO), Sun Synchronous Orbit (SSO), and Geostationary Transfer Orbit (GTO).

It appears that LEO and GTO are the most commonly available piggyback options for a commercial or educational cubesat launcher. The other orbits do occasionally allow a dual launch, which may be more cost effective if the opportunity arises, especially a Lunar or Mars injection. However, in general, these options are considered too costly.

For the initial design, a GTO option is considered the most economical for both the ΔV budget as well as the launch cost budget.

6.2.4. Momentum Dumping

Integrated ADC components are available, however, all are designed for maneuvering in Earth orbit. As such, most rely on magnetorquers to dump any momentum stored in the reaction wheels. One option to resolve this is to include separate thrusters.

Alternatively, it may be useful to use the main thrusters to dump momentum. This will require the spacecraft to have multiple thrusters or thrusters with vectoring capabilities. In case of a low thrust interplanetary transfer, where a thruster might be firing almost constantly, this is considered to be the most effective solution, as it does not require any additional components.

The magnetorquer system remains useful for momentum dumping in Earth, furthermore it is an inexpensive, simple and well tested system. It is proposed that any ADC system will include a magnetorquer for momentum dumping in planetary orbits. The other parts of the ADC systems, such as star trackers and reaction wheels, are considered to work in interplanetary space.

6.2.5. Solar Array Configuration

The low trust system is expected to draw a considerable amount of power. There are some options in advance for the configuration of the solar array.

Firstly, a body mounted array is the simplest and lightest solution, and commonly used in cubesats. However, it is not expected to yield sufficient power.

Secondly, a body folded array uses a single deployable hinge for each panel. Such a system remains simple, and has been flown on several cubesats such as the Delfi C3. A single deployable solar panel can double the total solar area.

Lastly, more complex deployable systems increase the solar area further, at the cost of higher weight and complexity. Such systems are customly developed as in Mani et al. [36], allowing the solar array to be as large as required. However, for our purposes, it is necessary to constrain the solar array to a cubesat form factor, as well as to consider COTS options. Therefore, the input power becomes a constraining factor.

For this research, a singly deployable solar array is considered as an initial guess. While a solar thermal thruster is not considered any longer, the configuration of a solar concentrator array would be of interest for any future missions with solar thermal propulsion.

6.2.6. Initial Budget

An initial design guess is made. This is not meant to be a realistic or even feasible budget for the spacecraft; instead, the initial guess is needed to find an initial interplanetary trajectory, which in turn provides an estimate for the propulsion requirements, which in turn drives the design.

The initial design guess is as in Figure 6.1, a 16 unit cubesat such that each component is assigned one or more units with the maximum allowed unit mass of 1.33 kg. Note that this is not a requirement, but an initial guess as well: while the mass of a cubesat mass can exceed this value, and sometimes does, this is considered a reasonable starting point for the mass of individual components which fit in one unit.

The guess includes four cubesat units per propellant type. Furthermore, the solar array is considered to have a single body body mounted panel, three simply deployable panels, and one doubly deployable panel, such that the solar area is maximized while maintaining simplicity. Lastly, two thrusters are included for each level of thrust, such that the thrust vector leads through the center of the spacecraft.

The solar panels were configured such that all faces contain a panel, which can fold out with simple hinging. This yields 40 face units of solar panel. The communication system (COM) is at the solar face, i.e. towards the approximate direction of Earth from Mars, payload is shielded from the sun.

This design leads to the budget estimate as listed in Table 6.3. Power is currently only considered for electric thrust and spacecraft: this is not at all accurate, but it will change as iterations occur and better estimates become available per component. In this figure and table, acronyms are used to refer to the components as listed in Table 6.4.

As such THR and RAD are initially ignored. Additionally, as there are two types of propellant, it is convenient to use the designations M0, M1, and M2 instead of “wet mass” and “dry mass”. This is done as follows. M0 is the dry (not considering potential attitude thrusters) mass of the spacecraft, with electrical propellant and chemical propellant depleted. This is the mass at the final orbit. M1 is the mass of the spacecraft with chemical propellant depleted and a full electrical propellant tank. This is the mass at Earth escape. Lastly,

	Mass [kg]	Volume [U]	Power [W]
OBC	1.33	1	0
EPS	1.33	1	0
COM	1.33	1	0
ADC	2.66	2	0
SOL	1.60	-	-74
ETH	2.66	2	50
CTH	2.66	2	0
EPR	1.33	2	-
CPR	5.32	4	-
STR	2.25	-	-
TOT	25.13	16	-24

Table 6.3: Initial budget.

OBC	On board computer, the processing unit that handles commands, data storage and data transfer.
EPS	Electrical power system, the unit that regulates storage and distribution of power.
SOL	Overarching designation for the solar arrays.
COM	The communication system.
ADC	Attitude determination and control, considered as a singular unit.
STR	Overarching designation for the structural components.
THR	Overarching designation for the thermal control system.
RAD	Overarching designation for radiation shielding.
ETH	The electric thruster system.
CTH	The chemical thruster system.
EPR	Overarching designation for the storage and distribution of propellant for the electric thruster.
CPR	Overarching designation for the storage and distribution of propellant for the chemical thruster.

Table 6.4: Overview of acronyms used.

M2 is the wet mass of the spacecraft. This is the mass after launch. To summarize: M0 + M_{EPR} = M1, and M1 + M_{CPR} = M2.

6.3. Initial Feasible Trajectory

Using the presented initial guess, iterations are done as described. The initial goal is to achieve a successful trajectory calculation. As noted, finding a feasible trajectory at all is a difficult numerical problem by itself, especially when presented with a bad initial guess. However, once a trajectory is found, it can be used as starting point for following iterations; if these iterations differ only slightly, the likeliness of successfully finding a trajectory rises drastically, and the time it takes to return a result reduces as well.

The results of these first steps are listed in Table 6.5. Note that a failure indicates that the trajectory calculation simply fails to generate a feasible solution. How this is mitigated is listed in the table as well, done either by changing the problem object, or by changing the algorithm settings.

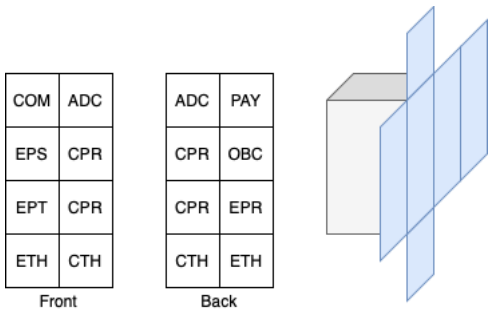

Figure 6.1: Layout of the initial guess, solar array in blue.

Iteration	Change	Result	Notes
1		Failure	Failure means failure to converge.
2	Change to `slsqp`	Failure	Different algorithms have different convergence properties.
3	Change to `ipopt`, increase number of segments	Failure	Higher number of segments more likely to converge.
4	Change to `slsqp`	Failure	
5	Change to `ipopt`, set initial guess to very high mass	Failure	Check to eliminate singularities.
6	Change to `slsqp`, reset mass	Failure	
7	Change to `ipopt` with MBH	Failure	Use monotonic basin hopping to find other local optima, new starting point found.
8	Change to `slsqp`, new starting point	Failure	Low constraint violations, increasing tolerance may resolve.
9	Increase tolerance	Success	Low accuracy: trajectory inconsistent by visual inspection.
10	Reduce tolerance	Success	Low accuracy: trajectory inconsistent by visual inspection.
11	Reduce tolerance	Success	

Table 6.5: Iterations up to the first successful trajectory calculation.

It is necessary to quantify the "low constraint violations" and "low accuracy". During iteration, it may become apparent that the constrained optimization problem, to which `pykep` reduces the transfer problem, just barely fails to converge. In other words, the "constraint violations" are logged in an order of magnitude less than the expected accuracy of the problem setup. This accuracy is partially dependent on the number of discretization nodes, and thus it is allowed to increase the margin for violating constraints such that the solution is still accepted. This is preferred to raising the number of nodes to a value higher than a reasonable value for the accuracy of the study, e.g. it makes little sense to use thousands of nodes and increase computation time drastically, while the mass of the spacecraft is uncertain in the order of 100 g.

The reason that constraint violations occur at all, is that the low thrust transfer problem is "infeasible almost everywhere" in the mathematical sense, meaning that the solutions are single points rather than ranges [21]. As such, a numerical scheme will never reach such a point exactly, so a constraint violation margin is necessary to reach any solution at all.

However, when the margin is set too high, the resultant trajectory will be inconsistent, which is clearly visible as 'jumps' between affected nodes. As such, it is necessary to find a margin which leads both to a feasible solution which is also consistent. This is done in the first few iterations: once this margin is found, it remains unchanged until the accuracy of the study increases and it becomes fruitful to increase the number of discretization nodes as well.

6.4. Initial Feasible Combined Design

With a feasible trajectory option, the second goal is to achieve a feasible system design, using parametric values for the components (i.e. not existing components). The remaining iterations optimize the trajectory and design parameters, until the difference is mass budgets is less than the considered accuracy, in this case in the order of grams.

In this stage, it has not yet been necessary to make any design choices: it has been sufficient to increase or decrease propellant masses and solar array output where necessary, iterating for the changes in mass overall.

This yields the first feasible combination of trajectory and system design, shown as iteration 17 in Table 6.6.

6.5. Initial Feasible COTS Design

The first feasible combined design is continued to be iterated using existing components. This is done by replacing all components by the existing components with the most similar characteristics, leaving as variables

Iteration	Thrust [N]	I_{sp} [s]	M1 [kg]	M0 [kg]	Time [days]
11	0.2	3000	20	16.5	1310
12	0.001	3000	20	16.5	1825
13	0.002	1500	20	13.6	1693
14	0.002	1500	20	13.6	1013
15	0.0014	3050	20	16.5	1825
16	0.0014	3050	14.4	11.9	1610
17	0.0014	3050	14.1	11.7	1166

Table 6.6: Set of iterations from the first successful trajectory onward.

	Component(s)	m [kg]	V [U]	P [W]	F [N]	I_{sp} [s]
OBC	Hyperion CP400.85	0.1	-	1		
EPS	ISIS iEPS A	0.2	1	1		
COM	Hyperion CubeCat	1.4	1	15		
ADC	2 x Hyperion iADCS400	3.4	2	12		
SOL	4 x 2U ISIS + 8 x EXA DSA	2.7	-	-64		
STR	ISIS 16 U cubesat structure	2.3	-	-		
ETH	JPL MiXi	0.2	1	50	0.0014	3050
CTH	Aerojet Rocketdyne MPS130	1.3	1	-	1	240
M0		11.6	6			
EPR	Custom tank	2.5	2			
M1		14.1	8			
CPR	Custom tank	1.1	1			
M2		15.2	9			

Table 6.7: Overview of first feasible iteration using existing components.

the propellant masses and their respective structural masses, and continuing iterations until the requirements are again met.

It must be noted that finding similar components is somewhat arbitrary, as a decision must be made between, for example, a more similar dry mass or a more similar thrust value. However, the setup of this procedure allows for switching out components between iterations, and thus the choice for a 'wrong' component, i.e. one that ultimately prevents the design budgets to close, can simply be undone. Additionally, it must be noted that an initial guess, as made in subsection 6.2.6, could also be done with COTS components immediately. The procedure as described in this report allows for a fast design setup if it is known beforehand that COTS components are preferred.

Replacement with COTS components and iteration until the requirements are met, results in the design as shown in Table 6.7, with the trajectory as shown in Figure 6.2. The transfer window is from May 2029 to January 2034, with a transfer time of approximately 1700 days. In this figure, the grey inner circle represents the orbit of Earth, the outer circle represents the orbit of Mars. Red parts of the orbit indicate powered segments, while blue parts indicate ballistic coasting. Note that, in accordance with the assumption made previously to use `pykep`, all power segments are low thrust, because high thrust was used exclusively to escape Earth.

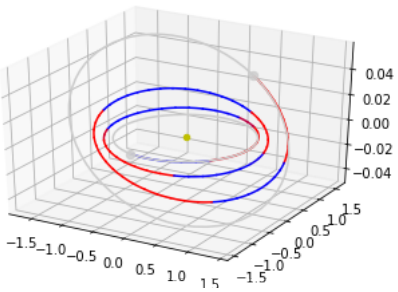

Figure 6.2: Interplanetary trajectory of the first feasible iteration using existing components, with axes in AU.

	Component	Margin [%]	Mass [kg]	With margin [kg]	Volume [U]
PAY	Placeholder			1.000	1.000
OBC	Hyperion CP400.85	5	0.007	0.007	0.100
EPS	ISIS iEPS A	5	0.184	0.193	0.265
COM	Hyperion CubeCat	5	1.330	1.397	0.960
ADC	Hyperion iADCS400	5	1.700	1.785	0.673
SOL	10 x EXA DSA + 2 x ISIS 1U	5	0.910	0.956	
STR	ISIS 12U CubeSat Structure	5	1.500	1.575	
ETH	2 x Modified Enpulsion Nano	10	1.360	1.496	1.650
CTH	2 x Modified Hyperion PM200	10	2.200	2.420	1.954
M0		5	10.829	11.370	
EPR	In Modified Enpulsion Nano	20	1.830	2.196	3.991
M1			13.200	13.566	
CPR	In Modified Hyperion PM200	20	0.989	1.186	0.914
M2				14.752	11.506

Table 6.8: Current design budgets.

For the sake of meeting the requirements, this design is sufficient. However, iterating further can lead to a more optimal solutions, with a lower mass or different cost metric. In this case, it is clear that the design is not optimal: only 9 of the available 16 cubesat units of volume are occupied. Notable as well is that approximately half of the trajectory is coasting, which means transfer time may be improved. As such, the design is iterated further to optimize volume usage.

6.6. Current Design

Using the previous iteration, a design option switch is made by replacing the 16 U cubesat structure to a 9 U cubesat structure. From this, the iteration procedure begins anew. Meanwhile, the the level of accuracy is increased step by step.

After a number of iterations, the design converges to a volume budget that consistently requires more than 9 units. As such, a switch is made again, this time to a 12 U cubesat structure. Iterating this further leads to the final design iteration for this thesis.

This iteration is presented in Figure 6.3. Note that here the component mass accuracy in grams is required, due to difference in order of mass between components. However, for miniature COTS products at high TRL, mass accuracy is considered relatively high. The budgets are considered using ESA's margin guidelines [59]. As applicable: 5% on mass and power budgets for COTS components; 10% on mass and power budget for modified COTS components; 5% on total dry mass for harness. Regarding propellant and ΔV, ESA recommends stacking margins of 5% for calculation inaccuracies, additional 5% for electric trajectory inaccuracies, and 2% for propellant residuals; this is instead replaced by an overestimate 20% margin on propellant. Note also that a power budget is now given per spacecraft operating mode, and listed in subsection 6.6.10.

The system design agrees with the trajectory design as presented in Figure 6.3. Again, the grey inner circle represents the orbit of Earth, the outer circle represents Mars. Red parts of the orbit indicate powered segments, blue parts indicate ballistic coasting. In this iteration, the transfer time is the limiting requirement: the transfer time is ≈1810 days, almost exactly 5 years, and almost all of the trajectory is with low thrust at maximum throttle. The launch window starts in December 2026, and ends in November 2031.

The design is further presented on separate pages. Figure 6.4 shows the external view of the satellite: two sides are covered with deployable solar panels, all other sides are either shielded with the aluminium panels included with the cubesat structure, or shield panels with cutouts for the ADC, COM, and payload. Deployment of the solar panels is more explicitly shown in Figure 6.10. Figure 6.5 shows the design without solar panels, shield panels, and internal structure. Components are presented as a simplified dimensions, because a CAD model is not freely available in most cases.

6.6.1. On Board Computer

The Hyperion CP400.85 is considered as the OBC, as mass, power and volume requirements are sufficient; with the lowest mass of the components under consideration. Volume is small enough to consider the CP400.85 with the EPS to occupy one cubesat unit, although in practice it is less. Storage and data handling perfor-

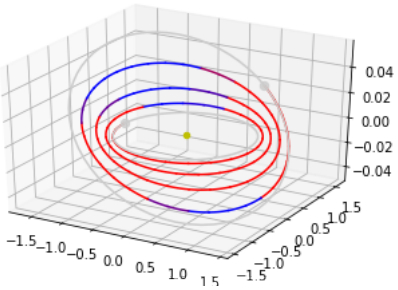

Figure 6.3: Interplanetary trajectory of the current design, axes in AU.

Hyperion CP400.85	
Mass [g]	7
Stack height [mm]	10
Idle power [W]	0.55
Peak power [W]	1
Radiation tolerance [krad]	25

Table 6.9: Some characteristics of the Hyperion CP400.85 OBC.

mance is more than sufficient, as this is limited by the communication system. Some characteristics and illustrations are presented in Table 6.9 and Figure 6.6 [65].

6.6.2. Electric Power System

For the EPS, the ISIS iEPS A component is considered as the lightest available system. Power storage is not a constraining factor: in Earth eclipse, power draw from the chemical thruster is sufficiently low. With the EPS's 22.5 Wh storage, the burn time is limited by heat dissipation rather than power storage. In interplanetary space and around Mars, eclipses do not have a significant effect on power availability. Some characteristics and illustrations are presented in Table 6.10 and Figure 6.7 [56].

6.6.3. Communication

While the design of the communication system is out of the scope of this report, two remarks can be made regarding the inclusions of the Hyperion CubeCat. Firstly, its performance and operating characteristics (≈1500 nm wavelength and ≈15 W power draw) are similar to an ongoing NASA study for deep space optical transmissions, which can be received on Earth with a 5 m diameter telescope [60]. Alternatively, the CubeCat is designed for inter satellite communication, so a relay system around Mars can also be used. In both cases, the pointing accuracy is the limiting factor. Some characteristics and illustrations are presented in Table 6.11 and Figure 6.8 [63].

6.6.4. Attitude Determination and Control

The pointing accuracy of the Hyperion iADCS400 is less than 30 arcseconds for a 12 U cubesat, which meets the 1800 arcsecond pointing requirement of the CubeCat. This means that at least, satellite to satellite communication is possible in planetary orbit; direct communication to Earth is subject to more research. Additionally, while it is beyond the scope of this research, it is considered safe to assume that the accuracy of the

ISIS iEPS A	
Mass [g]	189
Stack height [mm]	27
Storage [Wh]	22.5
Throughput power [W]	4x20
Number of MPPTs	6

Table 6.10: Some characteristics of the ISIS iEPS A.

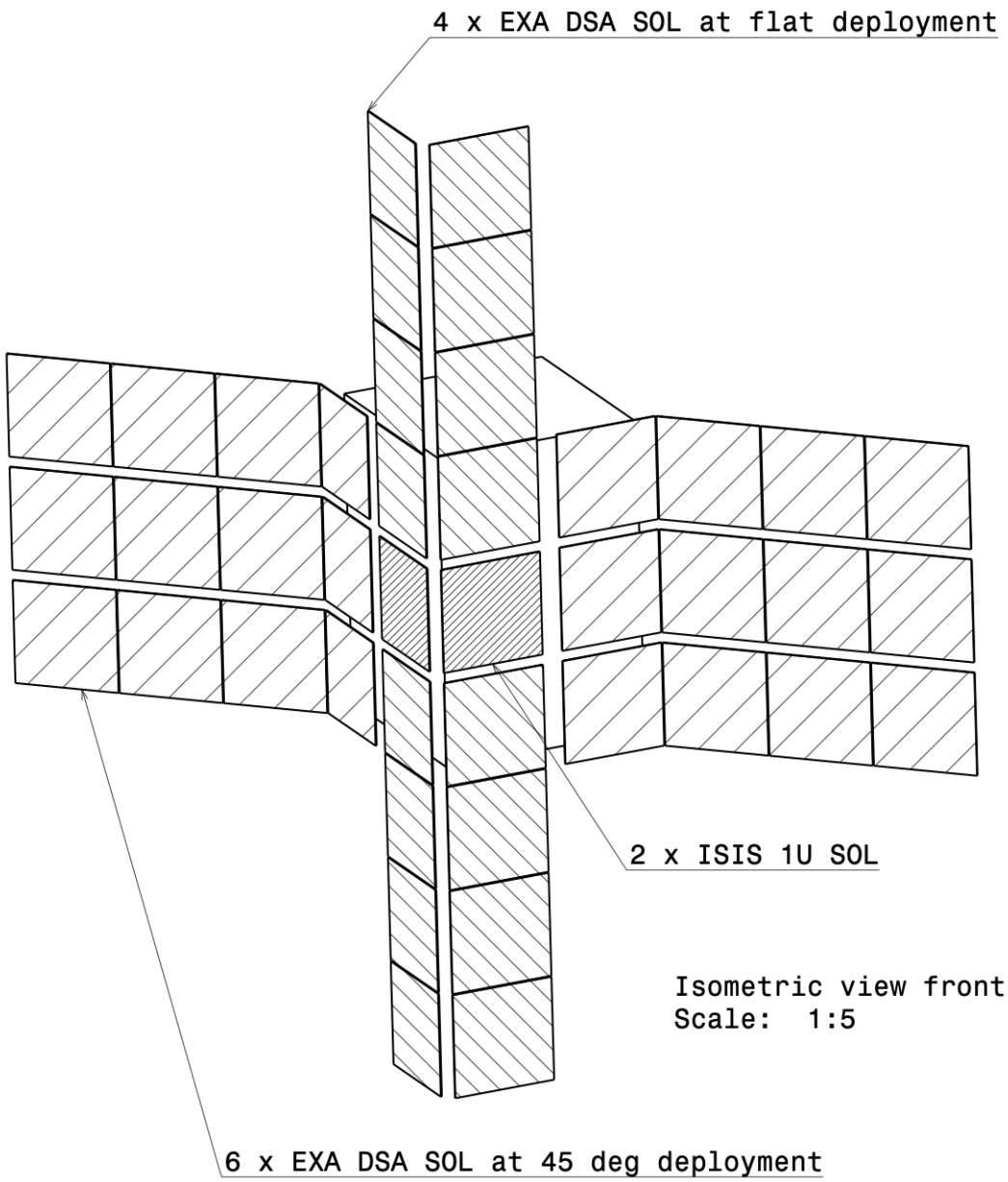

Figure 6.4: Exploded view of the solar array configuration.

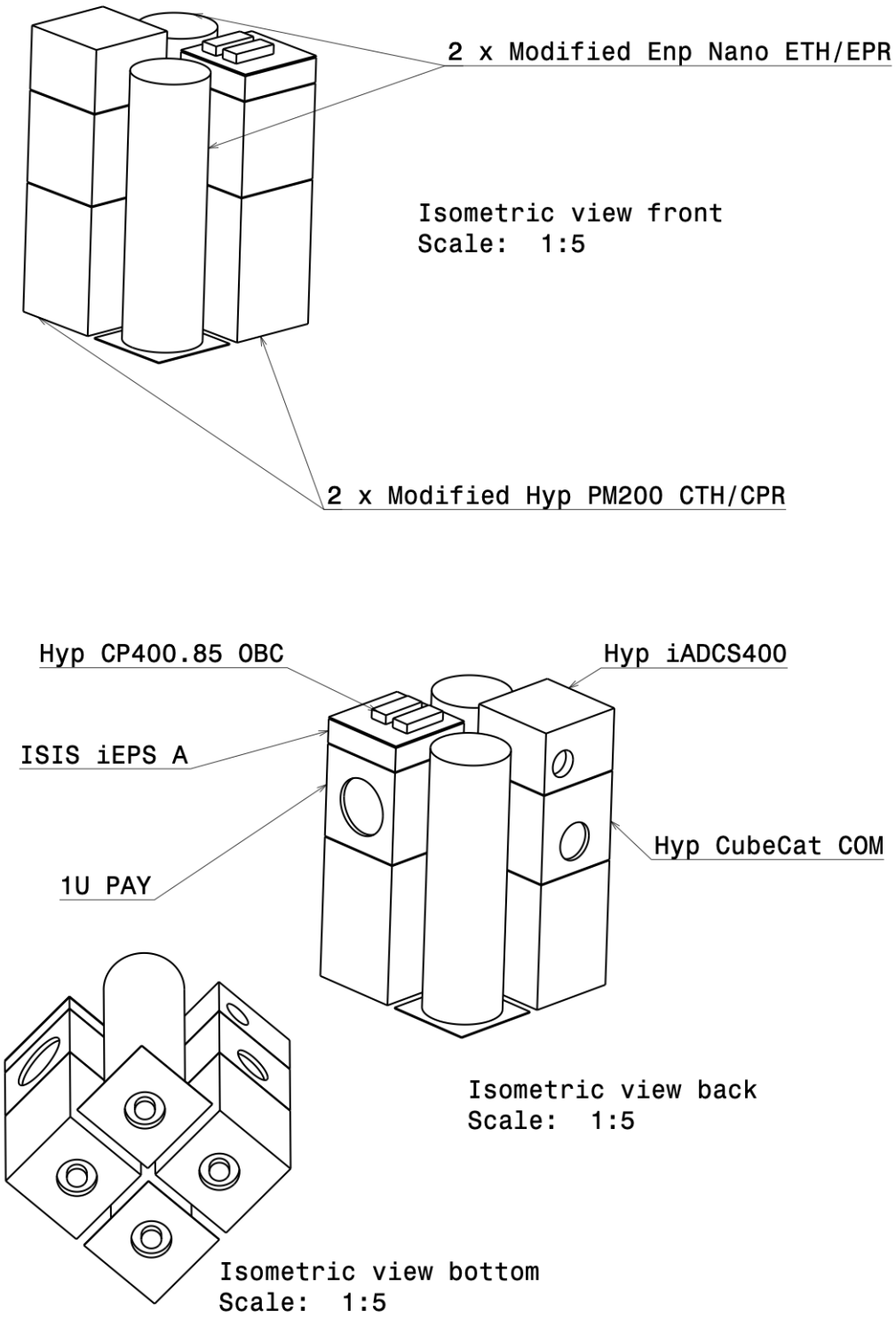

Figure 6.5: Internal view of the component configuration.

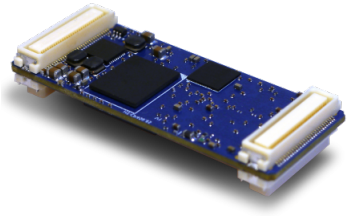

Figure 6.6: Illustration of the Hyperion CP400.85 OBC, courtesy of Hyperion Technologies.

Figure 6.7: Illustration of the ISIS iEPS A, courtesy of ISISPACE.

Hyperion CubeCat	
Mass [g]	1330
Stack height [mm]	96
Idle power [W]	1
Peak power [W]	15
Pointing accuracy [mrad]	8.7

Table 6.11: Some characteristics of the Hyperion CubeCat.

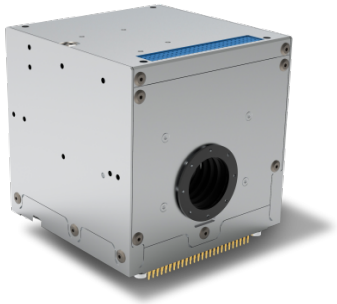

Figure 6.8: Illustration of the Hyperion CubeCat, courtesy of Hyperion Technologies.

Hyperion iADCS400	
Mass [g]	1700
Stack height [mm]	68
Idle power [W]	1
Peak power (saturated) [W]	50
Determination accuracy [mrad]	0.15
Control accuracy [mrad]	<< 18

Table 6.12: Some characteristics of the Hyperion iADCS400.

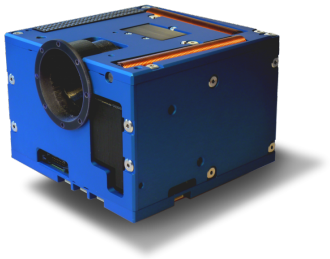

Figure 6.9: Illustration of the Hyperion iADCS400, courtesy of Hyperion Technologies.

iADCS400 star tracker and solar trackers, in combination with an optical payload, is sufficient for celestial navigation [10], with an expected 3σ of 1000 km [38].

Momentum dumping is provided internally using magnetorquers. When magnetorquers are ineffective, such as in interplanetary space, momentum dumping is instead provided by differential thrust of the low thrust systems. When this is insufficient, the high thrust systems can assist in cold gas mode, using any leftover propellant, as well as their vectoring capabilities.

Some characteristics and illustrations are presented in Table 6.12 and Figure 6.9 [62].

6.6.5. Solar Array

The EXA deployable solar array quadruples the potential solar surface, which is sufficient to run the electric thruster at almost full capacity at Mars' distance from the sun; taking into account its degradation lifetime of 5 years. Ten EXA units are used, of which six at a 45 degree deployment angle and 4 at flat deployment. Two single unit ISIS solar panels are used to cover the remaining panels.

The ISIS panels each have a mass of 50 g and generate 2.3 W when exposed to direct sunlight incident normally. The EXA solar arrays each have a mass of 135 g and generate 16.8 W when exposed to direct incident sunlight normally. They are self deploying upon a pin signal. An illustration is presented in Figure 6.10. Note that the largest version is used with three external panels, which fold in on the body in a zigzag shape [17].

6.6.6. Electric Thruster

Two Enpulsion Nano thrusters are selected, in order to both reduce travel time and align the thrust vector with the center of mass. The Nano is an indium field emission thruster with a variable performance dependent on voltage and power setting. Extrema per thruster are at either 0.3 mN and 4500 s, or at 0.35 mN and 3500 s. The former is selected in to reduce propellant mass. As the Nano contains its own propellant, it is proposed

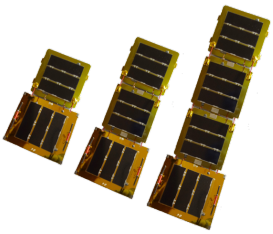

Figure 6.10: Illustration of various sizes of the EXA deployable solar array, courtesy of the Ecuadorian Space Agency.

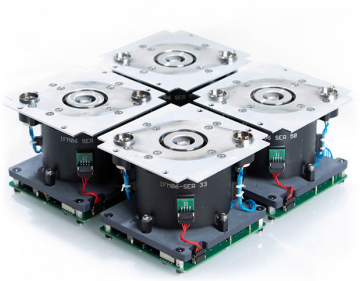

Figure 6.11: Illustration of a group of four units of the Enpulsion IMF Nano, courtesy of Enpulsion Spacecraft Technology.

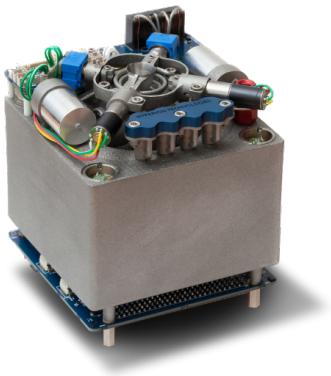

Figure 6.12: Illustration of the Hyperion PM200, courtesy of Hyperion Technologies.

to modify it to contain more propellant, rather than to design an separate tank system. The Nano has been demonstrated in orbit, and has no limit on its burn time. An illustration is presented in Figure 6.11, while four units are pictured, two are used in total in the proposed design [66].

6.6.7. Chemical Thruster

Similarly, two Hyperion PM200 nitrous oxide and propene liquid bipropellant thrusters are selected, modified to contain additional propellant. This component has a thrust of 0.5 N and an I_{sp} of 285 s. It needs to cooldown for a second every 11 seconds, which results in an average total thrust of 0.9 N. This can be increased by a heatsink, for example using the larger tank to take up heat, although the difference may not be significant enough to warrant the costs for this. By using high thrust calculations, it requires 5 passes though pericenter to reach escape velocity, which equals 9 passes through the van Allen belts. This uses a burn time (including cooldown) of 600 s and neglects gravity losses in favour of the 20% propellant margin [36]. An illustration is presented in Figure 6.12 [64].

6.6.8. Radiation

Lastly, all components are rated for the radiation environment in LEO. More specifically, the ADC is rated for a total dose of 45 krad and the OBC is rated for a total dose of 25 krad. The solar panels are rated for at least 4 years in LEO. For all components, information is lacking on their performance and reliability over time with respect to the total ionizing dose. This makes it difficult to assess the effect of the radiation outside Earth's magnetosphere. However, it is considered that most of the total radiation dose for outbound spacecraft is acquired during passage of the van Allen belts. The result for the high thrust Earth escape trajectory of 9 van Allen belt passes is therefore considered to be sufficiently low, compared to e.g. Mani et al. [38], such that the components are not expected to degrade much faster than rated for.

6.6.9. Interface Layout

The interface layout is illustrated in Figure 6.13.

By using the CubeSat standard, the components are connected simply through the cubesat bus. The EPS is powered by the solar panels directly. The iEPS module support six independent power inputs through

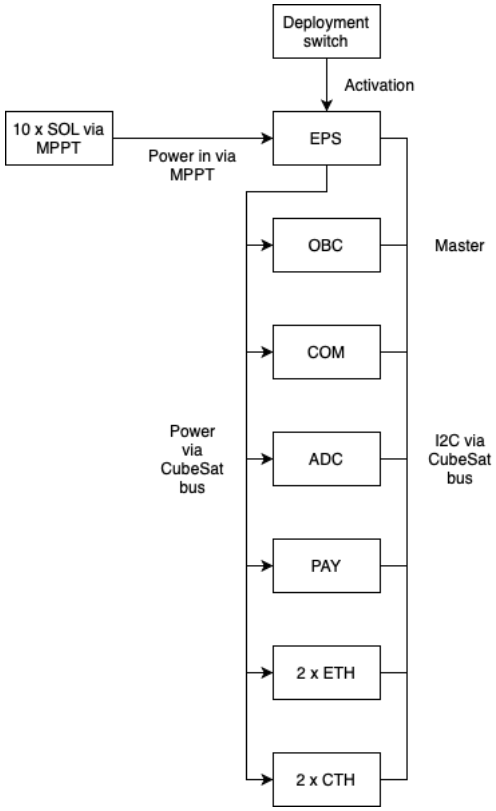

Figure 6.13: Interface layout of the components.

maximum power point trackers (MPPTs); this is sufficient, as 6 different input voltages are expected from the solar array configuration. The EPS distributes the power to at least the OBC, and to other components when commanded by the OBC.

Data handling happens simply though the data bus: all components support the I2C communication standard with the OBC as master. Since all components are self contained, and in the case of the thruster modules self pressurized, there is no need for a separate layout for pressure tanks, valves, etc.

6.6.10. Spacecraft Modes

For the power budget, it is necessary to define a number of spacecraft modes, which manages the level of activity and power draw of each component. In case the electric thruster is active, it receives all leftover power it can draw: this is often its optimal power draw of 40 W, but at a Mars distance from the sun, this can be 37 W when many other components are active. Note that vertically, all components are connected through the cubesat bus: this is standardized, which is one of the advantages of the CubeSat standard.

A short description of each mode is presented in Table 6.13. A schematic for mode changes follows in Figure 6.14. All modes have a link to safe mode if: the current mode completes its objective, a critical error occurs, the OBC reboots, the battery charge is low, or momentum storage is full.

Power usage is estimated per mode. Based on the respective datasheets of the current design, it is considered which components are enable and thus draw power, and which components are active and thus draw peak power instead of idle power. For modes where peak power is used, it is considered that this peak power is used throughout the entire duration of this mode. While this is an overestimate of the power draw, the solar array area is large enough to meet even this overestimate with a significant margin: the power needed to drive the electric thrusters is much larger than the other components. In fact, the battery receives a net charge in almost all cases. Even right after deployment when the EXA solar panels are still stowed, the spacecraft slowly charges during countdown mode. In the worst case scenario, namely firing the chemical thrusters in eclipse, the spacecraft should be able to run at peak power for 30 minutes before the batteries are empty. This will never happen, because the chemical thrusters do not need to fire in eclipse, they, the both the duration of an eclipse and of a burn are less than 30 minutes.

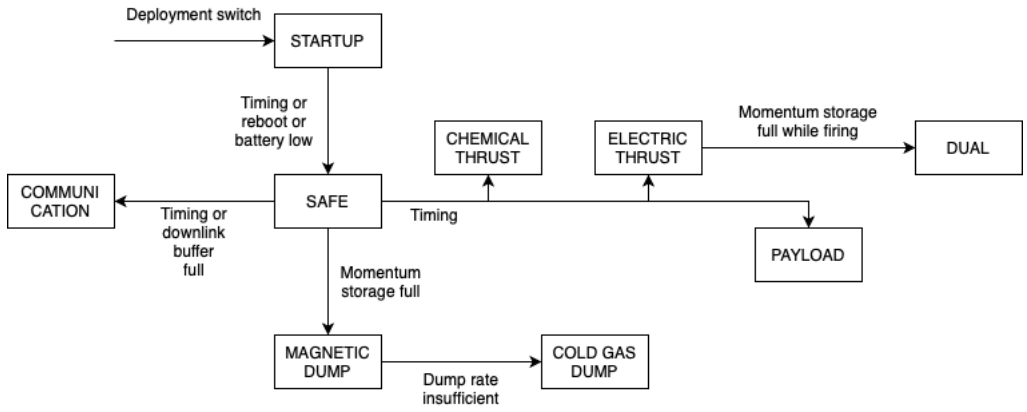

Figure 6.14: State machine for the proposed spacecraft modes.

Thus, it can be concluded that the budget closes for mass, volume and power use.

6.6.11. Alternatives

Whether alternatives to the listed components can be proposed depends on the component itself and on the results of further iterations with alternative components. For the OBC, it is very likely that many other components would work: the requirement for data throughput and storage depends on the payload and is limited by downlink performance instead, but alternatives have been found to be heavier. For the EPS, power storage is not critical, but other components with a similar throughput, such as the Endurosat EPS have been found to be heavier, which affects the mass budget through iterations. No other component has been found that could replace the CubeCat as COM module, although the design of the communication system beyond the link budget is recommended for further research. Simple patch antennas could work, but exclusively with a Mars relay in place, which significantly increases mission complexity. Similarly, the ADC system was found to be the most compact that meets pointing requirements for a cubesat of this size, alternatives such as the CubeSpace ADCS could work through further iterations. Lastly, many other solar array configurations could work, although the current solution maximizes the area for the number of MPPTs available.

6.7. Design Recommendations

There are a number of design recommendations to ensure this design can be completed. Each of these has been discussed before.

Firstly, it is necessary to confirm that the proposed modifications to the thruster tanks can be realised without needing to adapt the rest of the design further. As of now, it is assumed that enlargement of the tanks in the axial direction is not constrained by the pressure in the tanks, as such the wall thickness is not expected to increase significantly.

Secondly, it must be confirmed that the individual components can survive at least 5 years of radiative doses in interplanetary space and around Mars. While each component is listed to last more than 5 years of LEO radiation, it is difficult to estimate lifetimes for an interplanetary transit, considering the lack of documentation on ionizing doses and the availability of radiative models for the specified trajectory. In all, the aspects of spacecraft design beyond the scope of this report, such as thermal control, data handling, and communication, must be investigated in depth.

Lastly, it is recommended to research the options for a Venus gravity assist to Mars. For the current case, this did not seem promising, but this may be different for different initial guesses for design and trajectory.

6.8. Verification and Validation

In systems design, verification and validation are important parts of the design process, for both the driving factors for the design as well as the design itself.

Startup Mode: activated by the deployment switch, the EPS is active by hardware design as soon as it receives solar power. It activates the OBC merely to countdown until deployment, as required by cubesat deployment guidelines. Active: EPS, OBC (countdown). Estimated power use: 0.5 W.
Safe Mode: activated from startup mode if the countdown completes, or if the OBC reboots, or if the battery is low. Safe mode activates from other modes if the current mode completes its objective, a critical error occurs, OBC reboots, battery is low, or momentum storage is full while the current mode cannot handle this. Solar panels deploy, if not already deployed. ADCS activates, solar panels are pointed at the Sun. Battery Active: EPS, OBC, ADC (point at Sun), SOL (deploy), COM (passive). Estimated power use: 10 W peak, 3 W nominal.
Payload Mode: activates according to timing. Active: EPS, OBC, ADC (point at target), PAY (capture). Estimated power: 8 W peak.
Communication Mode (Downlink): activates according to timing, downlink buffer state. Active: EPS, OBC, ADC (point at Earth or relay), COM (active). Estimated power: 23 W peak.
Magnetic Dump Mode: Activates if momentum storage is full. Active: EPS, OBC, ADC (magnetorquers on). Estimated power: 7 W peak.
Cold Gas Momentum Dump Mode: Activates if momentum storage is full and magnetorquer dump rate is insufficient. Active: EPS, OBC, ADC (magnetorquers on), CTH (cold gas, vectoring). Estimated power: 40 W peak.
Chemical Thrust Mode: Activates according to timing. Active: EPS, OBC, ADC (point prograde), CTH (liquid bipropellant, vectoring). Estimated power: 40 W peak, 32 W nominal.
Electric Thrust Mode: Activates according to timing. Active: EPS, OBC, ADC (point prograde), ETH (max power). ETH power available at Mars distance from Sun: 72 W at peak draw, 80 W at nominal draw (ideal is 80 W).
Dual Thrust Mode: Activated from Electric Thrust Mode if momentum storage is full and magnetorquer and differential thrust dump rate is insufficient. Active: EPS, OBC, ADC (point prograde), ETH (max power), CTH (cold gas, vectoring). ETH power available at Mars distance from Sun: 46 W at peak draw, 74 W at nominal draw (ideal is 80 W).

Table 6.13: Overview of modes.

6.8.1. Requirement Validation

The few top level requirements have been validated as both being in the correct form, as well as following the objective of this thesis. As such the requirements are well defined and supported by rationales linked to similar feasibility studies.

As for the internal requirements for each component specifically, this could not be validated per se. However, each internal requirement was linked to a specific numeric value. For a final 'converged' iteration, all these requirements are met, so the design is internally consistent.

6.8.2. System Verification

The design meets the requirements set previously. A list is given to denote requirement compliance.

- REQ-1: The spacecraft shall be launched as a piggyback option or secondary payload.

  Met: subrequirements met.

  - REQ-1.1: The spacecraft shall adhere to the CubeSat standard.
    Met: design follows the 12U CubeSat standard.
  - REQ-1.2: The spacecraft shall have a volume of at most 16 U.
    Met: design follows the 12U CubeSat standard.
  - REQ-1.3: The spacecraft shall have a mass of at most 21.3 kg
    Met: design has a wet mass (M2) of 14.8 kg.
  - REQ-1.4: The spacecraft shall be launched into an initial orbit for which piggyback options exist.
    Met: piggyback launches to GTO orbits are commercially available.

- REQ-2: The spacecraft shall allow a payload of 1 U volume

  Met: design starting point.

- REQ-3: The spacecraft shall allow a payload of 1.3 kg

  Met: design starting point.

- REQ-4: The spacecraft shall allow a payload to draw 1 W of operating power when not performing a maneuver

  Met: design starting point. Payload may draw 1 W at all times.

- REQ-5: The spacecraft shall reach a final orbit around Mars.

  Met based on estimates, needs additional research: ΔV budget closes for a transfer orbit from GTO to a Mars ballistic capture. ΔV budget for momentum dumping closes based on estimates; needs further investigation. Radiative tolerance is considered sufficient based on estimates; needs more data and testing.

- REQ-6: The spacecraft shall reach its final orbit in at most 5 years.

  Met: total transfer time is 1810 days, slightly less than 5 years.

- REQ-7: The spacecraft shall be launched in the year 2030 at the latest.

  Met: the launch window starts in December 2026.

- REQ-8: The spacecraft shall not use hydrazine thrusters.

  Met: chemical thruster uses nitrous oxide and propane, electrical thruster uses indium.

Some additional notes must be made to the expected compliance with REQ-5: the ability to reach a Mars orbit. While the ΔV budget for the transfer has been an integral part of the optimization procedure, the ΔV budget for momentum dumping has been considered only from estimates, although from historical trends it should fall within the ΔV margins; the same holds for radiation tolerance of the components. As such these values were held fixed during optimization.

The calculated trajectory passes pykep's internal verification assertions, and aligns with transfer times and ΔV requirements as presented in, for example, Mani et al. [36]. The design itself is difficult to verify further, as no similar cubesat mission could be found, which is the case especially regarding propulsion systems. The overall layout of the design is very similar to other cubesat layouts, but this is due to the standardized nature of the cubesat format.

As mentioned before, the addition of a radiation model to the trajectory optimization software would require additional data on the total ionizing dose (TID) in such trajectories for the selected components. Secondly, a better optimization of the ΔV budget for momentum dumping requires a per iteration calculation of the moments of inertia, for example in an automated CAD environment; and a model disturbances dependent on spacecraft shape and trajectory. This might not have significant advantages over an approach using estimates and margins.

Similarly, the development of navigation software dependent on a star tracker, solar trackers and potentially optical payloads is left as future work.

For further verification, it is recommended that each component is subjected to further testing, specifically regarding radiative and thermal tolerance beyond Earth orbits. This holds for the integrated model as well.

6.8.3. System Validation

For the spacecraft itself, the development of an actual cubesat mission to Mars would provide final validation of the design presented in the latter part of this thesis. This would show that a payload can be delivered to Mars using a piggyback launched cubesat. For the sake of this thesis, analyses on e.g. stakeholders and risk were left aside; these should be considered for a validation mission.

For the development of a framework to design dual thrust spacecraft and trajectories, the presented design gives an initial indication that the framework is useful and can lead to optimized results. Validation of this framework first requires further completion: the inclusion of more variables in each iteration, further automation, and finally further use in the design of different missions.

6.8.4. Conclusions

The design presented in this chapter solves the design problem as stated in chapter 3, and thus answers the first research question on the feasibility of a standalone cubesat mission to Mars. While an automated optimization method was not completed for this thesis, the manual strategy has proven sufficient. During the design process, it has become apparent that the feasibility of interplanetary cubesat missions does is indeed very dependent on the optimality of the trajectory and the systems design.

While much additional work is recommended before the design can be finalized, the presented design is seen as a good starting point for further work. Additionally, the design is complete in terms of components and integration, such that integration tests could conceivably be performed next.

Lastly, the design shows the convenience that results from the use of COTS systems, which significantly reduces the time needed per iteration.

7

Conclusion and Recommendations

It can be concluded that, while an automated approach based on optimal control has not been successful, a manual approach based on the same concepts has resulted in a feasible design.

It is useful to return to the research questions as listed in the introduction.

What is a useful framework for optimizing low thrust trajectories?

From the literature study, it is concluded that there are, in general, four optimization strategies for low thrust trajectories. Firstly, Edelbaum's approximation is a simple solution for a simplified use case, but this use case is common for Earth orbiting missions. It is useful for fast calculations in use cases where its assumptions hold. Secondly, the indirect and direct methods do not rely on the same assumptions, and are therefore used for the optimization of general trajectories. Both transform the optimal control problem into a boundary value problem, but in different ways: the direct method discretizes a low thrust trajectory as many high thrust ones, the direct method finds an analytic expression for the optimal control at any point in time. Regardless, both must be numerically solved as a constrained optimization problem, and therefore discretized. Lastly, the novel evolutionary neurocontrol method is considered a promising alternative. It works by linking control variables with the spacecraft state through a neural network, which is optimized though an evolutionary scheme. In general, the direct and indirect methods are popular subjects of research, and both are used heavily in industry to design mission trajectories.

What is a useful framework for optimizing in tandem the trajectory and systems design of a dual thrust spacecraft?

The indirect method is found to be solvable with static control variables as the dynamic control variables; these can be used to encode system parameters, making it the first choice for this thesis. For a numerical implementation using a BVP solver, the general equations for a dual thrust spacecraft are derived. Additionally, this provides an understanding of the internal workings of general OCP solvers and higher level trajectory optimization programs. Further derivations are automated in `sympy`, and the resultant system is implemented using a BVP solver in `scipy`. Alternative implementations are considered as well. Firstly, general OCP solver `gekko` is used using the same equations, internally performing derivations and optimization, and theoretically equivalent to the `scipy` implementation. Secondly, it is proposed to use `pykep`'s low thrust capabilities in combination with manual high thrust and system design calculations. Lastly, it is proposed to implement a custom problem object in `pykep`, with the same high thrust and system design capabilities. The latter two options are considered most promising.

Which implementation is useful for combined optimization of a dual thrust system?

Several issues are encountered when implementing the indirect method in `scipy`. Most importantly, it has not been possible to get the system to converge. The initial guess of the states can be implemented using any coordinate system, but different issues appear depending on the system used. For the costates, however, strategies to determine an consistent initial guess rely on an iterative approach. For each change in spacecraft properties, this guess is invalid and must be recalculated. This means that an integrated optimization method as proposed has no benefits over a nested optimization scheme using existing means.

The lack of a consistent method to find a useful initial guess is also apparent in the implementation using `gekko`. In a similar fashion, it has not been possible to get the system to converge. It is assumed that this is due to the same issues; the black box nature of OCP solvers makes it easier to implement, but more difficult to investigate issues.

Lastly, manual use of `pykep`'s existing functionality is without issues, but requires some assumptions in the use case. Most importantly, it is necessary to assume that all low thrust propellant is used first, and the high thrust system used only later. This is a reasonable assumption, but removes the possibility of investigating intermittent or even dual usage. Implementing a custom problem object within `pykep` is expected to remove the need for these assumptions, but will likely require a rewrite of `pykep`'s internal working, which is beyond the scope of this thesis. As such, a manual implementation using existing pykep functionality is considered the most useful for this report. For future studies, however, it remains useful to consider the additional requirements as given below.

What is a feasible design for a standalone cubesat mission to Mars?

A possible design is generated using the selected implementation. Based on an arbitrary single unit payload, the design is started from an initial guess, which is unusual in system design, and repeatedly iterated together with its trajectory. From the first feasible COTS design, due to the overestimated volume budget, it is decided to switch the cubesat structure, and iterate to convergence once more. The resultant design itself is a 12 U cubesat with a wet mass under 15 kg, and a dry mass of 11.4 kg. The design consists of high TRL COTS components, most of which are unmodified. For the COTS thruster components, only the propellant tank size needs to be adjusted. In all, one side of the spacecraft contains deployable solar panels, while payload and ADC ports are on the opposite side. The resultant trajectory includes a high thrust Earth escape from GTO, with 9 passes through the Van Allen belts. This is followed by a low thrust orbit to Mars of slightly less than 5 years, of which almost all is powered flight.

This concludes the research questions for this thesis. Additionally, recommendations have been given throughout this report, and are summarised here. Firstly, it is recommended that further research is done towards finding an integrated optimization procedure, as was attempted in this report. This may be a variation on the approach presented here, or it can be based on the evolutionary model as discussed further. Secondly, it is recommended that the procedure used in the second part of this report is tested further in other mission designs. Recommendations were given towards a higher level of automation, and the inclusion of models such as a radiative model, such that more variables can be evaluated at each iteration. Thirdly, it is recommended that the cubesat design presented in this report is developed further. This is said with regard to further iterations of the design itself, and confirmation of estimations on radiation tolerance and momentum dumping. It is further recommended to increase the scope of the design, specifically towards thermal control, communication, and trajectory options. Lastly, it is recommended that an independent interplanetary cubesat mission be undertaken by an academic institute, such as the TU Delft. While the costs of such a mission would be above that of a LEO missions such as the Delfi family, it would not be several orders of magnitude greater, considering the material costs and launch costs. Such a demonstration of cubesat technology would likely yield great prestige to the institute that does so, and advance research and investment towards small scale satellite missions.

Postface

While a preface is more common in these types of documents, it seems more logical to place a discussion at the end of the report. This discussion relates to the personal aspects of the MSc thesis, and to personal views on the quality of the work produced.

First and foremost, the entirety of the thesis work has been a learning experience unlike any previous course work and project work, in the BSc track and the first year of the MSc track. Starting with the literature study, the choice for a subject without any prior knowledge has led to a considerable difficulty throughout the first six months of the project. However, it has been an enlightening process to gradually understand more and more of optimal control theory and optimization techniques in general. This has resulted in a different personal view on astrodynamics and on the nature of design problems in general. As control theory and system design have been personal favourite parts of aerospace engineering, this has been a great opportunity to combine them. It has also been a learning experience to work on this project individually, as individual work is remarkably uncommon throughout the aerospace tracks, both in the BSc and the MSc. Specifically, it has been difficult but insightful to learn to balance the reliability on oneself with the availability of others.

The inability to present a full fledged optimization system as intended has been experienced as a significant setback, and continues to leave doubts on the usefulness of this thesis and its contribution to further studies. However, the presented spacecraft is the result of what has become a personal design philosophy, and is presented with confidence that an institution such as the TU Delft could have an interplanetary mission for students to work on, if it is willing to make the investments.

Ir has become an additional personal goal to continue the work done for this thesis. It should be possible to at least finish a low thrust trajectory optimizer, without any further features like dual thrust and systems design, for personal use. Completing and maintaining a working system would be a valuable learning experience in addition to the attempts so far, and such a program may be useful for any future projects.

The use of fast generation and evaluation of solutions, which ended up being the core of this thesis, seems like a useful design view for future projects as well. Especially the idea of using a neural network for optimizing control was an interesting introduction to artificial intelligence. Looking back, this path could have been explored instead of the attempts presented in this report, but it cannot be said weather this would have been more fruitful. Regardless, these ideas can be interesting to implement in any future coursework at the Control & Simulation department.

Lastly, the design of low thrust trajectories remains an active field in research and industry. In fact, `pykep` is currently in active development, and internship opportunities exist for work on these systems specifically. Perhaps the experience gained in this thesis can be of use.

I would like to end this report by expressing my gratitude to the many people whose companionship, support, and love I have had the pleasure to receive. I wish to thank my parents and my brother for their support not only throughout my time at university, but as long as I can remember. I wish to thank my good friend with whom I started the BSc in 2015, adventured through the minor in 2017, and will soon finish the MSc track. I wish to thank my friends whom I spent time with for giving me the opportunity to relax, and whom I lived with for giving me a feeling of home, especially throughout the last few months of social isolation. I wish to thank my supervisor, for his support and guidance for this thesis, not unlike during the Design Synthesis Exercise two years ago; our correspondence was always fast, professional, and straight to the point, which I very much appreciated. Lastly, I wish to thank the reader, whoever they may be. I hope there will be more from me to read in the future.

Domas Syaifoel
Wassenaar, December 2020

Bibliography

[1] Christina Aas, Barry TC Zandbergen, Rob J Hamann, and Eberhard KA Gill. Scales: A system level tool for conceptual design of nano-and microsatellites. In *Small Satellites for Earth Observation: 7th International Symposium of the International Academy of Astronautics (IAA), 4-8 May 2009, Berlin, Germany*, 2009.

[2] Christina LO Aas, Barry TC Zandbergen, Rob J Hamann, and Eberhard KA Gill. Development of a system level tool for conceptual design of small satellites. In *Proceedings of the 7th Annual Conference on Systems Engineering Research: CSER 2009, 20-23 April 2009, Loughborough University, UK*. Research School of Systems Engineering, Loughborough University, 2009.

[3] Logan D. R. Beal, Daniel C. Hill, R. Abraham Martin, and John D. Hedengren. Gekko optimization suite. *Processes*, 6(8), 2018. ISSN 2227-9717. doi: 10.3390/pr6080106. URL `http://www.mdpi.com/2227-9717/6/8/106`.

[4] David A Bearden. Small-satellite costs. *Crosslink*, 2(1):32–44, 2001.

[5] Régis Bertrand and Richard Epenoy. New smoothing techniques for solving bang–bang optimal control problems—numerical results and statistical interpretation. *Optimal Control Applications and Methods*, 23(4):171–197, 2002.

[6] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[7] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. Routledge, 2018.

[8] Lorenzo Casalino. Approximate optimization of low-thrust transfers between low-eccentricity close orbits. *Journal of Guidance, Control, and Dynamics*, 37(3):1003–1008, 2014.

[9] Lorenzo Casalino and Guido Colasurdo. Improved edelbaum's approach to optimize low earth/geostationary orbits low-thrust transfers. *Journal of guidance, control, and dynamics*, 30(5):1504–1511, 2007.

[10] Stefano Casini, Iosto Fodde, Steven Engelen, Bert Monna, Angelo Cervone, and Eberhard Gill. Towards the use of commercial-off-the-shelf small-satellite components for deep-space cubesats: a feasibility and performance analysis. 2020.

[11] Andrea Cassioli, Dario Izzo, David Di Lorenzo, Marco Locatelli, and Fabio Schoen. Global optimization approaches for optimal trajectory planning. In *Modeling and optimization in space engineering*, pages 111–140. Springer, 2012.

[12] Pascal Chabert, J Arancibia Monreal, Jérôme Bredin, Lara Popelier, and Ane Aanesland. Global model of a gridded-ion thruster powered by a radiofrequency inductive coil. *Physics of Plasmas*, 19(7):073512, 2012.

[13] Bernd Dachwald. Low-thrust trajectory optimization and interplanetary mission analysis using evolutionary neurocontrol. *Doctor Tesis-Universität der Bundeswehr München Fakultät für Luft-und Raumfahrttechnik Institut für Raumfahrttechnik*, 2004.

[14] Bernd Dachwald, Andreas Ohndorf, and Bong Wie. Solar sail trajectory optimization for the solar polar imager (spi) mission. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6177, 2006.

[15] Laurence Charles Ward Dixon and MC Biggs. The advantages of adjoint-control transformations when determining optimal trajectories by pontryagin's maximum principle. *The Aeronautical Journal*, 76(735): 169–174, 1972.

[16] Brian R Donius and Joshua L Rovey. Ionic liquid dual-mode spacecraft propulsion assessment. *Journal of Spacecraft and Rockets*, 48(1):110–123, 2011.

[17] Agencia Espacial Civil Ecuatoriana. Deployable solar array datasheet. Technical report, 2016.

[18] Theodore N Edelbaum. Propulsion requirements for controllable satellites. *Ars Journal*, 31(8):1079–1089, 1961.

[19] Pascaline Grondein, Trevor Lafleur, Pascal Chabert, and Ane Aanesland. Global model of an iodine gridded plasma thruster. *Physics of Plasmas*, 23(3):033514, 2016.

[20] Darren L Hitt, Charles M Zakrzwski, and Michael A Thomas. Mems-based satellite micropropulsion via catalyzed hydrogen peroxide decomposition. *Smart Materials and Structures*, 10(6):1163, 2001.

[21] Dario Izzo, Will Binns, Alessio Mereta, Christopher Iliffe Sprague, dhennes, Bert Van den Abbeele, Chris Andre, Krzysztof Nowak, Nat Guy, Alberto Isaac Barquín Murguía, Pablo, Frédéric Chapoton, GiacomoAcciarini, Moritz v. Looz, dietmarwo, Mike Heddes, Anatoli Babenia, Baptiste Fournier, Johannes Simon, Jonathan Willitts, Mateusz Polnik, Sanjeev Narayanaswamy, and Jack Yarndley. esa/pykep: Optimize. October 2020. doi: 10.5281/zenodo.4091753. URL `https://doi.org/10.5281/zenodo.4091753`.

[22] Fanghua Jiang, Hexi Baoyin, and Junfeng Li. Practical techniques for low-thrust trajectory optimization with homotopic approach. *Journal of guidance, control, and dynamics*, 35(1):245–258, 2012.

[23] Jean Albert Kechichian. Reformulation of edelbaum's low-thrust transfer problem using optimal control theory. *Journal of Guidance, Control, and Dynamics*, 20(5):988–994, 1997.

[24] Jean Albert Kechichian. Low-thrust eccentricity-constrained orbit raising. *Journal of Spacecraft and Rockets*, 35(3):327–335, 1998.

[25] Jean Albert Kéchichian. *Applied Nonsingular Astrodynamics: Optimal Low-Thrust Orbit Transfer*, volume 45. Cambridge University Press, 2018.

[26] Mischa Kim. Continuous low-thrust trajectory optimization: techniques and applications. 2005.

[27] Craig A Kluever. Optimal earth-moon trajectories using combined chemical-electric propulsion. *Journal of guidance, control, and dynamics*, 20(2):253–258, 1997.

[28] Craig A Kluever. Using edelbaum's method to compute low-thrust transfers with earth-shadow eclipses. *Journal of Guidance, Control, and Dynamics*, 34(1):300–303, 2011.

[29] Craig A Kluever and Bion L Pierson. Optimal earth-moon trajectories using nuclear electric propulsion. *Journal of Guidance, Control, and Dynamics*, 20(2):239–245, 1997.

[30] David Krejci, Marco Gomez Jenkins, and Paulo Lozano. Staging of electric propulsion systems: Enabling an interplanetary cubesat. *Acta Astronautica*, 160:175–182, 2019.

[31] J Kriz. A uniform solution of the lambert problem. *Celestial mechanics*, 14(4):509–513, 1976.

[32] S Lee, A Mehrparvar, D Pignatelli, J Carnahan, R Munakata, W Lan, A Toorian, and A Hutputanasin. Cubesat design specification rev. 13. *The CubeSat Program, Cal Poly SLO*, 2014.

[33] John Leonard. Systems engineering fundamentals. Technical report, Defence Systems Management, COLL Fort Belvoir VA, 1999.

[34] Paola Libraro. *A globally nonsingular quaternion-based formulation for all-electric satellite trajectory optimization*. PhD thesis, Princeton University, 2016.

[35] K Mani, Stefano Boccelli, F Topputo, A Cervone, et al. Electric propulsion characterization for a standalone mars cubesat. pages 1–15, 2019.

[36] Karthik V Mani, Angelo Cervone, and Francesco Topputo. Combined chemical–electric propulsion for a stand-alone mars cubesat. *Journal of Spacecraft and Rockets*, pages 1–15, 2019.

[37] Karthik Venkatesh Mani, Francesco Topputo, and Angelo Cervone. Dual chemical-electric propulsion systems design for interplanetary cubesats. In *Space Propulsion Conference 2018*, pages 1–12, 2018.

[38] Karthik Venkatesh Mani, Alvaro Sanz Casado, Vittorio Franzese, Angelo Cervone, and Francesco Topputo. Systems design of mario: Stand-alone 16u cubesat from earth to mars. In *70th International Astronautical Congress (IAC 2019)*, pages 1–17, 2019.

[39] Giovanni Mengali and Alessandro A Quarta. Trajectory design with hybrid low-thrust propulsion system. *Journal of Guidance, Control, and Dynamics*, 30(2):419–426, 2007.

[40] Todd Mosher. Spacecraft design using a genetic algorithm optimization approach. In *1998 IEEE Aerospace Conference Proceedings (Cat. No. 98TH8339)*, volume 3, pages 123–134. IEEE, 1998.

[41] Todd Mosher. Conceptual spacecraft design using a genetic algorithm trade selection process. *Journal of Aircraft*, 36(1):200–208, 1999.

[42] Todd Mosher, Mark Barrera, Dave Bearden, and Norman Lao. Integration of small satellite cost and design models for improved conceptual design-to-cost. In *1998 IEEE Aerospace Conference Proceedings (Cat. No. 98TH8339)*, volume 3, pages 97–103. IEEE, 1998.

[43] Beny Neta and David Vallado. On satellite umbra/penumbra entry and exit positions. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF MATHEMATICS, 1997.

[44] Andreas Ohndorf. *Multiphase low-thrust trajectory optimization using evolutionary neurocontrol*. PhD thesis, Technische Universität Delft, 2016.

[45] Steven R Oleson, Roger M Myers, Craig A Kluever, John P Riehl, and Francis M Curran. Advanced propulsion for geostationary orbit insertion and north-south station keeping. *Journal of Spacecraft and Rockets*, 34(1):22–28, 1997.

[46] Vidhya Pallichadath, S Radu, MAC Silva, DC Guerrieri, MS Uludag, D Maxence, B Zandbergen, and A Cervone. Integration and miniaturization challenges in the design of micro-propulsion systems for picosatellite platforms. In *Sevilla, Spain. 3AF, ESA and CNES, Space Propulsion Conference*, 2018.

[47] VG Petukhov, AV Ivanyukhin, and Woo Sang Wook. Joint optimization of control and main trajectory and design parameters of an interplanetary spacecraft with an electric propulsion system. *Cosmic Research*, 57(3):188–203, 2019.

[48] JOHN E PRUSSING. Optimal four-impulse fixed-time rendezvous in the vicinity of a circular orbit. *AIAA Journal*, 7(5):928–935, 1969.

[49] Giuseppe D Racca. New challenges to trajectory design by the use of electric propulsion and other new means of wandering in the solar system. *Celestial Mechanics and Dynamical Astronomy*, 85(1):1–24, 2003.

[50] Joshua Rovey, Christopher T Lyne, Alex J Mundahl, Nicolas Rasmont, Matthew S Glascock, Mitchell J Wainwright, and Steven P Berg. Review of chemical-electric multimode space propulsion. In *AIAA Propulsion and Energy 2019 Forum*, page 4169, 2019.

[51] Ya G Sapunkov and Yu N Chelnokov. Construction of optimum controls and trajectories of motion of the center of masses of a spacecraft equipped with the solar sail and low-thrust engine, using quaternions and kustaanheimo-stiefel variables. *Cosmic Research*, 52(6):450–460, 2014.

[52] C Sauer. Optimization of multiple target electric propulsion trajectories. In *11th Aerospace Sciences Meeting*, page 205, 1973.

[53] Marsil AC Silva, Daduí C Guerrieri, Angelo Cervone, and Eberhard Gill. A review of mems micropropulsion technologies for cubesats and pocketqubes. *Acta Astronautica*, 143:234–243, 2018.

[54] J Sims and S Flanagan. Preliminary design of low-thrust interplanetary missions. 1999.

[55] Jon Sims, Paul Finlayson, Edward Rinderle, Matthew Vavrina, and Theresa Kowalkowski. Implementation of a low-thrust trajectory optimization algorithm for preliminary design. In *AIAA/AAS Astrodynamics specialist conference and exhibit*, page 6746, 2006.

[56] Innovative Solutions In Space. ieps datasheet. Technical report, 2019.

[57] Sara Spangelo and Benjamin Longmier. Optimization of cubesat system-level design and propulsion systems for earth-escape missions. *Journal of Spacecraft and Rockets*, 52(4):1009–1020, 2015.

[58] Sara Spangelo, Derek Dalle, and Benjamin Longmier. Integrated vehicle and trajectory design of small spacecraft with electric propulsion for earth and interplanetary missions. *29th Annual AIAA/USU Conference on Small Satellites*, 2015.

[59] SRE-PA. Margin philosophy for science assessment studies. Technical report, European Space Research and Technology Centre, 2012.

[60] Robert L Staehle, Brian Anderson, Bruce Betts, Diana Blaney, Channing Chow, Louis Friedman, Hamid Hemmati, Dayton Jones, Andrew Klesh, Paulett Liewer, et al. Interplanetary cubesats: opening the solar system to a broad community at lower cost. 2012.

[61] D Syaifoel. Coupled optimization of systems design and trajectory for dual propulsion spacecraft. 2020.

[62] Hyperion Technologies. iadcs400 datasheet. Technical report, 2015.

[63] Hyperion Technologies. Cubecat datasheet. Technical report, 2019.

[64] Hyperion Technologies. Pm200 datasheet. Technical report, 2019.

[65] Hyperion Technologies. Cp400.85 datasheet. Technical report, 2019.

[66] Enpulsion Spacecraft Technology. Imf nano datasheet. Technical report, 2018.

[67] Gregory J Whiffen and Jon A Sims. Application of a novel optimal control algorithm to low-thrust trajectory optimization. *AAS 01-209*, 2001.

[68] CH Yam, DD Lorenzo, and D Izzo. Low-thrust trajectory design as a constrained global optimization problem. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 225(11):1243–1251, 2011.