

# Evaluation of Direct Search Algorithms to Trajectory Optimization for a Perpetually Flying Fixed-Wing Solar UAV Providing Network Coverage

by

Jared Miller

A dissertation submitted in partial fulfillment  
of the requirements for the degree of  
Master of Science  
(Computer Science and Information Systems)  
in The University of Michigan - Flint  
2021

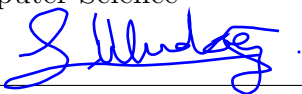
Committee:

**Advisor:**

Suleyman Uludag, Ph.D.

Associate Professor of

Computer Science

  
\_\_\_\_\_  
*Signature*

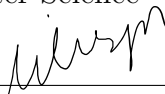
April 25, 2021  
\_\_\_\_\_  
*Date*

**Reader:**

Halil Bisgin, Ph.D.

Assistant Professor of

Computer Science

  
\_\_\_\_\_  
*Signature*

April 25, 2021  
\_\_\_\_\_  
*Date*



COMPUTER SCIENCE AND  
INFORMATION SYSTEMS

Copyright © 2021 Jared Miller

All Rights Reserved

# TABLE OF CONTENTS

<b>LIST OF FIGURES</b> . . . . .	iii
<b>ABSTRACT</b> . . . . .	iv
<b>CHAPTER</b>	
<b>I. Introduction</b> . . . . .	1
<b>II. Related Work</b> . . . . .	4
2.1 A Taxonomy of UAV Platforms . . . . .	5
2.2 Rotary-Wing . . . . .	7
2.3 Fixed-Wing . . . . .	8
2.3.1 Network Performance Analysis . . . . .	9
2.3.2 Long-Endurance Flight . . . . .	9
2.4 Our Contribution . . . . .	10
<b>III. System Model</b> . . . . .	11
3.1 System Parameter Selection . . . . .	11
3.2 Measurement . . . . .	15
3.2.1 Propulsion Power . . . . .	15
3.2.2 Solar Power . . . . .	16
3.2.3 Battery Charge . . . . .	17
3.2.4 Network Throughput Estimation . . . . .	18
3.3 Initial Results . . . . .	18
<b>IV. Trajectory Optimization</b> . . . . .	23
4.1 Representation Challenges and Selection . . . . .	24
4.2 Optimization Algorithms . . . . .	27
4.2.1 Nelder-Mead . . . . .	28
4.2.2 Particle Swarm Optimization . . . . .	30

4.3	Results . . . . .	31
4.3.1	Constraints . . . . .	31
4.3.2	Optimizer Performance . . . . .	33
<b>V. Conclusions and Future Work . . . . .</b>		<b>37</b>
<b>APPENDICES . . . . .</b>		<b>39</b>
A.1	Context . . . . .	40
A.2	Circular Arc Segment Equations . . . . .	43
A.2.1	Circular Arcs . . . . .	45
A.2.2	Lines . . . . .	48
B.1	Modules . . . . .	50
B.1.1	Aircraft.py . . . . .	50
B.2	Running the Optimizer . . . . .	53
<b>BIBLIOGRAPHY . . . . .</b>		<b>56</b>
<b>References . . . . .</b>		<b>57</b>

# LIST OF FIGURES

## Figure

1.1	Example UAV Platforms . . . . .	2
2.1	A taxonomy of UAVs. . . . .	6
3.1	Output of AirplaneDesign from [45] . . . . .	12
3.2	System Model . . . . .	14
3.3	Baseline Trajectories . . . . .	20
3.4	Baseline Trajectory Energy Balance and Throughput . . . . .	21
3.5	Ladder Trajectory - Throughput vs Altitude . . . . .	22
4.1	Examples of segments between the waypoints (0, 0) and (1, 1). . . . .	26
4.2	Altitude Schedule Variables . . . . .	27
4.3	Nelder-Mead Operations in $\mathbb{R}^2$ . . . . .	29
4.4	Particle Swarm Optimization Velocity Update . . . . .	30
4.5	Optimizer Performance for 0Wh and -500Wh Budgets . . . . .	34
4.6	Comparison of Estimated to Simulated User Throughput . . . . .	34
4.7	Optimizer Performance for Various Budgets . . . . .	35
A.1	Forces acting on the aircraft . . . . .	41

# ABSTRACT

Evaluation of Direct Search Algorithms to Trajectory Optimization for a Perpetually Flying Fixed-Wing Solar UAV Providing Network Coverage

by

Jared Miller

Advisor: Suleyman Uludag

Recent advances in battery and solar-cell technology have allowed for small-scale solar-powered fixed-wing aircraft to achieve perpetual-endurance flight, where they harvest enough energy to fly for long periods of time without recharging. We consider how this capability can be utilized in a networking context to enable rapid deployment of wireless Internet coverage in areas where it has been disrupted or is otherwise unavailable, providing a relay between portable ground-based devices such as cell phones and the currently developing Very-Low-Earth-Orbit satellite Internet layer. In particular, we investigate the impact of the aircraft trajectory on the level of service provided and the energy balance of the vehicle. We develop a framework for evaluating and optimizing these trajectories in three-dimensions and compare two optimization methods, Particle Swarm Optimization and the Nelder-Mead method, in optimizing the throughput of the trajectory while respecting the energy constraints implicit to providing perpetual-endurance flight.

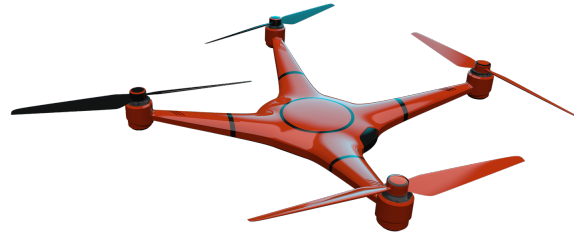
# CHAPTER I

## Introduction

The Unmanned Aerial Vehicle (UAV) is currently emerging as a highly capable platform for a number of applications. Also known as drones or Unmanned Aerial Systems (UAS), these platforms are characterized by their mobility and ability to be controlled remotely with no on-board human operator. While they have been used for military applications for many years, decreasing prices and increasing capabilities have allowed for many commercial and civilian applications as well, ranging from remote surveying and monitoring to retail delivery services [1]. An application area that has received attention in recent years, which also promises to be a future enabler for the drone ecosystem, is that of using UAV platforms to host telecommunications equipment to provide network coverage [2–7]. The high mobility, favorable line of sight characteristics at higher altitudes, along with the higher speed and lower cost of deployment are all factors that make a UAV platform an attractive alternative to ground based infrastructure in a variety of circumstances. For instance, UAV platforms may be used to quickly deploy network coverage to areas where ground infrastructure is unavailable, has been disrupted by natural disaster, or to augment existing infrastructure during periods of high demand.

Two UAV platforms are most commonly investigated for networking applications: fixed-wing platforms such as in Figure 1.1a and rotary-wing platforms as shown in Fig-





(a) Fixed-Wing Platform: NASA Helios [8]    (b) Rotary-Wing Platform: Illustration [9]

Figure 1.1: Example UAV Platforms

ure 1.1b (a further breakdown of UAV platforms is given in section 2.1). Each platform has distinct advantages and disadvantages over the other. Fixed-wing platforms have a much greater energy efficiency, and are better able to carry larger payloads, but must maintain a minimum speed to stay airborne. Rotary-wing platforms, on the other hand, can hover in place, but are generally less efficient and cannot carry as heavy of payloads. In a networking context, the ability to hover in place can improve the quality of communication as the system is less dynamic, leading to a more consistent signal level and less need to perform handover between different stations. However, the lower energy efficiency shortens the duration during which a rotary-wing platform can remain airborne, requiring more frequent refueling or recharging. Another interesting avenue that is more readily applicable to fixed-wing platforms is a synergy with photovoltaic (PV) energy-harvesting, as the wings provide a large area on which to place solar panels (and larger wings can also make the aircraft more efficient). This can allow the fixed-wing platform to fly continuously during the daytime. Given enough battery storage, it can also fly continuously through the night, see for example [10]. This level of persistence has clear advantages for networking applications, as outages or degradation due to frequent landing could be largely eliminated.

Even given this greatly increased endurance from the on-board PV, the aircraft will still operate on tight energy margins and may have to make sacrifices in net-

work performance for increased energy efficiency or vice versa under certain conditions. Selecting a trajectory that balances the energy efficiency of the craft against the needed levels of service for multiple users is a challenging task. Several strategies exist to increase the energy margins of a perpetual-endurance platform [11–13]. However these strategies are often detrimental to providing network service. While there have been several approaches investigated to increase the networking performance or the energy efficiency of a fixed-wing aircraft trajectory [14, 15], to the best of our knowledge there have been none which consider the conditions necessary for perpetual-endurance flight. The remainder of this work is organized as follows. Chapter II provides an overview of related works from the areas of aerial networking and perpetual-endurance aircraft design. Chapter III describes our system model and energy measurement framework. Chapter IV details the optimization methodology, chosen trajectory representation, and presents the results of the optimizations. Finally, Chapter V provides some concluding remarks and areas for future investigation. The appendices include Appendix A, which contains the derivation of the mathematical model used in our energy analysis, and Appendix B, which enumerates the various modules present in our code, their roles, and their relations.

## CHAPTER II

### Related Work

The areas of aerial networking and long-endurance flight have each seen much interest in recent years [2–7, 16, 17]. Rotary-wing platforms have attracted much of the interest in the networking arena due to their superior precision mobility and station-keeping characteristics. Several groups are also investigating improvements to low-altitude long-endurance fixed-wing platforms and the underlying technologies such as solar cells and energy storage, which permits them to carry heavier payloads and increases their overall energy robustness. With this continued improvement, the advantage of long duration flight may eclipse the mobility advantages of rotary-wing aircraft for many applications. To better put developments concerning these two platforms into perspective, we give a brief taxonomy of aerial platforms, followed by some of the recent investigations into rotary-wing networking platforms used in conjunction with energy harvesting systems, and then look at both the engineering and networking perspectives on the use of fixed-wing platforms. A listing of these works and the aspects they consider is given in Table 2.1, including whether rotary-wing or fixed-wing aircraft are considered, when trajectory is considered whether it is in two or three dimensions, whether energy use is considered, whether network services are considered, if and how energy-harvesting is considered, and what optimization technique is used when optimization is discussed.

Work	Wing Type	# Dims.	Considers Energy	Considers Network	Energy Harvesting	Optimization Technique <sup>1</sup>
Zeng et al. [18]	Rotary	2D	Yes	Yes	-	SCA <sup>2</sup>
Bozkaya et al. [19]	Rotary	3D	Yes	Yes	Station	Custom Alg.
Amorosi et al. [20]	Rotary	-	Yes	Yes	Station	MILP, GA <sup>2</sup>
Chiaraviglio et al. [21]	Rotary	-	Yes	Yes	Station+Grid	BBSR (novel) <sup>2</sup>
Sun et al. [22]	Rotary	3D	Yes	Yes	On-Board	Various
Sekander et al. [23]	Rotary	3D	Yes	Yes	On-Board	-
Huang et al. [24]	Fixed	2D	Yes	-	On-Board	-
Leutenegger et al. [25]	Fixed	3D	Yes	-	On-Board	-
Meyer et al. [26]	Fixed	-	Yes	-	On-Board	-
Oettershagen et al. [10, 27–29]	Fixed	-	Yes	-	On-Board	-
Bolandhemmat et al. [11]	Fixed	3D	Yes	-	On-Board	Various
Marriott et al. [12]	Fixed	3D	Yes	-	On-Board	Greedy
Zeng et al. [14]	Fixed	2D	Yes	Yes	-	GA <sup>2</sup>
Qiu et al. [15]	Fixed	2D	Yes	Yes	-	SCA <sup>2</sup>
Anicho et al. [30]	Fixed	3D	Yes	Yes	On-Board	-
Anicho et al. [31]	Fixed	-	-	Yes	-	-
Anicho et al. [32]	Fixed	-	-	Yes	On-Board	-
Anicho et al. [33]	Fixed	-	-	Yes	On-Board	Q-Learning
This Work	Fixed	3D	Yes	Yes	On-Board	Various

Table 2.1: An Overview of Related Works

## 2.1 A Taxonomy of UAV Platforms

A number of different approaches for providing network access from aerial platforms has been explored. Figure 2.1 gives a breakdown of the most common approaches. Platforms are largely divided into Low Altitude Platforms (LAPs) and High Altitude Platforms (HAPs), where the dividing altitude is, as a general rule of thumb, around 10 kilometers (though this definition varies by jurisdiction). LAPs include rotary-wing aircraft (e.g. quad-copter, hex-copter) and airships<sup>3</sup> (e.g. Blimps, Zeppelins), while HAPs include balloon-based platforms. Fixed-wing aircraft (e.g. a traditional airplane with wings and forward propulsion) can fit into either category. Both airships and balloons use a lifting gas such as hydrogen, helium, or heated air to fly (or float, as the case may be), while fixed-wing and rotary-wing aircraft must actively expend energy to maintain their altitude. LAPs are generally smaller and easier to deploy but have lower endurance (measured in hours). HAPs are generally

<sup>1</sup>[34] gives a good overview of various trajectory optimization techniques. Though it is in the context of spacecraft trajectories, many of the approaches are still applicable here.

<sup>2</sup>Successive Convex Approximation (SCA), Mixed-Integer Linear Programming (MILP), Genetic Algorithm (GA), Balance energy Bought Sold and throughput Revenue (BBSR)

<sup>3</sup>Some sources, such as [7], also classify airships as a HAP.

larger and harder to deploy, but have greatly increased endurance (measured in days to months for balloons and airships), either because they are passively floating or via on-board energy harvesting<sup>4</sup>.

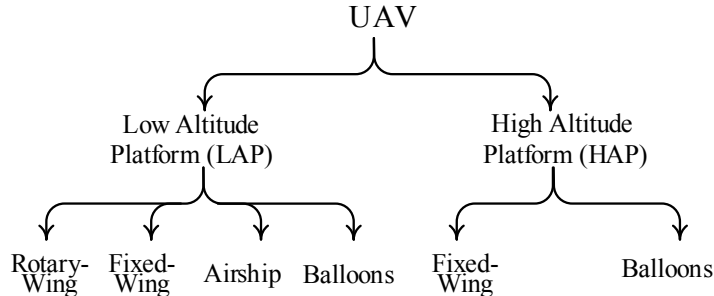


Figure 2.1: A taxonomy of UAVs.

While rotary-wing UAVs have increased flexibility in navigating complex spaces, as well as the ability to hover in place, they are often less energy efficient than a fixed-wing platform and must land more frequently to recharge. They are also limited in their ability to support higher-massed payloads, large battery stores, or energy-harvesting equipment<sup>5</sup>. For longer term network applications, this property of rotary-wing UAVs exacerbates the requirement of increased redundancy to avoid outages, as one must have a second UAV on stand-by for when the first needs to land and recharge. The fixed-wing platform addresses to some degree the efficiency and mass-limit concerns of a rotary-wing UAV, at the expense of decreased flexibility in that it must maintain a minimum flight speed. However, although though less frequently, the fixed-wing platform would still need to land to refuel or recharge. In recent years, several groups have attempted to eliminate the need to land to recharge as well, with the inclusion of on-board energy harvesting, generally PV, and sufficient battery capacity to fly through the night. Several such platforms have been designed

---

<sup>4</sup>Extending beyond this, we will also point out satellite platforms, which are very hard to deploy and adjust, but have extremely long to virtually infinite endurance compared to the usefulness of their payloads.

<sup>5</sup>This is not to say that PV has not been included on rotary-wing UAVs as well [22, 23], but for the moment this can only increase their endurance during the day, and they are unable to carry sufficient energy to fly through the night.

and created with greatly increased endurance, to the point that they can remain airborne continuously (we adopt their terminology of “perpetual-endurance”) during much or all of the year [10, 11]. Some groups are currently exploring the application of this capability to network platforms [35], although several more attempts have been made only to be found commercially unviable at present [36, 37].

## 2.2 Rotary-Wing

There is an abundance of existing literature analyzing rotary-wing aircraft providing network coverage, but here we focus on just a few works that use the platform in conjunction with an energy harvesting system. The first approach one can take is to have remote ground charging stations with energy harvesting where the UAVs return to between providing service. Such an approach is the context for [20] and [21] (from related authors). In the former, the authors investigate various approaches to maximize the energy of the system subject to providing coverage with a Mixed Integer Linear Programming (MILP) as well as a Genetic Algorithm (GA). In the latter, the authors develop models to perform joint optimization of the energy consumption of the system alongside the throughput revenue. In both cases, the challenge lies in scheduling the UAVs trips to and from the charging station, while still providing sufficient coverage to meet their goals. Another work using this approach is [19], though here a more algorithmic approach is used to optimize placement to maximize coverage alongside flight endurance. On the other hand, in [22] the authors investigate a rotary-wing platform with integrated PV, and investigates the tradeoff between maintaining a low altitude to provide network coverage and a high altitude to harvest more energy (i.e. to be above the clouds). Their system is presented as providing continuous flight during the day, though it cannot sustain long-term flight

at night<sup>6</sup>. A combination approach is considered in [23], where the rotary-wing craft is equipped with on-board energy harvesting and a ground charging station is also available. However, this work is largely the construction of a statistical model for harvested energy vs. outage time and doesn't present any concrete examples of system performance. Throughout, we can see that while energy harvesting has the potential to augment rotary-wing aircraft, there is still a need for ground-based infrastructure to provide energy at certain times.

## 2.3 Fixed-Wing

On the fixed-wing side of the literature, there is a division between investigations into network optimization and long-endurance flight, with little work combining the two. This is perhaps understandable as the current state of technology greatly limits the payload capacity of such vehicles relative to their size, but this limit is largely due to the battery capacity required to operate through the night. Increasing battery energy density (energy stored per kg) has a great impact on this limit, as the amount of energy stored can be increased without increasing the overall size of the aircraft. As the market for electric energy storage is expected to greatly expand over the next decade [39], especially in the transportation sector which also benefits from denser storage, one can expect this situation to continue improving. Indeed, some groups are already developing technology which claims to almost double the energy density of current batteries [40]. In any case, the works on these two sides of the literature paint an interesting picture of future capabilities.

---

<sup>6</sup>This work does assume (most notably) a high solar cell efficiency of 40% which is not physically possible with commonly used single-junction solar cells which have a maximum theoretical efficiency in the low 30% [38] and implies that the presently much more costly multi-junction solar cells are considered.

### 2.3.1 Network Performance Analysis

The optimization of a fixed-wing trajectory with respect to energy efficiency (network utility per unit of energy used) is investigated in [14]<sup>7</sup>. The authors present simple analytical models for evaluating the energy consumption (including a derivation in their appendices) and the available throughput of a given constant-altitude fixed-wing trajectory with a single ground user. Approaches for both unconstrained (no limits on velocity or power use, physically impossible trajectories produced) and constrained trajectories are given for the rate-maximization, energy-minimization, and efficiency-maximization problems, with corresponding statistics for each. This gives us a simplified base upon which other features might be added to expand upon the capabilities or realism of the setup (in our case, variable-altitude, multiple users, and energy harvesting for perpetual-endurance flight). The trajectory formulation used in [14], though, is extremely granular (discretized to 0.2 second time steps) and thus may not be suitable for long-duration optimization. In [30–33] the authors investigate aspects of HAPS placement in a variety of circumstances, as well as routing approaches and vehicle replacement scenarios.

### 2.3.2 Long-Endurance Flight

NASA investigated solar powered aircraft as early as the 1980's [41]. This program continued for several decades, though experimental flights ended when the prototype was destroyed in flight in 2003 [42]. In more recent years (2011+), several other groups have designed, built, and flown other prototype aircraft capable of long-endurance flight. [25] provides an overview of some of these earlier works, as well as some of the modelling for such platforms. This work is expanded on by a related group in a series of papers that resulted in an 81-hour continuous flight [10, 27–29]. While the payload mass and power capacity of their aircraft is extremely limited, the capability

---

<sup>7</sup>[18] provides an analogous analysis for rotary-wing.



for perpetual-endurance flight in a platform massing under 10 kg is impressive.

In the past couple years, a group at Facebook has been investigating various methods for the optimization of the energy balance of a solar powered HAP UAV. Various approaches were attempted in [11] such as the Interior Point method, the Nelder-Mead method, and a machine learning approach based on an Adaptive Neuro-Fuzzy Inference System (ANFIS). Given the behaviors observed in that work, [12] investigates the use of a greedy approach to perform similar planning with a much lower computational cost. The work is likely in relation to Facebook’s project Aquila [43], which for the moment has been cancelled [36]. One interesting technique exploited in both works is to increase altitude during the day when excess solar power is readily available and coast back down during the night when solar power is unavailable. This technique provides an avenue for storing additional energy as gravitational potential, which can decrease the battery mass required to fly through the night, and is also discussed in [13].

## 2.4 Our Contribution

While much work has been done both in the areas of perpetual-endurance flight and aerial networks, relatively little has been done combining these two for LAPs. In particular, no other works were found investigating optimization of a fixed-wing aircraft trajectory providing network coverage under perpetual-endurance flight conditions. This work attempts to address this niche by developing a framework for representing and analyzing such trajectories (with the inclusion of a variable-altitude/3D component), as well as for the optimization of such trajectories. In particular, we investigate the Particle Swarm Optimization and Nelder-Mead methods to perform this optimization, though the framework allows for relatively easy integration of other optimization algorithms<sup>8</sup>.

---

<sup>8</sup>An overview of the implementation is given in Appendix B, and the code is available at [44]

## CHAPTER III

# System Model

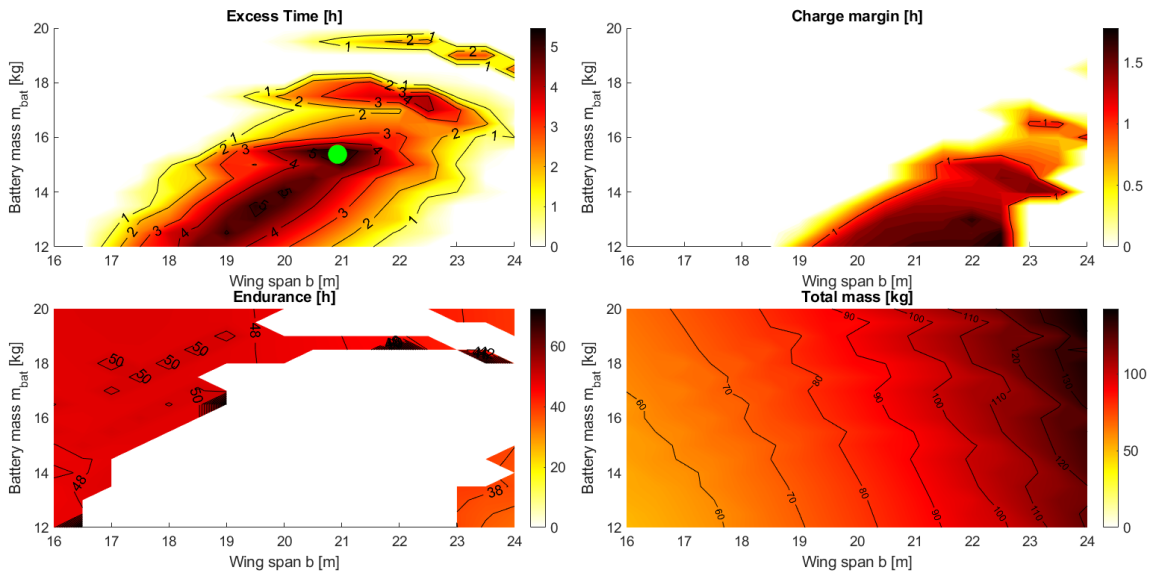
### 3.1 System Parameter Selection

The hypothetical platform we will be analyzing will be an extension of that designed in [10]: an unmanned fixed-wing aircraft with solar cells on the wings to provide on-board energy harvesting and sufficient battery capacity to provide energy throughout the night when no solar power is available. Such a platform, often referred to as a Low-Altitude Long-Endurance (LALE) UAV, is capable of harvesting sufficient energy in flight that it does not need to land to recharge. However, depending on the configuration of the aircraft and the specific circumstances, this may only be possible at certain latitudes or during certain times of the year, as these both have a large impact on the amount of harvestable energy available. Furthermore, these aircrafts have particular Size, Weight, and Power (SWAP) constraints on their payloads, and special care must be taken with respect to these limits. For this analysis, we generate such a configuration for our desired payload characteristics using the code [45] related to [10]. This process is described below.

The MATLAB module `AirplaneDesign` from [45] can be configured with desired system parameters such as payload mass, payload power, battery energy density, latitude, time of year, and several others. Given these, ranges of configurations for wingspan and battery mass can be evaluated in bulk and the resulting performance

plotted (see Figure 3.1) to give a sense of which parameter combinations produce good results in terms of endurance. The following metrics are produced when perpetual-endurance flight is achieved based on the input parameters:

- Minimum State of Charge (MSoC) of the batteries: indicates the minimum amount of energy available in the batteries over the course of the day. For example, the batteries might dip to 40% before sufficient sunlight is available to charge them.
- Excess Time (Top Left of Figure 3.1): duration the airplane could fly if sunlight was unavailable at the start of a day. Effectively, if there is a very cloudy day this tells us how many hours past sunrise the plane could fly.
- Charge Margin (Top Right of Figure 3.1): the duration during which the batteries are full, or how much excess “sun time” there is. This determines how much “non-sun time” (e.g. cloudiness) the plane can experience per day before perpetual-endurance becomes unattainable.



If perpetual-endurance is not reached then the total endurance (Bottom Left of Figure 3.1), or total time the aircraft can fly starting with a full battery, will be available. The total mass of the aircraft (Bottom Right of Figure 3.1) is always given, which includes the battery mass as well as the structural mass required by that particular wingspan. With this data available we selected our configuration by maximizing the total excess time (Lime Circle in Figure 3.1). The parameters used for this process and the resulting aircraft configuration parameters and resulting values are shown in Table 3.1.

Table 3.1: Aircraft Parameters and Endurance Values

Design Parameters		Airplane Configuration and Values	
Name	Value	Name	Value
$P_{payload}$	109 W	Wingspan	21 m
$m_{payload}$	6 kg	$m_{bat}$	15.5 kg
$P_{prop,max}$	600 W	AR	20.5
Day of Year	Dec. 1	MSoC	26.23%
Latitude	36° N	$T_{exc}$	5.47 h
Longitude	84° W	$m_{struct}$	50.16 kg
$h_0$	1000 m	$m_{total}$	90.16 kg
Energy density	650 Wh/kg <sup>1</sup>	$P_{level}$	468.29 W

In order to provide for network coverage, we consider the inclusion of the equipment necessary to create an LTE cell using the specifications of [46] as a guide. We also consider some form of satellite backhaul, which several commercial entities are currently in the process of developing [47] or actively deploying [48]. Thus, the system acts as a relay between easily human-portable devices such as cell-phones and the satellite layer, which requires heavier and less portable equipment to communicate with. This architecture is depicted in Figure 3.2. Thus, the aircraft is capable of providing network services to ground users to augment existing infrastructure or create coverage in areas where infrastructure is disrupted or otherwise unavailable. The inclusion of on-board energy harvesting allows the platform to greatly exceed the en-

---

<sup>1</sup>We are assuming that certain in-development battery technologies come to fruition to achieve this density [40].

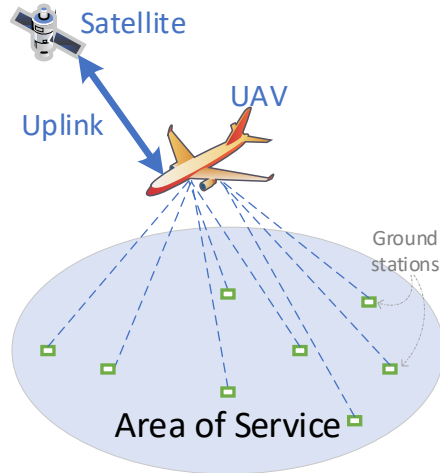


Figure 3.2: System Model

duration of many other airborne communication platforms such as rotary-wing UAVs, decreasing the requirement for frequent landings and multiple platforms to provide continuous coverage. Such a platform is then evaluated in a simulated environment with multiple ground users under simple geometric flight-area constraints.

The primary difference from other works investigating similar platforms in a networking context [14, 15, 49] is the goal of achieving perpetual-endurance flight and the inclusion of a variable altitude. Such perpetual-endurance flight is discussed in several other works [10, 11] from an engineering perspective, but without the inclusion of a networking context. The inclusion of variable altitude allows for additional energy to be stored as gravitational potential energy, augmenting the battery stores, and is discussed (again only from an engineering or efficiency optimization perspective) in works such as [11, 13]. This technique may allow the aircraft to extend its window of perpetual-endurance viability further into the winter months with lower specifications while being unneeded in the summer months, or to increase energy reserves in advance of poor weather conditions.

## 3.2 Measurement

### 3.2.1 Propulsion Power

In order to perform the desired optimization trials, we first had to develop a framework for measuring the energetic performance of a given trajectory. From a mathematical perspective, this consisted of two parts: finding a representation for the trajectory, and measuring the energy and network characteristics of that trajectory. Initially, we created two different types of trajectory segment from which to compose the trajectory: a linear segment that could ascend or descend and a circular arc segment that maintained a constant altitude. Later, a more general segment representation was developed that could represent both of these specific cases, in addition to an ascending/descending circular arc. For each segment, we assume a constant velocity. Specifically, we assume a constant Angle-of-Attack (AoA), which determines the required velocity to maintain the course. From this we can determine the position at any particular time as well as the required power. The relevant formulae are:

Required power<sup>2</sup>:

$$P_{prop} = |T| \cdot v \quad (3.1)$$

For a circular arc segment:

$$\sin(\phi) = \frac{m}{r [\tan(\theta + \alpha)(L_1 \sin(\theta) + D_1 \cos(\theta)) + (L_1 \cos(\theta) - D_1 \sin(\theta))]} \quad (3.2a)$$

$$v^2 = \frac{rg \sin(\phi)}{\cos(\phi)} \quad (3.2b)$$

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (3.2c)$$

For a linear segment (circular arc with an infinite radius):

$$v^2 = \frac{mg}{[L_1 \sin(\theta) + D_1 \cos(\theta)] \tan(\theta + \alpha) + [L_1 \cos(\theta) - D_1 \sin(\theta)]} \quad (3.3a)$$

---

<sup>2</sup>we assume no powered deceleration, thus the thrust is never negative

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (3.3b)$$

The full derivation for these is presented in Appendix A.

$v$	Velocity (m/s)	$T$	Thrust (N)
$m$	Aircraft Mass	$r$	Turn Radius (m)
$\phi$	Roll Angle (to center)	$\theta$	Angle of Ascent
$\alpha$	Angle of Attack	$\rho$	Air Density (kg/m <sup>3</sup> )
$S$	Wing Surface Area (m <sup>2</sup> )	$g$	Standard Gravity (m/s <sup>2</sup> )
$L_1$	$1/2 \cdot \rho \cdot C_L \cdot S$	$C_L$	Coefficient of Lift
$D_1$	$1/2 \cdot \rho \cdot C_D \cdot S$	$C_D$	$C_{D,wing} + C_{D,par} + C_{D,ind}$
$C_{D,wing}$	Wing Drag Coefficient	$C_{D,par}$	Parasitic Drag Coefficient
$C_{D,ind}$	Induced Drag Coefficient	$P_{prop}$	Propeller power (W)

Table 3.2: Symbols and Expressions for Equation 3.2 and Equation 3.3.

### 3.2.2 Solar Power

Now that we can determine the power required for the propulsion system for various trajectory segments, we must also be able to determine the solar power collected. This is strongly dependent on the attitude of the aircraft (specifically the solar cells) relative to the direction of the sun. As the solar panels are mounted on the wings of the aircraft (we assume they are facing directly vertical in the rest orientation for simplicity), power input will be highest at local solar noon when the sun is highest in the sky. Orienting the panels towards the sun will increase the energy collected over time, which might be accomplished either by banking towards the sun (and thus turning), flying towards the sun and descending, flying away from the sun and ascending, adjusting the AoA, or some combination thereof. Thus, each segment must compute its relative orientation over time, encoded as the azimuth (in degrees east of north) and tilt (degrees below horizontal). Given this information we can compute the solar power as in [10]

$$P_{solar}^{nom} = I_{solar}(\varphi_{lat}, h, \delta, t, \vec{n}_{sm}) \cdot A_{sm} \cdot \eta_{sm} \cdot \eta_{mppt} \quad (3.4)$$

where  $I_{solar}$  is the incident solar radiation as a function of geographical latitude  $\varphi_{lat}$ , altitude  $h$ , current day-of-year  $\delta$ , local time  $t$ , and solar module normal vector  $\vec{n}_{sm}$  (encoded as the aforementioned azimuth and tilt). Further,  $A_{sm}$  is the area of the solar modules,  $\eta_{sm}$  is the solar module efficiency, and  $\eta_{mppt}$  is the efficiency of the maximum power point tracker module. In our implementation, the computation of  $I_{solar}$  is provided by [50].

In order to determine the azimuth and tilt of the solar modules, one can take the base orientation of the panels and apply a rotation matrix from the current attitude (see Equation A.6) and heading to get the normal vector. The azimuth is then simply the angle of this vector as projected on the XY plane, and the tilt is the angle of the vector above the XY plane. In the event that the normal vector remains vertical (no pitch or roll), the azimuth can remain undefined (use any value) and we take a tilt of  $0^\circ$ .

### 3.2.3 Battery Charge

Given the above calculations, we have a net power of

$$P_{net} = P_{solar}^{nom} - P_{prop} - P_{pld} \quad (3.5)$$

When  $P_{net} > 0$  the batteries can be charged, and when  $P_{net} < 0$  energy must be taken from the batteries. In the event the batteries are full, no more energy may be stored. We take the battery charge and discharge coefficients of 0.95 and 1.03 respectively from [10], which determines the efficiency of storing energy in the batteries. [10] also defines an exponential charge limiting above a 90% State of Charge, but we do not consider that here for ease of computation.



### 3.2.4 Network Throughput Estimation

To estimate network throughput we use the equations presented in [14] for instantaneous channel capacity in bits/second:

$$R(t) = B \log_2 \left( 1 + \frac{P\beta_0}{\sigma^2 d^2} \right) \quad (3.6)$$

Where  $B$  is the channel bandwidth in Hz,  $P$  is the transmit power,  $\beta_0$  is the channel power at 1 meter,  $\sigma^2$  is the noise power, and  $d$  is the distance between transmitter and receiver. The noise power is calculated as

$$\sigma^2 = N_0 B \quad (3.7)$$

Where  $N_0$  is the noise power spectrum density in dBm/Hz. Note that  $P$ ,  $\beta_0$ ,  $\sigma^2$ , and  $N_0$  are frequently in a logarithmic/decibel scale, and should be multiplied/divided on a linear scale (or added/subtracted on the logarithmic scale, respectively) for these equations. Additionally, to validate our estimation is indicative of what we might see in a real system, we evaluate the throughput in a simulated environment, considering the impacts of multiple users and the characteristics of our satellite backhaul. These simulations, performed in NS3 [51], model each user as producing some level of traffic according to a Poisson Pareto Burst Process (PPBP) [52] such that the network link is saturated. The parameters for the radio equipment, satellite link, user distribution, and traffic generation (per-user) are given in Table 3.3. All other parameters are taken from the NS3 defaults.

## 3.3 Initial Results

In order to get a rough feeling for the various measures of trajectories in our feasible space, we look at a small number of hand-selected trajectories. Depicted

Table 3.3: Other Simulation Parameters

<b>Description</b>	<b>Value</b>
Transmission power	30 dBm
Channel bandwidth	4.5 MHz
Noise power spectrum density	-174 dBm/Hz
Backhaul Data Rate	1 Gbps
Backhaul Latency	20 ms
Number of Ground Users	5
User Distribution Radius	5 km
PPBP Mean Burst Arrivals	1
PPBP Mean Burst Time Length	1 second
PPBP Burst Intensity	1 Mbps

in Figure 3.3, these are called the Circle, Bowtie, and Ladder trajectories. The Circle trajectory is simply a circular path at a constant altitude. The Bowtie trajectory is a path around two circles, crossing in the center, also at a constant altitude<sup>3</sup>. Finally, the Ladder trajectory has the same path as the Bowtie trajectory in the XY (horizontal)<sup>4</sup> plane, but ascends during the day (when excess solar power is available) and descends at night, as is shown in figure Figure 3.3c. A perspective view of the ladder trajectory is shown in Figure 3.3d.

We can then derive the energy and network characteristics of these trajectories using the methods described above. The energy characteristics of the three trajectories are plotted over a 48-hour period (to better show the pattern over multiple days) below in Figure 3.4a, Figure 3.4b, and Figure 3.4c, with the throughputs plotted in figure Figure 3.4d over a 24-hour period. The energy plots can be rapidly generated, while the throughput<sup>5</sup> plot represents the average over 30 runs in NS3 for each trajectory<sup>6</sup>, with two standard deviations shaded for each curve.

<sup>3</sup>This might also be described as a “figure-eight” or “infinity” shape.

<sup>4</sup>Note that the X dimension represents West to East, and the Y dimension represents South to North.

<sup>5</sup>Note that in this instance we were investigating the upload rates, and not the download rates.

<sup>6</sup>In our setup, the NS3 simulations ran approximately in real-time (1 second = 1 simulation

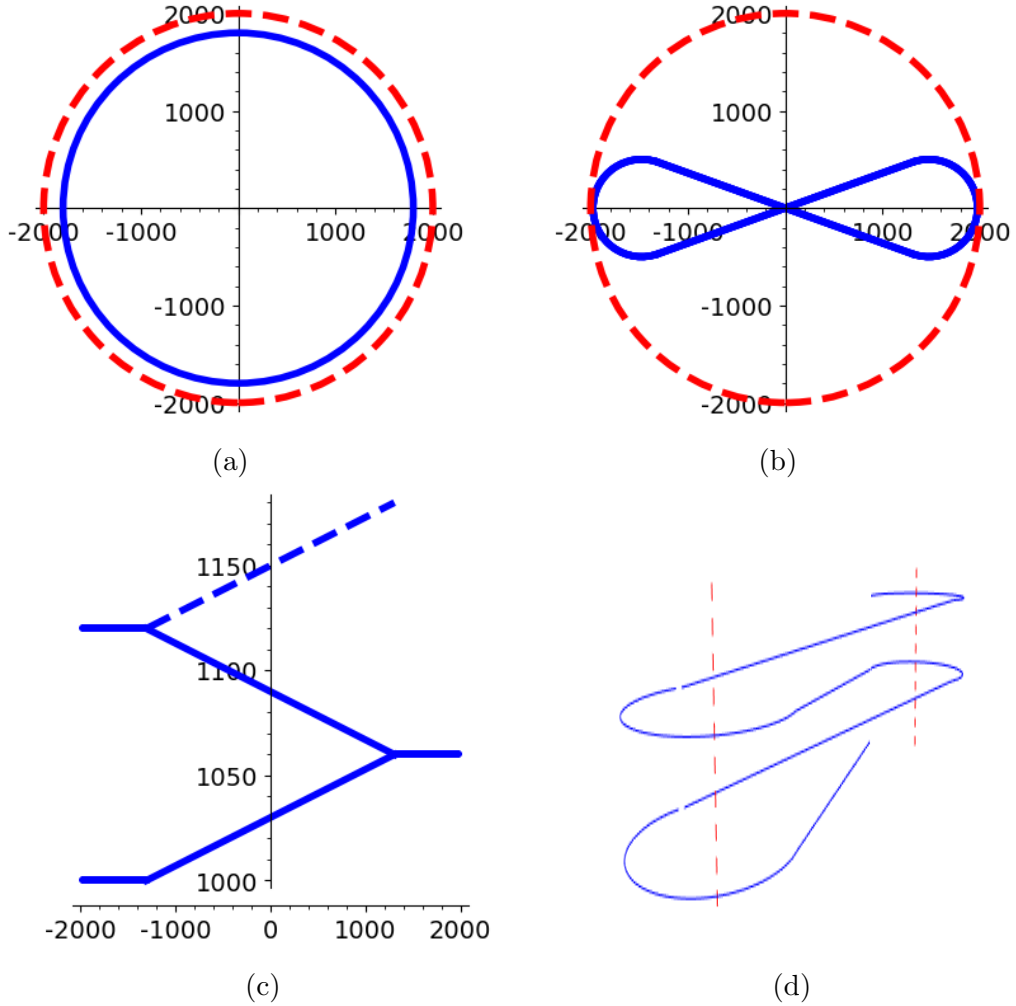


Figure 3.3: Baseline Trajectories. (a) Circle Top View, (b) Bowtie and Ladder Top View, (c) Ladder Side View, (d) Ladder Perspective View. The dashed line represents (a, b) the flight area radius, (c) the continuation of the flight path, (d) and the vertical centers of the trajectory. All units in meters.

We can see that an appreciable amount of energy was stored in altitude for the ladder trajectory, around 20% of the capacity of the batteries. Additional energy was also harvested while at higher altitude, though much of this benefit is lost as the batteries are fully charged. Finally, take note that more power is needed at higher altitudes due to the decreased air density. The throughput plot reveals that the circle and Bowtie trajectories have very similar throughput rates, with a slight advantage to the Bowtie trajectory. The Ladder trajectory begins and ends the period with a

---

second) and are single-threaded.

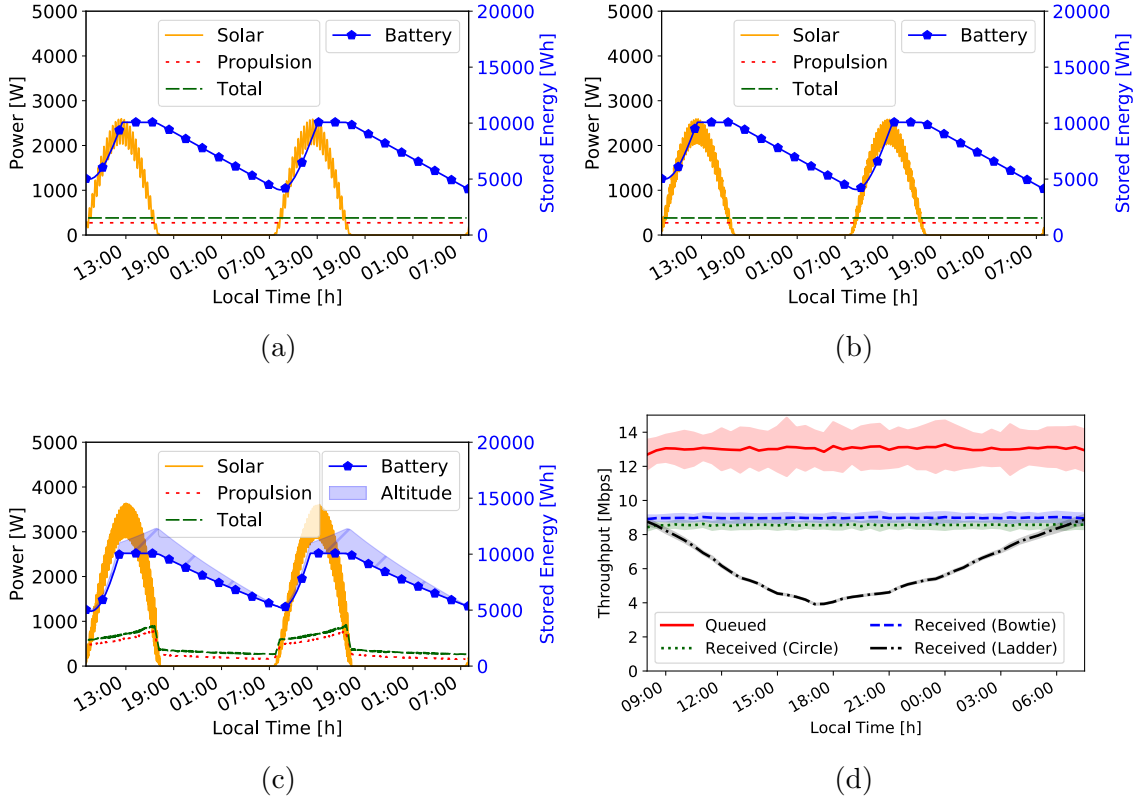


Figure 3.4: Trajectory Energy Balance and Throughput. (a) Circle Energy, (b) Bowtie Energy, (c) Ladder Energy, (d) Comparison of Throughput (upload).

throughput similar to the other two trajectories, but this diminishes throughout the day as the aircraft approaches its highest point in the evening. This relationship is more easily visible in Figure 3.5. Summary values for each of these plots is available in Table 3.4.

We can see from these few examples that there is a direct tradeoff in using altitude to store energy when our target application depends on distance to targets on the ground: the higher we go the less throughput is available. However, we also see that even at a fixed altitude there are different paths we can take that might increase either our energetic or networking performance. In Chapter IV we will take a look at a couple methods for finding trajectories that have improved network conditions over predefined trajectories while considering the constraints on energy balance, as well as other constraints on the aircraft that we did not reach here, such as thrust

and velocity limits.

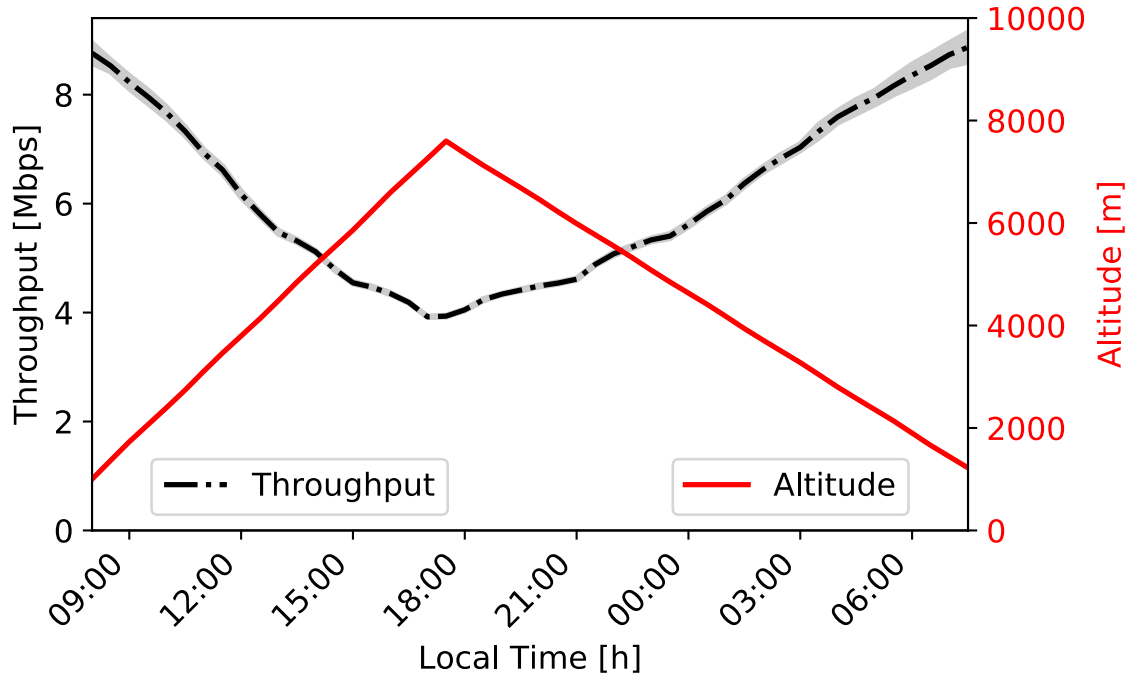


Figure 3.5: Ladder Trajectory - Throughput vs Altitude

Table 3.4: Baseline Simulation Results. Throughput figures represent the average system throughput.

Trajectory	MSoC	Min. Throughput	Mean Throughput	Max Throughput
Circle	40.1%	8.42 Mbps	8.56 Mbps	8.63 Mbps
Bowtie	40.3%	8.91 Mbps	8.98 Mbps	9.03 Mbps
Ladder	52.8%	3.92 Mbps	6.13 Mbps	8.87 Mbps

## CHAPTER IV

# Trajectory Optimization

The optimization of fixed-wing aircraft trajectories has been investigated at great length in the general sense, as well as for optimizing energy balance in HALE systems [11, 12] and in improving the characteristics of aerial networks [2], among many other topics. However, there is relatively little literature on optimizing a fixed-wing trajectory of a aircraft with perpetual-endurance from a networking perspective, as technology is only currently developing to the point where such systems are commercially viable. Here we will apply a parallel<sup>1</sup> variant of the Nelder-Mead method [53] (sometimes called the non-linear simplex method), as well as the Particle Swarm Optimization (PSO) algorithm originally described in [54]. The Nelder-Mead method was previously used to optimize the efficiency of the trajectory of a perpetual-endurance flying aircraft in [11]. The end goal of the optimization here is to maximize the mean throughput of our ground users subject to constraints on the net energy over the course of the day and various physical and flight volume constraints, given our aircraft specifications from section 3.1 and a particular time of year and geographic location

---

<sup>1</sup>Here “parallel” means multiple vertices are updated in each iteration, and the evaluations can thus be spread across multiple processor cores as they are independent.

## 4.1 Representation Challenges and Selection

The first item to address in all of these optimization approaches is how the trajectory is represented in the solution space that the optimization algorithm is working in. All of the optimization algorithms we use here have some encoding of the trajectory as a series of values (i.e. a vector) as  $\vec{x} = x_0, x_1, \dots, x_n$ , which is used to construct the trajectory, gather relevant statistics, and produce the value of an objective function  $f(\vec{x})$  which is used to drive the optimization algorithm. A number of characteristics were conjectured to be useful in the process of developing the representation, namely

- The trajectory should not be made of (exclusively) very short segments. For example, 20-second segments would imply  $4,320 \cdot \text{variables}/\text{segment}$  variables to represent a 24-hour period. Larger numbers of dimensions led to longer evaluation times and less improvement in our early optimization attempts.
- The trajectory should have a fixed length input. That is, we should not need to add or remove variables during the optimization process, and all trajectories should have the same input “shape”.
- The trajectory representation should produce implicitly “smooth” trajectories. That is, there should be no “teleportation” or sharp corners. Mathematically, the trajectory should be continuous and differentiable.
- The trajectory representation should produce trajectories with a duration equal to one 24-hour period, as this is the duration over which we are evaluating the trajectory.

A number of different representations were investigated<sup>2</sup>, and two representations were finally considered. The first, inspired by the representation in [11], consists of a number of segments of equal time duration (e.g. every segment is 20 seconds long),

---

<sup>2</sup>Many of the other candidates remain in the code, see subsection B.1.1.2

with three variables for each segment: AoA, pitch, and roll. This has the advantage of guaranteeing a particular duration for the whole trajectory, simply by dividing that duration into fixed length segments. It is also implicitly smooth, as the variables determining the rates of change (AoA, pitch, and roll determine velocity, altitude change rate, and heading respectively) are directly set. However, we determined that this approach does not scale well to long durations. Firstly, dividing the 24-hour period up into short time spans that can still control the flight dynamics finely enough required a large number of dimensions. For 20-second segments, it will be  $3 \cdot \frac{24 \cdot 60 \cdot 60}{20} = 12,960$  dimensions. Secondly, and perhaps more importantly, the values near the start of the trajectory have an impact on every position after them. Thus, a slight change in altitude or heading early on will impact every subsequent point. This discourages changes to the early points, as they may make the trajectory infeasible with respect to the flight-volume constraints, or require traversing a valley in the objective function, requiring many other points to change to return to feasibility/optimalty.

The second, and the final, representation we settled on for this work somewhat lessens these shortcomings. The representation consists primarily of a sequence of “waypoints” consisting of a position and heading, as seen in Figure 4.1. For each adjacent pair of waypoints, two circular arcs are constructed which are tangent to the headings, incident to the waypoint locations, tangent to each other, and of equal radii<sup>3</sup>. As there are multiple possible solutions to this description, we choose the shortest path that involves no change in heading at the tangent point of the two arcs. Thus, each segment is determined only by those two waypoints near it, and each waypoint only impacts the segments adjacent to it. Additionally, the AoA for each segment is stored in the representation. Thus, we eliminate the large impact-at-a-distance of any dimension as seen in the first representation. However, you may have noticed that this representation does not implicitly create a trajectory that is

---

<sup>3</sup>This is very similar to biarc interpolation as described in [55], though we arrived at a slightly different formulation.



24-hours in length. To correct this, we scale the trajectory in the XY plane to reach this 24-hour duration. This somewhat negates the first benefit of this representation, as a particularly large segment may cause all others to shrink, but the impact is not nearly as drastic as before. To implement this scaling, we utilized the fixed-point iteration method<sup>4</sup>, determining the duration of a candidate trajectory proportional to 24-hours and scaling the XY coordinates by the inverse of that value. This converges to 24 hours within a small number (less than 10) of iterations.

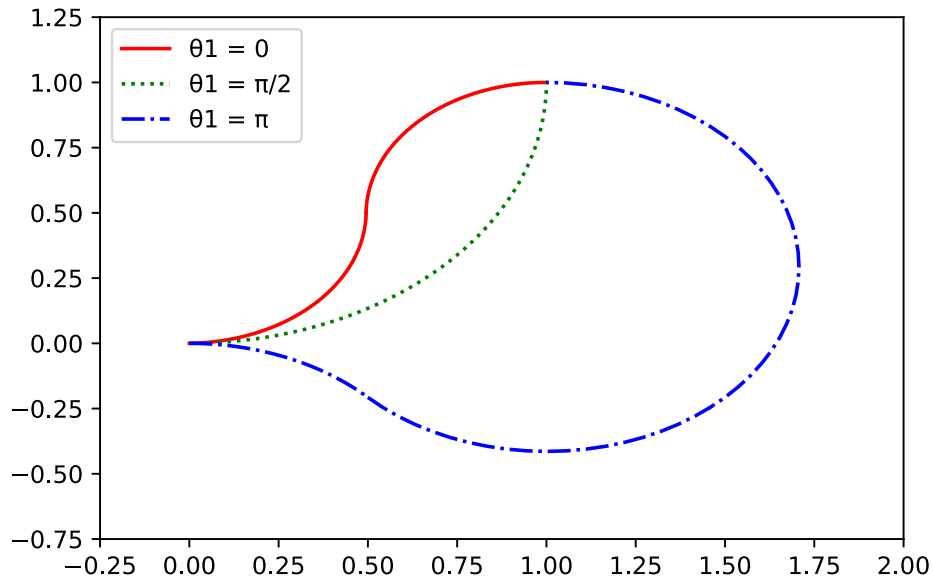


Figure 4.1: Examples of segments between the waypoints  $(0, 0)$  and  $(1, 1)$ . The heading of the first waypoint is 0 radians, and the heading of the second waypoint is indicated in the legend.

Even this, however, did not create a problem that was amenable to optimization. The primary factor determining the throughput available to ground users is the altitude of the aircraft. As such, in order to avoid becoming infeasible due to too rapid altitude changes, multiple components of the vector must change in parallel in order for this metric to improve (i.e. we cannot move a single point down without adjusting its neighbors). While this may eventually occur, it was not observed in any of the trial runs. Instead only small improvements were made and the algorithm

<sup>4</sup>See for example [56] section 2.2.

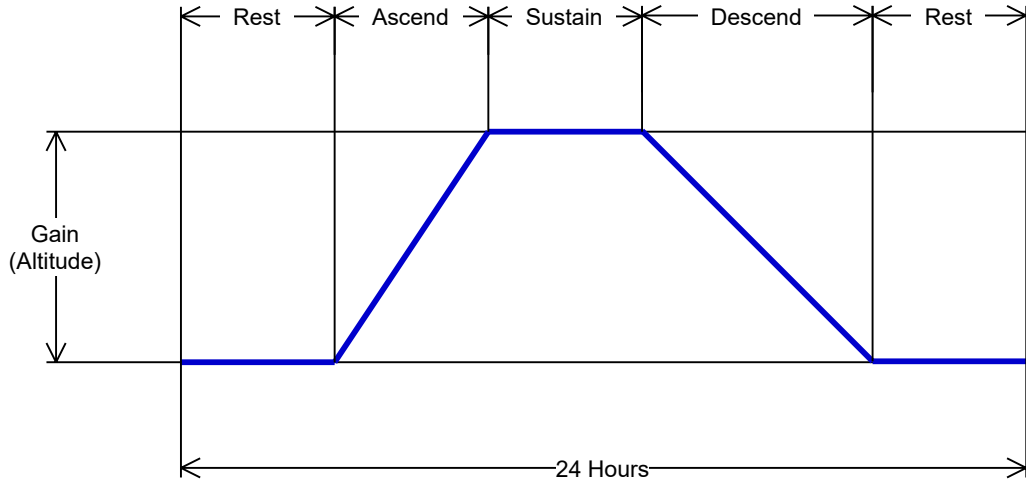


Figure 4.2: Altitude Schedule Variables

appeared to remain stuck in a local optimum almost immediately. To remedy this, we adopted an “altitude scheduling” approach, adding some additional variables into the trajectory representation to describe the altitude pattern that should be followed throughout the day. These variables describe the total change in altitude throughout the day, and the proportions of the day that should be spent in each of the five phases show in Figure 4.2: morning rest, ascend, sustain, descend, evening rest. Though this compromise leaves much to be desired in terms of the expressiveness of the representation (more complex altitude patterns are not representable), it does allow the optimizer to find improvements in the trajectories.

## 4.2 Optimization Algorithms

The basic premise of an optimization algorithm is to take some input (in our case a vector of real numbers), evaluate some objective function over that input (in our case the mean throughput), and then modify the input to try and minimize or maximize the objective function. There may also be constraints on the input variables, either directly or indirectly. In our case, for example, a direct constraint on the inputs might be that AoA is limited to the range of zero plus or minus 10 degrees. Indirectly, we

might require that the position of the aircraft remain within some region, which is the result of several input variables. The strategies for modifying the inputs when performing the optimization are numerous (see for example the listings in [34] or [57]), but here we will focus on just two: the Nelder-Mead (NM) method and Particle Swarm Optimization (PSO). Brief descriptions of the variants we utilize are given below, but many other varieties and modifications exist.

#### 4.2.1 Nelder-Mead

The Nelder-Mead method, first proposed in [58], is an optimization algorithm that uses the vertices of a simplex in the search space as candidate solutions, adjusting the worst off vertex in each iteration. A simplex is the interior space (i.e. convex hull) defined by  $n + 1$  point in  $n$  dimensions, such as a line segment in  $\mathbb{R}^1$ , a triangle in  $\mathbb{R}^2$ , a tetrahedron in  $\mathbb{R}^3$ , and so on<sup>5</sup>. Each vertex is then a candidate solution, and keeps track of its corresponding value determined by the objective function. New vertices are generated according to the following rules, where the centroid  $\bar{P}$  is the mean of all the other points:

- Reflection: The worst point mirrored over the centroid as  $P_R = \bar{P} + \alpha(\bar{P} - P_{worst})$
- Expansion: The reflected point expanded away from the centroid as  $P_E = \bar{P} + \gamma(P_R - \bar{P})$
- Contraction: The point contracted towards the centroid as  $P_C = \bar{P} + \rho(P_{worst} - \bar{P})$

This is represented for  $\mathbb{R}^2$  in Figure 4.3a, where the black lines represent the original simplex. The algorithm first checks the reflection point. If that point is better than the best vertex so far, then the expansion point is checked. If the reflection point

---

<sup>5</sup>Note that if three or more vertices are colinear (e.g. forming a line in  $\mathbb{R}^2$ ) then the simplex is said to be degenerate. In the case of Nelder-Mead, this prevents the algorithm from searching perpendicularly to that line and may hinder the search.

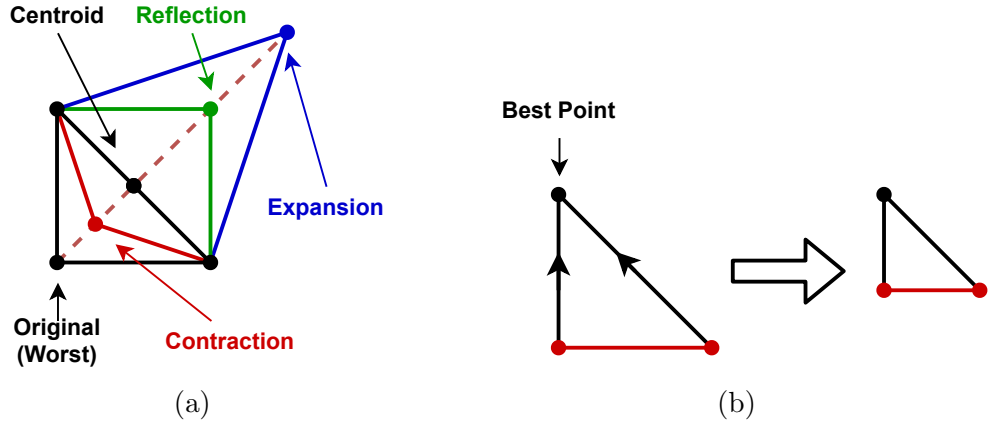


Figure 4.3: Nelder-Mead Operations in  $\mathbb{R}^2$ . (a) Vertex Generation and (b) Shrink Operation.

is not better than the next-worst vertex in the simplex, then the contraction point is checked. The checked point with the best fitness replaces the worst vertex. If none of the checked points is better than the next-worst vertex, all points in the simplex are shrunk towards the centroid as  $P'_i = P_{best} + \sigma(P_i - P_{best})$ , as seen in Figure 4.3b. The coefficients  $\alpha$ ,  $\gamma$ ,  $\rho$ , and  $\sigma$  are respectively the reflection, expansion, contraction, and shrink coefficients. As an example, in our case we take them to be 1, 2, 0.5, and 0.5 respectively<sup>6</sup>.

Notice that this version of NM is inherently serial, and only one vertex may be updated at a time. To allow parallel computation (specifically of the objective function) we use the modifications described in [53] of updating the  $k$  worst vertices, spreading these evaluations over multiple cores. The algorithm remains largely the same, except that the “next-worst vertex” becomes the  $k + 1^{th}$  worst vertex, and the centroid is that of all vertices except the  $k$  worst<sup>7</sup>.

<sup>6</sup>These parameters are the usual default values found in the literature, though we also explored adaptive coefficients as described in [59].

<sup>7</sup>In that paper it is even shown that updating vertices in parallel in this manner may decrease the total number of updates/evaluations required during optimization for some problems. Investigating if this is the case for our problem would be an interesting later investigation.

### 4.2.2 Particle Swarm Optimization

Particle Swarm optimization is a population based optimization algorithm. A number of particles (the “swarm”) which represent candidate solutions are placed into the solution space. Each particles has a position, a velocity, and the coordinates and value of the best point it has visited. The swarm itself also keeps track of the position and value of the best position that any of the particles has visited. In each iteration, a particle updates its velocity, and then updates it’s position according to that velocity multiplied by some learning rate  $\gamma$ . The updated velocity is the sum of the following components:

- Inertia: The current velocity multiplied by some factor  $w$
- Cognitive: The distance to the particle’s personal best, multiplied by some factor  $c_1$
- Social: The distance to the swarms global best, multiplied by some factor  $c_2$

Additionally, the Cognitive and Social vectors are modified stochastically, where each dimensional component is multiplied by a random uniform number in the range  $[0, 1)$ . This is represented in Figure 4.4, without the random modifications to the cognitive and social vectors for simplicity. In our case, we use an inertia factor in the

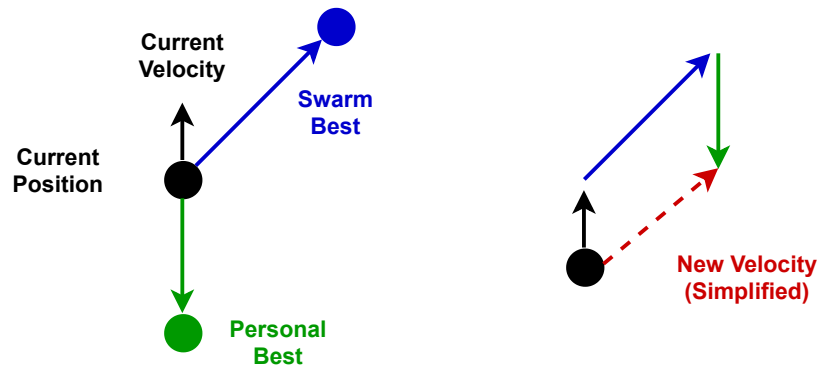


Figure 4.4: Particle Swarm Optimization Velocity Update

approximate range of  $[0.9, 1)$  (varying by attempt, sometimes decreasing over time to encourage convergence), have  $c_1 = c_2 = 2$ , and use  $\gamma = 0.1$ .

## 4.3 Results

Here we'll take a look at how well these algorithms perform on the given problem, both from the perspective of the throughput estimation as well as a closer look at the simulated network performance of the generated trajectories for two particular scenarios. First, we'll discuss the specific constraints used and how they are enforced, as well as the initial conditions for the trajectories.

### 4.3.1 Constraints

As a reminder, both the PSO and NM algorithms operate over a vector of real numbers representing the trajectory. This vector is composed of the altitude schedule (Figure 4.2), where the gain is in meters and the other values represent the proportion of the day during which that segment occurs, as  $(gain, rest, ascend, sustain, descend, rest)$ , followed by the waypoints. Each waypoint contains five values:  $(x, y, heading, \alpha_1, \alpha_2)$ . The trajectory is then built, and statistics on the position, velocity, energy levels, and throughput are generated. These statistics are then fed into our objective function, which is the sum of one "reward" and multiple "constraint penalty" factors. The reward value in our case is the throughput, expressed as the average Mbps/user over the time period. The constraint factors are non-positive values which increase in magnitude (we chose to make this increase quadratic) with respect to the amount the constrain is violated. For example, we have a radius constraint which penalizes the objective value for every meter the aircraft flies outside the radius. This has the effect of strongly discouraging the optimizer from exploring outside our feasible region, while also guiding it to the feasible region if it begins outside. The specific constraint factors and coefficients (1 unless otherwise stated) we utilize are as follows:

- Radius Penalty: Meters outside a 2000 meter radius, coefficient of  $10^{-6}$ .
- Altitude Penalty: Meters below 1,000 meters or above 10,000 meters.
- Energy Penalty: Watt-hours below some “budget” of allowed energy (battery and potential) loss over the time period.
- Thrust Penalty: Newtons above 100 N of thrust.
- Speed Penalty: Meters per second below 6 m/s or above 25 m/s.

The budget used for the energy penalty is varied, to examine its impact on throughput and trajectory. Note that a positive value for the budget indicates energy may be expended over the course of the day, while a negative budget indicates energy must be gained to meet the target. This might allow for more energy to be expended when network resources are in higher demand, and conserved otherwise<sup>8</sup>. A starting point is provided to the optimizer, along with a list of offsets, to construct the initial populations. This point represents a circular trajectory with a radius of 1,800 meters, with four waypoints (separated by  $\frac{\pi}{2}$  radians), and an initial altitude schedule with a gain of 3,000 meters and even periods. The offsets, which are each multiplied by a random variable  $X \sim U(-1, 1)$  and added to the initial point, have values of (400, 0.2, 0.2, 0.2, 0.2, 0.2) for the altitude schedule and (100, 100, 0.1, 1, 1) for each waypoint. Due to the tendency of the PSO algorithm to attain high velocities and potentially leave the feasible region, bounds are also placed on the values. For the full simulations, the optimizer is allowed to run for 1,000 iterations, and for 250 iterations in the budget comparisons<sup>9</sup>.

---

<sup>8</sup>This might also be useful for more granular optimization scheduling, allowing more budget to be allocated to high demand times of day, but this is not explored here.

<sup>9</sup>As there is a diminishing return for more iterations, a more intelligent termination condition would be desirable in the future.

### 4.3.2 Optimizer Performance

In order to determine whether our throughput estimation is reasonable, and to see how the optimizers perform with an excess of iterations allowed, we'll first look at the 1,000 iteration runs. Both algorithms were run with energy budgets of 0 Wh and -500 Wh (the aircraft must end up with 500 *more* Wh than it started with), and allowed a maximum of either 1,000 iterations or  $2^{17} \approx 128,000$  function evaluations. The performance over time, plotted in Figure 4.5, displays a few interesting characteristics. Firstly, take note that in the low iterations the objective function is outside the frame, indicating that the trajectory is infeasible. Second, we can see that in both cases the PSO algorithm takes an early lead, but then appears to be stuck in a local optimum. Some iterations later, it is overtaken by Nelder-Mead. Finally, we can see a similar feature on the Nelder-Mead curve for both budgets, though at different iteration numbers, where it reaches a plateau and then proceeds further. This is possibly due to the identical initial conditions of the optimizer (the starting simplex), suggesting a similar route is taken under both energy budget constraints.

Taking the best trajectory for each optimizer/budget combination, we can examine the differences between our estimated throughput and the value determined through simulation in Figure 4.5. Notably, the simulated throughput ends up being between 73% and 78% of the estimated value, dropping to the lower end when the craft is further away from the ground. This confirms that the throughput estimation is reasonably proportional to what we might expect in reality over the region of interest.

Next, we can look at the performance of the optimizers over multiple runs with different initial points and random number generator seeds. In Figure 4.7 we display the mean objective function values over 5 runs of each optimizer (with identical random-number-generator streams between optimizers/budgets), with the shaded regions indicating the minimum/maximum value range for each iteration. We can see that with decreasing energy budgets the objective value achieved in 250 iterations



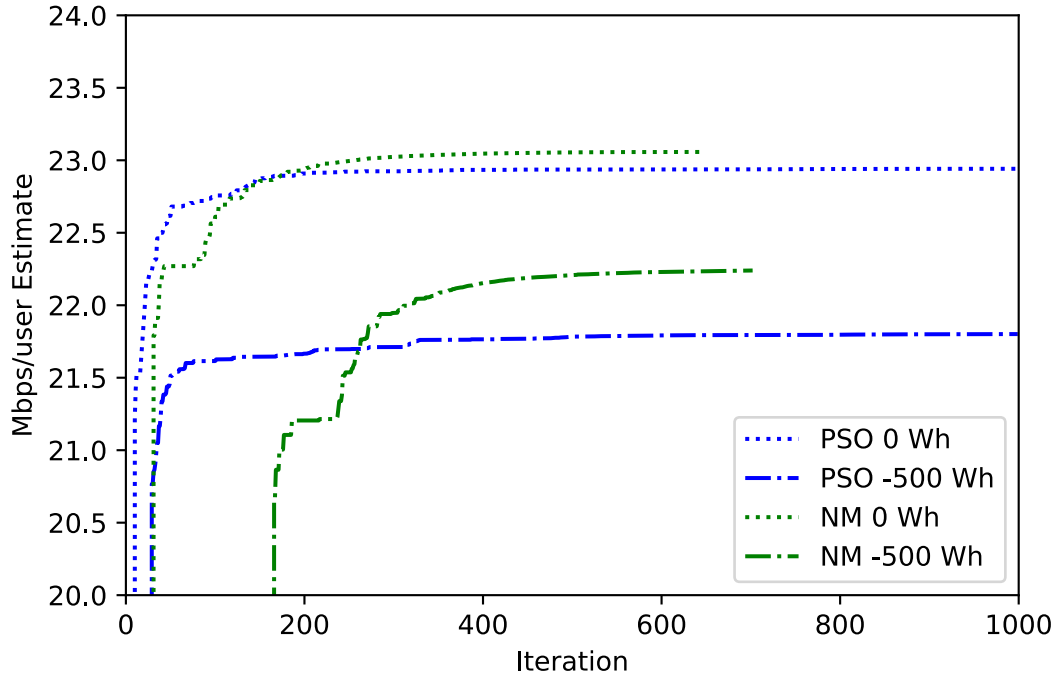


Figure 4.5: Optimizer Performance for 0Wh and -500Wh Budgets

decreases as well. Between the two algorithms, we observe that PSO reaches the feasible region much faster than NM, but remains at a local optimum below what NM reaches after more iterations. (This is the case for each budget except for -750 Wh, but we'll conjecture the same pattern would play out over more iterations). We can also see that lower energy budgets increases the number of iterations needed to find feasible solutions and optimize the results. It is quite possible that this is due to an

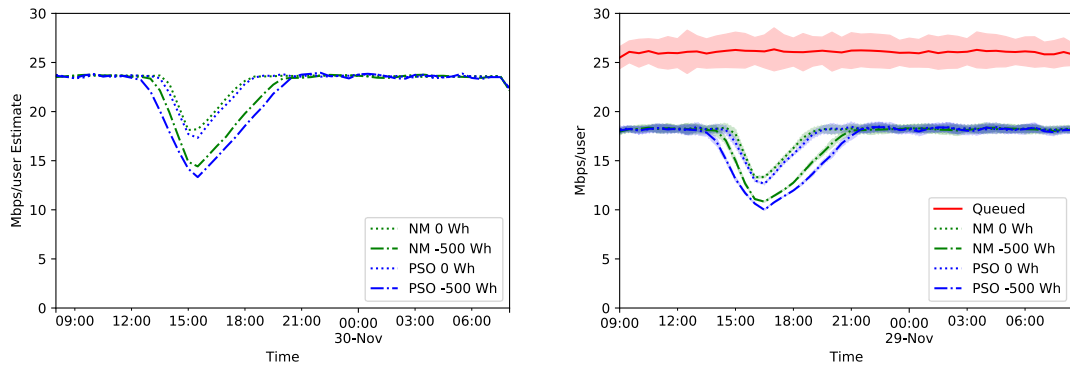


Figure 4.6: Comparison of Estimated to Simulated User Throughput

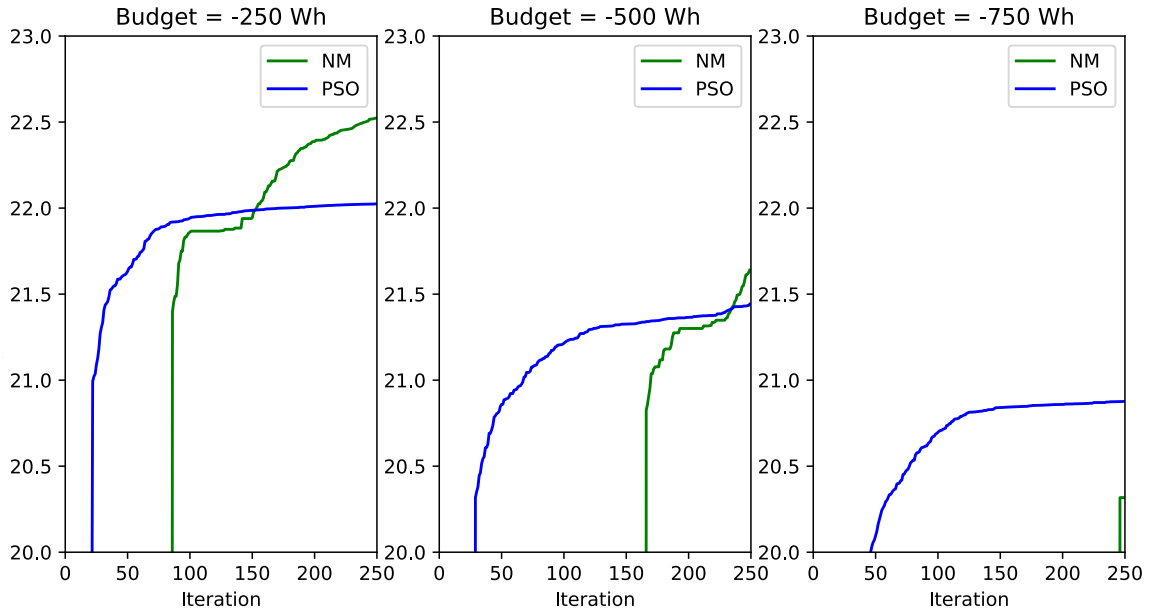


Figure 4.7: Optimizer Performance for Various Budgets. Minimum-Maximum Range Shaded.

increased distance from our initial guess to the feasible space, though other factors may also be at play.

One last item of note is that the horizontal profile of the trajectory did not differ much from the starting circular trajectory, and most of the optimization appears to have occurred in the altitude schedule. More modifications did occur in the PSO optimizer, but these appear to have been detrimental to the overall process. This can be observed by replacing all components of the vector unrelated to the altitude schedule with the template trajectory, returning the trajectory to a simple circle but ascending and descending according to the determined schedule. In doing this, we see a negligible change in throughput, a small change in energy balance for the NM optimized trajectories, and a much larger increase in the PSO optimized trajectories, see Table 4.1. This suggests that our optimizers were unable to find any improvements to be made in the XY plane given our representation. In the case of PSO, the most-likely case is that the algorithm approaches the feasible space correctly, but

Energy Budget	Optimizer	Optimized Trajectory	Altitude-Schedule Only
-250 Wh	PSO	251.44 Wh	370.41 Wh
-500 Wh	PSO	505.02 Wh	627.71 Wh
-750 Wh	PSO	755.46 Wh	887.23 Wh
-250 Wh	NM	262.83 Wh	267.97 Wh
-500 Wh	NM	544.69 Wh	568.11 Wh
-750 Wh	NM	757.43 Wh	773.93 Wh

Table 4.1: 24-hour Energy Gain: Optimized vs Altitude-Schedule Only

remains stuck in a local optimum after that point. For comparison, the optimizations performed in [11] differed from circular trajectories such that it becomes an ellipsoid with the major axis roughly aligned to the sun.

## CHAPTER V

### Conclusions and Future Work

In this work we develop a framework for evaluating trajectories for perpetual-endurance, solar-powered fixed-wing aircraft tasked with providing network coverage, as well as multiple methods for optimizing these trajectories to maximize the provided throughput under energy constraints. We demonstrate our measurement framework on a small number of baseline trajectories, and then perform optimizations for throughput subject to various energy constraints to compare the performance of the two optimization algorithms. The results demonstrate that the algorithms are capable of modifying the base trajectory to meet the energy feasibility constraints, and to increase the network performance beyond that. We also observe that given our starting conditions, more stringent energy constraints lead to slower convergence and lower estimated throughput.

Several avenues exist for further investigation. Firstly, given our representation, much of the information that might be gained in one “loop” of the trajectory that might be useful in the next has no impact on it. The optimizer has no way to re-use this information. Alternative representations might better exploit any such patterns. Secondly, the use of a singular altitude scheduler limits the expressiveness of the representation. It might have been advantageous to, for example, ascend somewhat while heading away from the sun, and descend while approaching it, while maintaining

an overall net increase in altitude. A potential addition to the representation would be some manner of offset for each waypoint, to facilitate local deviations from the altitude schedule while maintaining the overall pattern.

Aside from changes to the representation, there are many, many other methods for performing this manner of optimization. [34] provides a good overview of many of these. In addition to this, one might also perform a multi-objective search, allowing operators some freedom to choose between expending more energy when more throughput is required, or conserving more energy when network demand is not as high.

At a lower level, the code used for evaluating these trajectories is moderately inefficient. A single trajectory evaluation, including constructing the segments, determining the poses, and gathering energy and throughput information, took around 4-5 seconds in the final version. As the evaluation must be performed many thousands of times during the process of optimization, this poses a large bottleneck to iterating on approaches and gathering statistics.

## APPENDICES

## APPENDIX A

### Aerodynamics

Here we will derive the equations necessary to determine the power needed by an aircraft to maintain steady-state flight in a simplified environment. We also present some equations to determine trajectory characteristics given control inputs.

#### A.1 Context

The following is based largely on the work in [14], where the authors work a similar problem, sans the addition of solar energy or an altitude component.

A fixed wing aircraft has the following primary forces acting upon it while in motion (see Figure A.1):

- Weight/gravity towards the Earth
- Thrust from the propeller
- Lift, modelled as  $90^\circ$  above the velocity vector
- Drag, modelled in the direction opposite the velocity vector

Thus, for an aircraft travelling in a straight and level trajectory, weight will be down, thrust will be forward, lift will be up, and drag will be to the rear. In our

case, we consider that the aircraft can have a velocity above or below horizontal, and that the aircraft can have an attitude above or below its velocity vector. In this case, lift is still modelled as above and perpendicular to the velocity vector, drag parallel to the velocity vector, and thrust potentially above/below the velocity vector, as in Figure A.1a. The angle of the velocity above horizontal will be referred to as  $\theta$ , and the angle of the aircraft (and incidentally thrust) above  $\theta$  to be the Angle-of-Attack  $\alpha$ . Finally, to execute a turn, the aircraft rolls at angle  $\phi$  such that the wing in the direction of the center moves down, as seen in Figure A.1b.

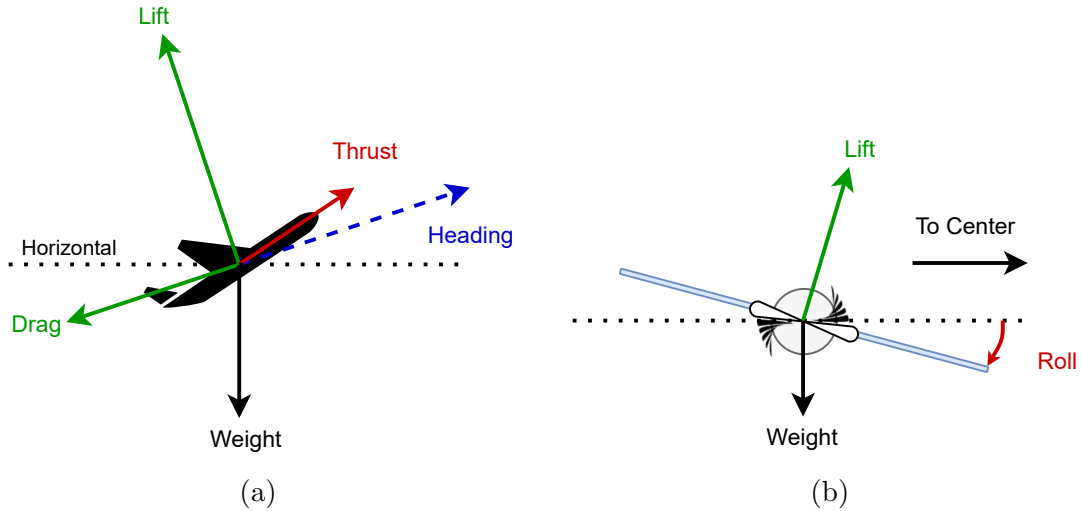


Figure A.1: (a) Forces acting on the aircraft. (b) When turning, the aircraft rolls and the angled lift contributes to the net centripetal force.

The magnitudes of these forces are given below in Equation A.1. Note that  $C_L$  and  $C_{D,wing}$  can be determined with XFOIL [60] and are functions of  $\alpha$  and the Reynolds Number. A collection of coefficients for different airfoils is available at [61]. The air density  $\rho$  is modelled as in [62]. Equation A.1f and Equation A.1g are defined for use in the intermediate equations.



$$F_L = 1/2 \cdot \rho \cdot C_L \cdot S \cdot v^2 = L_1 \cdot v^2 \quad (\text{A.1a})$$

$$F_D = 1/2 \cdot \rho \cdot C_D \cdot S \cdot v^2 = D_1 \cdot v^2 \quad (\text{A.1b})$$

$$C_D = C_{D,wing} + C_{D,par} + C_{D,ind} \quad (\text{A.1c})$$

$$C_{D,par} = 0.074 \cdot Re^{-0.2} \quad (\text{A.1d})$$

$$C_{D,ind} = \frac{C_L^2}{\pi \cdot e_0 \cdot A_R} \quad (\text{A.1e})$$

$$L_1 = 1/2 \cdot \rho \cdot C_L \cdot S \quad (\text{A.1f})$$

$$D_1 = 1/2 \cdot \rho \cdot C_D \cdot S \quad (\text{A.1g})$$

We have the following variables used in the above:

- $m$ : aircraft mass
- $g$ : local gravitational acceleration (e.g.  $9.8m/s^2$ ). Varies slightly with altitude
- $C_L$  and  $C_D$ : Coefficients of lift and drag, a function of the airfoil used and Angle of Attack  $\alpha$
- $v$ : aircraft velocity relative to its medium. Note that we assume that there is no sideslip for our calculations.
- $S$ : Wing surface area
- $\rho$ : Air density. Decreases with altitude [62].
- $\pi$ : The ratio of a circle's area to it's radius squared
- $e_0$ : Oswald Efficiency, a function of the shape of the wings.
- $A_R$ : Aspect ratio of the wings:  $\frac{wingspan}{wingchord}$

Given the situation of the aircraft, the resulting thrust and velocity must be determined for a steady state condition. These values are derived in the following sections. Once these are known, the power needed by the aircraft propulsion system is then  $P = |T| \cdot v$  as in [14]. It should be noted that this only works for approximately level flight, and for extreme inputs (e.g. the aircraft oriented vertically and hovering) the results may be invalid (e.g. the hovering aircraft has zero velocity, implying zero power use). Constraining AoA and altitude change rate appropriately should prevent this scenario from occurring.

## A.2 Circular Arc Segment Equations

Starting with the above, we can derive the equations for the velocity, thrust, and power required for an aircraft to traverse a path defined by a circular arc (as projected on the XY plane) ascending at some angle  $\theta$ . We also consider the case of a straight path as  $r = \infty$ .

We begin with the following net force:

$$\vec{F} = \vec{F}_T + \vec{F}_L + \vec{F}_D + \vec{F}_g \quad (\text{A.2})$$

For the following we use a forward-left-down (FLD) reference frame, where the  $x+$  direction is forward,  $y+$  is left, and  $z+$  is down. Relative to the rest orientation, we arrive at the attitude of the aircraft by pitching (about the  $y$  axis), rolling (about the  $x$  axis), and yawing (about the  $z$  axis), in that order. The respective rotation matrices are then:

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \quad (\text{A.3})$$

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{pmatrix} \quad (\text{A.4})$$

$$R_z(\psi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (\text{A.5})$$

Applied in reverse order, our rotation matrix is thus

$$R(\theta, \phi, \psi) = R_z(\psi) \cdot R_x(\phi) \cdot R_y(\theta) \quad (\text{A.6})$$

By applying the appropriate pitch and roll rotations to our thrust, lift, and drag vectors, we arrive at the magnitudes of these forces along each dimension. Note that for the thrust vector, we consider the AoA in addition to the angle of ascent/pitch, as the aircraft (and thus the propellers) are oriented above the velocity vector (for a positive AoA). We arrive at

$$\vec{F}_T = \begin{pmatrix} T \\ 0 \\ 0 \end{pmatrix} \cdot R(\theta + \alpha, \phi, 0) = T \cdot \begin{pmatrix} \cos(\theta + \alpha) \\ \sin(\theta + \alpha) \cdot \sin(\phi) \\ -\sin(\theta + \alpha) \cdot \cos(\phi) \end{pmatrix} \quad (\text{A.7a})$$

$$\vec{F}_L = \begin{pmatrix} 0 \\ 0 \\ -F_L \end{pmatrix} \cdot R(\theta, \phi, 0) = F_L \cdot \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \cdot \sin(\phi) \\ -\cos(\theta) \cdot \cos(\phi) \end{pmatrix} \quad (\text{A.7b})$$

$$\vec{F}_D = \begin{pmatrix} -F_D \\ 0 \\ 0 \end{pmatrix} \cdot R(\theta, \phi, 0) = F_D \cdot \begin{pmatrix} -\cos(\theta) \\ -\sin(\theta) \cdot \sin(\phi) \\ \sin(\theta) \cdot \cos(\phi) \end{pmatrix} \quad (\text{A.7c})$$

$$\vec{F}_g = \begin{pmatrix} 0 \\ 0 \\ m \cdot g \end{pmatrix} \quad (\text{A.7d})$$

### A.2.1 Circular Arcs

In the circular arc case, our net force must be equal to the centripetal force needed to maintain the turn. In the case of a straight path, our net force must be zero. We begin with the case for a non-infinite radius, breaking out the above equations along each dimension:

$$0 = T \cos(\theta + \alpha) - L \sin(\theta) - D \cos(\theta) \quad (\text{A.8a})$$

$$\frac{mv^2}{r} = T \sin(\theta + \alpha) \sin(\phi) + L \cos(\theta) \sin(\phi) - D \sin(\theta) \sin(\phi) \quad (\text{A.8b})$$

$$0 = -T \sin(\theta + \alpha) \cos(\phi) - L \cos(\theta) \cos(\phi) + D \sin(\theta) \cos(\phi) + mg \quad (\text{A.8c})$$

We then use Equation A.8b and Equation A.8c to reduce  $\phi$  to a function of  $v^2$ . Factoring out  $\sin \phi$  and  $\cos \phi$  we have:

$$\frac{mv^2}{r} = \sin(\phi)[T \sin(\theta + \alpha) + L \cos(\theta) - D \sin(\theta)] \quad (\text{A.9a})$$

$$mg = \cos(\phi)[T \sin(\theta + \alpha) + L \cos(\theta) - D \sin(\theta)] \quad (\text{A.9b})$$

Dividing the functions of  $\phi$  from the right-hand-side to the left-hand-side, we can

then set the two equations equal to each other and cancel like terms:

$$\frac{mv^2}{r \sin(\phi)} = [T \sin(\theta + \alpha) + L \cos(\theta) - D \sin(\theta)] = \frac{mg}{\cos \phi} \quad (\text{A.10a})$$

$$\frac{mv^2}{r \sin(\phi)} = \frac{mg}{\cos(\phi)} \quad (\text{A.10b})$$

$$\frac{v^2}{r \sin(\phi)} = \frac{g}{\cos(\phi)} \quad (\text{A.10c})$$

Next, we will solve for  $v^2$  as a function of  $\phi$ , starting from Equation A.8a and Equation A.8c. Substituting in Equation A.1a and Equation A.1b and rearranging for T, we have

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (\text{A.11a})$$

$$T = \frac{-L_1 v^2 \cos(\theta) \cos(\phi) + D_1 v^2 \sin(\theta) \cos(\phi) + mg}{\sin(\theta + \alpha) \cos(\phi)} \quad (\text{A.11b})$$

$$\frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} = \frac{-L_1 v^2 \cos(\theta) \cos(\phi) + D_1 v^2 \sin(\theta) \cos(\phi) + mg}{\sin(\theta + \alpha) \cos(\phi)} \quad (\text{A.11c})$$

Multiplying the denominators across and collecting the  $v^2 \cos(\phi)$  terms we have

$$\begin{aligned} & L_1 v^2 \sin(\theta) \sin(\theta + \alpha) \cos(\phi) && mg \cos(\theta + \alpha) \\ & + D_1 v^2 \cos(\theta) \sin(\theta + \alpha) \cos(\phi) && = +D_1 v^2 \sin(\theta) \cos(\phi) \cos(\theta + \alpha) \\ & && -L_1 v^2 \cos(\theta) \cos(\phi) \cos(\theta + \alpha) \end{aligned} \quad (\text{A.12a})$$

$$v^2 \cos(\phi) \left( \begin{array}{l} L_1 \sin(\theta) \sin(\theta + \alpha) + D_1 \cos(\theta) \sin(\theta + \alpha) \\ + L_1 \cos(\theta) \cos(\theta + \alpha) - D_1 \sin(\theta) \cos(\theta + \alpha) \end{array} \right) = mg \cos(\theta + \alpha) \quad (\text{A.12b})$$

We then divide across  $\cos(\theta + \alpha)$  and solve for  $v^2$

$$v^2 \cos(\phi) (L_1 \sin(\theta) \tan(\theta + \alpha) + D_1 \cos(\theta) \tan(\theta + \alpha) + L_1 \cos(\theta) - D_1 \sin(\theta)) = mg \quad (\text{A.13a})$$

$$v^2 = \frac{mg}{\cos(\phi) [\tan(\theta + \alpha)(L_1 \sin(\theta) + D_1 \cos(\theta)) + (L_1 \cos(\theta) - D_1 \sin(\theta))]} \quad (\text{A.13b})$$

Combining Equation A.10c and Equation A.13b we can solve for  $\phi$ :

$$\frac{v^2}{r \sin(\phi)} = \frac{g}{\cos(\phi)} \rightarrow v^2 = \frac{rg \sin(\phi)}{\cos(\phi)} \quad (\text{A.14})$$

$$\frac{rg \sin(\phi)}{\cos(\phi)} = \frac{mg}{\cos(\phi) [\tan(\theta + \alpha)(L_1 \sin(\theta) + D_1 \cos(\theta)) + (L_1 \cos(\theta) - D_1 \sin(\theta))]} \quad (\text{A.15})$$

Cancelling the  $g$  and  $\cos(\phi)^{-1}$  terms, and dividing by  $r$ , we thus have

$$\sin(\phi) = \frac{m}{r [\tan(\theta + \alpha)(L_1 \sin(\theta) + D_1 \cos(\theta)) + (L_1 \cos(\theta) - D_1 \sin(\theta))]} \quad (\text{A.16})$$

We can then use Equation A.13b again to find  $v^2$ . Finally, we can use our velocity

to find the needed thrust with Equation A.11a. Our final values are thus

$$\sin(\phi) = \frac{m}{r [\tan(\theta + \alpha)(L_1 \sin(\theta) + D_1 \cos(\theta)) + (L_1 \cos(\theta) - D_1 \sin(\theta))]} \quad (\text{A.17a})$$

$$v^2 = \frac{rg \sin(\phi)}{\cos(\phi)} \quad (\text{A.17b})$$

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (\text{A.17c})$$

### A.2.2 Lines

To solve for the linear segments, we take the Y/Left component to be 0, and will thus have a roll  $\phi = 0$  as no lateral force is needed. Excluding the appropriate terms from Equation A.8, we have the following two equations to solve for  $v$  and  $T$ :

$$0 = T \cos(\theta + \alpha) - L_1 v^2 \sin(\theta) - D_1 v^2 \cos(\theta) \quad (\text{A.18a})$$

$$0 = -T \sin(\theta + \alpha) - L_1 v^2 \cos(\theta) + D_1 v^2 \sin(\theta) + mg \quad (\text{A.18b})$$

Rearranging for  $T$  we get

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (\text{A.19a})$$

$$T = \frac{-L_1 v^2 \cos(\theta) + D_1 v^2 \sin(\theta) + mg}{\sin(\theta + \alpha)} \quad (\text{A.19b})$$

We then set these two equations equal to each other, multiply across, and partially distribute the denominators

$$\frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} = \frac{-L_1 v^2 \cos(\theta) + D_1 v^2 \sin(\theta) + mg}{\sin(\theta + \alpha)} \quad (\text{A.20a})$$

$$[L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)] \sin(\theta + \alpha) = [-L_1 v^2 \cos(\theta) + D_1 v^2 \sin(\theta) + mg] \cos(\theta + \alpha) \quad (\text{A.20b})$$

$$[L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)] \sin(\theta + \alpha) = [-L_1 v^2 \cos(\theta) + D_1 v^2 \sin(\theta)] \cos(\theta + \alpha) + mg \cos(\theta + \alpha) \quad (\text{A.20c})$$

We can then collect the  $mg$  term and divide by  $\cos(\theta + \alpha)$

$$mg \cos(\theta + \alpha) = [L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)] \sin(\theta + \alpha) + [L_1 v^2 \cos(\theta) - D_1 v^2 \sin(\theta)] \cos(\theta + \alpha) \quad (\text{A.21})$$

$$mg = \frac{[L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)] \sin(\theta + \alpha) + [L_1 v^2 \cos(\theta) - D_1 v^2 \sin(\theta)] \cos(\theta + \alpha)}{\cos(\theta + \alpha)} \quad (\text{A.22})$$

$$mg = [L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)] \tan(\theta + \alpha) + [L_1 v^2 \cos(\theta) - D_1 v^2 \sin(\theta)] \quad (\text{A.23})$$

Finally, we collect our  $v^2$  term

$$mg = v^2 [L_1 \sin(\theta) + D_1 \cos(\theta)] \tan(\theta + \alpha) + [L_1 \cos(\theta) - D_1 \sin(\theta)] \quad (\text{A.24})$$

$$v^2 = \frac{mg}{[L_1 \sin(\theta) + D_1 \cos(\theta)] \tan(\theta + \alpha) + [L_1 \cos(\theta) - D_1 \sin(\theta)]} \quad (\text{A.25})$$

As in the circular case we can use Equation A.11a to find our thrust. Our final equations for a linear segment are then

$$v^2 = \frac{mg}{[L_1 \sin(\theta) + D_1 \cos(\theta)] \tan(\theta + \alpha) + [L_1 \cos(\theta) - D_1 \sin(\theta)]} \quad (\text{A.26})$$

$$T = \frac{L_1 v^2 \sin(\theta) + D_1 v^2 \cos(\theta)}{\cos(\theta + \alpha)} \quad (\text{A.27})$$



## APPENDIX B

### Code

At present, the code can be accessed at <https://github.com/jarmillemich/umf-thesis>. The bulk of the evaluation and optimization code is under `/jarmillemich-ns3/python/thesis`, with several notebooks in the directory above. Also in the directory above are the files `runner.py`, `runner2.py`, and `optimizerunner.py`, which can be used to run the NS3 simulations for the baseline trajectories, for the optimized trajectories, and to run the optimization algorithms, respectively. The code can be run in the SageMath [63] environment (tested in 9.0), which is capable of hosting the (Jupyter/IPython [64]) notebooks.

#### B.1 Modules

##### B.1.1 Aircraft.py

The Aircraft module contains the Aircraft class and some supporting code (most notably the altitude model from [62]). The Aircraft class represents the various physical characteristics of the fixed wing model, as well as code related to calculating the produced solar energy. Aircraft mass, wing configuration (span, chord, airfoil, solar cell fill ratio) and various component efficiencies (solar modules, propeller, oth-

ers) may be configured in the constructor. Methods are available to determine the velocity, thrust, and power of different classes of trajectory segment, as well as to calculate the bank angle from the turn radius (and vice versa) in any particular set of conditions.

#### **B.1.1.1 ThesisCraft.py**

This module contains the specifications for the reference Aircraft instance used in our analysis.

#### **B.1.1.2 Trajectory.py**

The Trajectory class represents a collection of trajectory segments that make up a particular aircraft trajectory. These are also present in this module as the LineSegment, ArcSegment, and GeneralSegment classes. Each Segment class contains methods for:

- velocityThrustPower: calculating the relevant velocity, thrust, and power values for a particular Aircraft instance
- toPoses: calculating the pose (position and attitude, as well as velocity, thrust, and power) of the aircraft at various time points
- render, renderTop, renderSide: Helper methods used to visualize the trajectory

The (base) Trajectory class itself contains the same named methods for rendering all of the segments of the trajectory, as well as a method to compute the total length of all of the segments in the trajectory. A number of other trajectory representations are currently co-located in this module. Most notably, CircleTrajectory, BowtieTrajectory, and SimpleLadderTrajectory represent our three baseline trajectories demonstrated in section 3.3.

### **B.1.1.3 Flight.py**

The Flight class represents a combination of an Aircraft and a Trajectory instance, as well as the Angles of Attack for each of the segments of the Trajectory. Incidentally, it also contains the configuration for the aircraft radio. Notable methods include:

- toPoses: Invokes toPoses on all the trajectory segments, and stitches together the resulting series of data.
- toSim: Creates a PathMobilityModel for use in NS3 simulations

### **B.1.1.4 Scenario.py**

The Scenario class represents a collection of user locations, which can be used to estimate the available throughput with the posesToThroughput method. There are also helper methods from creating randomly located populations of users.

### **B.1.1.5 EvalHelper.py**

The EvalHelper.py module contains the “Judge” class, which contains several functions for evaluating trajectories<sup>1</sup>. Methods to generate the various energy and altitude vs. throughput plots such as Figure 3.4 are also available here.

### **B.1.1.6 ./optimize**

The optimize directory contains the code related to running the optimization routines. BaseOptimizer.py contains the base class for the other optimizers, including code related to distributing fitness evaluations among multiple threads. The class used for representing our solution vectors, with methods to manipulate them, is also located here. The functions.py module contains several functions that can be used to build expressions over statistics gathered in the EvalHelper module. For example, one

---

<sup>1</sup>There is also some legacy code related to early experiments with genetic algorithms.

can create a fitness expression as “throughputReward() + energyPenalty(-500) \* 0.5” to create an expression that will reward a flight for providing throughput, and penalize the flight for not meeting an energy budget of -500 Wh, with a coefficient of 0.5. These expressions can be used with the FitnessHelper and SplineyFitnessHelper classes in the same module, which are responsible for constructing and evaluating the trajectories. The remainder of the modules in this directory are the implementations of the various optimization algorithms. PSOv2.py and ParallelNelderMead.py contain the code used for this work, PSO.py contains an earlier implementation of the PSO algorithm, and GA.py and Genetics.py contain code for running a genetic algorithm (not currently integrated with the remainder of the code).

## B.2 Running the Optimizer

The first item needed to run the optimization is to define the fitness function. In our case, this function takes the vector, constructs the trajectory representation, computes statistics, and then produces a value based on those statistics. We have defined the “SplineyFitnessHelper”<sup>2</sup> class to build this function. For example, we may construct this helper as

```

helper = SplineyFitnessHelper(
    judge, craft, times,
    expr = [
        # === Optimize this ===
        # Throughput, in Mbps/user
        throughputReward() / 1e6 / len(scene.users),

        # === Subject to these constraints (penalty functions) ===
        # Flight volume constraints

```

---

<sup>2</sup>“Spliney” referring to our formulation, which is similar to but distinct from splines.

```

    radiusPenalty(2000) * 1e-6,
    altitudePenalty(1000, 10000),
    # Energy budget constraint
    energyPenalty(args.energy, gravityCoeff = 1.0),
    # Some aircraft/modelling constraints
    thrustPenalty(hi = 100),
    speedPenalty(lo = 6, hi = 25)
],
# Scale trajectory to evenly fit into a 24-hour window
desiredDuration = 24*3600,
# Use our altitude-scheduling model
zMode = 'schedule'
)

```

The other important piece of configuration to note is how the initial population (particles or vertices, in our case) are generated. This process is implemented as described in subsection 4.3.1. The remaining pieces of configuration such as population size, number of threads to use for parallel fitness evaluation, and parameters specific to each optimizer can also be configured in the optimizer constructor. The `BaseOptimizer` class provides an `iterateMany` method which can then be used to run an appropriate number of iterations. This method also takes a callback function `cb` that is evaluated after every iteration and can be used for reporting. A simplified example of this can be seen below. See the script `optimizerrunner.py`, which was used to generate the optimization results in subsection 4.3.2, for a full example.

```

optimizer = PSO(
    numParticles,
    numCodons,
    createParticle,

```

```
    fitness ,
    processes=args.threads ,
    wSchedule = wRamp,
    bounds=bounds
)

optimizer.iterateMany(iterations = 250, cb=onIteration)
```

## BIBLIOGRAPHY

## REFERENCES

- [1] *Commercial UAVs worldwide*. Tech. rep. did-28867-1. Statista Research, Dec. 2019. URL: <https://www.statista.com/study/28867/commercial-uavs/>.
- [2] Y. Zeng, R. Zhang, and T. J. Lim. “Wireless communications with unmanned aerial vehicles: opportunities and challenges”. In: *IEEE Communications Magazine* 54.5 (2016), pp. 36–42. DOI: 10.1109/MCOM.2016.7470933.
- [3] L. Gupta, R. Jain, and G. Vaszkun. “Survey of Important Issues in UAV Communication Networks”. In: *IEEE Communications Surveys Tutorials* 18.2 (2016), pp. 1123–1152. DOI: 10.1109/COMST.2015.2495297.
- [4] N. Hossein Motlagh, T. Taleb, and O. Arouk. “Low-Altitude Unmanned Aerial Vehicles-Based Internet of Things Services: Comprehensive Survey and Future Perspectives”. In: *IEEE IoT Journal* 3 (2016). DOI: 10.1109/JIOT.2016.2612119.
- [5] H. Shakhathreh et al. “Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges”. In: *IEEE Access* 7 (2019), pp. 48572–48634. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2909530.
- [6] Y. Zeng, Q. Wu, and R. Zhang. “Accessing From the Sky: A Tutorial on UAV Communications for 5G and Beyond”. In: *Proceedings of the IEEE* 107.12 (2019), pp. 2327–2375. DOI: 10.1109/JPROC.2019.2952892.
- [7] M. Mozaffari et al. “A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems”. In: *IEEE Comm. Surv. Tutorials* 21.3 (2019).
- [8] *Helios Prototype Flying Wing*. Sept. 2009. URL: <https://www.nasa.gov/centers/dryden/multimedia/imagegallery/Helios/ED01-0209-6.html>.
- [9] AlanLiu. *UAV Industrial Design*. Feb. 2016. URL: <https://pixabay.com/photos/uav-industrial-design-design-robot-1204472/>.
- [10] Philipp et al Oettershagen. “Design of small hand-launched solar-powered UAVs: From concept study to a multi-day world endurance record flight”. In: *Journal of Field Robotics* 34.7 (2017), pp. 1352–1377. DOI: 10.1002/rob.21717. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21717>.
- [11] H. Bolandhemmat, B. Thomsen, and J. Marriott. “Energy-Optimized Trajectory Planning for High Altitude Long Endurance (HALE) Aircraft”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 1486–1493. DOI: 10.23919/ECC.2019.8796240.



- [12] J. Marriott et al. “Trajectory Optimization of Solar-Powered High-Altitude Long Endurance Aircraft”. In: *2020 6th Int’l Conf. on Control, Automation and Robotics*. 2020. DOI: 10.1109/ICCAR49639.2020.9107998.
- [13] Sean Montgomery. “DESIGN OF A 5 KILOGRAM SOLAR POWERED UNMANNED AIRPLANE FOR PERPETUAL SOLAR ENDURANCE FLIGHT”. MA thesis. San José State University, May 2013.
- [14] Y. Zeng and R. Zhang. “Energy-Efficient UAV Communication With Trajectory Optimization”. In: *IEEE Trans. on Wireless Comm.* 16.6 (2017).
- [15] Chen Qiu et al. “Backhaul-aware trajectory optimization of fixed-wing UAV-mounted base station for continuous available wireless service”. In: *IEEE Access* 8 (2020).
- [16] G. A. Akpakwu et al. “A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges”. In: *IEEE Access* 6 (2018), pp. 3619–3647.
- [17] X. Cao et al. “Airborne Communication Networks: A Survey”. In: *IEEE Journal on Selected Areas in Communications* 36.9 (2018), pp. 1907–1926.
- [18] Y. Zeng, J. Xu, and R. Zhang. “Energy Minimization for Wireless Communication With Rotary-Wing UAV”. In: *IEEE Trans. on Wireless Comm.* 18.4 (2019).
- [19] E. Bozkaya et al. “AirNet: Energy-Aware Deployment and Scheduling of Aerial Networks”. In: *IEEE Transactions on Vehicular Technology* 69.10 (2020), pp. 12252–12263. DOI: 10.1109/TVT.2020.3019918.
- [20] L. Amorosi, L. Chiaraviglio, and J. Galán-Jiménez. “Optimal Energy Management of UAV-Based Cellular Networks Powered by Solar Panels and Batteries: Formulation and Solutions”. In: *IEEE Access* 7 (2019), pp. 53698–53717.
- [21] L. Chiaraviglio et al. “Multi-Area Throughput and Energy Optimization of UAV-aided Cellular Networks Powered by Solar Panels and Grid”. In: *IEEE Transactions on Mobile Computing* (2020), pp. 1–1.
- [22] Y. Sun et al. “Optimal 3D-Trajectory Design and Resource Allocation for Solar-Powered UAV Communication Systems”. In: *IEEE Transactions on Communications* 67.6 (2019), pp. 4281–4298. DOI: 10.1109/TCOMM.2019.2900630.
- [23] Silvia Sekander, Hina Tabassum, and Ekram Hossain. *On the Performance of Renewable Energy-Powered UAV-Assisted Wireless Communications*. 2019. arXiv: 1907.07158 [cs.NI].

- [24] Y. Huang et al. “Endurance Estimate for Solar-Powered Unmanned Aerial Vehicles”. In: *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. Vol. 1. 2017, pp. 66–70. DOI: 10.1109/IHMSC.2017.22.
- [25] Stefan Leutenegger, Mathieu Jabas, and Roland Y. Siegwart. “Solar Airplane Conceptual Design and Performance Estimation”. In: *Journal of Intelligent & Robotic Systems* 61.1 (Jan. 2011), pp. 545–561. ISSN: 1573-0409. DOI: 10.1007/s10846-010-9484-x. URL: <https://doi.org/10.1007/s10846-010-9484-x>.
- [26] J. Meyer et al. “Design considerations for a low altitude long endurance solar powered unmanned aerial vehicle”. In: *AFRICON 2007*. 2007, pp. 1–7.
- [27] P. Oettershagen et al. “A solar-powered hand-launchable UAV for low-altitude multi-day continuous flight”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3986–3993. DOI: 10.1109/ICRA.2015.7139756.
- [28] P. Oettershagen et al. “Perpetual flight with a small solar-powered UAV: Flight results, performance analysis and model validation”. In: *2016 IEEE Aerospace Conference*. 2016, pp. 1–8.
- [29] Philipp Oettershagen. *High-Fidelity Solar Power Income Modeling for Solar-Electric UAVs: Development and Flight Test Based Verification*. 2017. arXiv: 1703.07385 [cs.R0]. URL: <https://arxiv.org/abs/1703.07385>.
- [30] O. Anicho et al. “Autonomous Unmanned Solar Powered HAPS: Impact of Latitudes and Seasons on Power and Communications Coverage”. In: *2018 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*. 2018, pp. 7–12. DOI: 10.1109/COMNETSAT.2018.8684145.
- [31] O. Anicho et al. “Integrating Routing Schemes and Platform Autonomy Algorithms for UAV Ad-hoc Infrastructure based networks”. In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. 2018, pp. 1–5. DOI: 10.1109/ATNAC.2018.8615237.
- [32] O. Anicho et al. “Autonomously Coordinated Multi-HAPS Communications Network: Failure Mitigation in Volcanic Incidence Area Coverage”. In: *2019 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*. 2019, pp. 79–84. DOI: 10.1109/COMNETSAT.2019.8844057.
- [33] O. Anicho et al. “Comparative Study for Coordinating Multiple Unmanned HAPS for Communications Area Coverage”. In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 467–474. DOI: 10.1109/ICUAS.2019.8797881.

- [34] Runqi Chai et al. “Overview of Trajectory Optimization Techniques”. In: *Design of Trajectory Optimization Approach for Space Maneuver Vehicle Skip Entry Problems*. Singapore: Springer Singapore, 2020, pp. 7–25. DOI: 10.1007/978-981-13-9845-2\_2. URL: [https://doi.org/10.1007/978-981-13-9845-2\\_2](https://doi.org/10.1007/978-981-13-9845-2_2).
- [35] “Alphabet’s loon and softbank corp.’s hapsmobile complete development of communications payload for hawk30 aircraft”. In: (Feb. 2020). URL: [https://www.hapsmobile.com/en/news/press/2020/20200206\\_01/](https://www.hapsmobile.com/en/news/press/2020/20200206_01/).
- [36] Y. Maguire. *High altitude connectivity: The next chapter*. June 2018. URL: <https://engineering.fb.com/2018/06/27/connectivity/high-altitude-connectivity-the-next-chapter/>.
- [37] A. Westgarth. “Saying goodbye to Loon”. In: (Jan. 2021). URL: <https://medium.com/loon-for-all/loon-draft-c3fceb11f3f>.
- [38] Sven Rühle. “Tabulated values of the Shockley–Queisser limit for single junction solar cells”. In: *Solar Energy* 130 (2016), pp.139–147. ISSN: 0038-092X. DOI: <https://doi.org/10.1016/j.solener.2016.02.015>. URL: <https://www.sciencedirect.com/science/article/pii/S0038092X16001110>.
- [39] *Energy Storage Grand Challenge: Energy Storage Market Report*. Tech. rep. U.S. Department of Energy, Dec. 2020. URL: [https://www.energy.gov/sites/prod/files/2020/12/f81/Energy%20Storage%20Market%20Report%202020\\_0.pdf](https://www.energy.gov/sites/prod/files/2020/12/f81/Energy%20Storage%20Market%20Report%202020_0.pdf).
- [40] *Licerion® high energy density lithium-metal rechargeable batteries*. URL: <https://sionpower.com/products/>.
- [41] *Pathfinder Plus*. Apr. 2002. URL: <https://web.archive.org/web/20030620174825/http://www.dfrc.nasa.gov/Research/Erast/pathfinder.html>.
- [42] Brian Dunbar. *Helios Prototype Solar-Powered Aircraft*. May 2017. URL: <https://www.nasa.gov/centers/dryden/history/pastprojects/Helios/index.html>.
- [43] Yael Maguire and Kevin Waclawicz. “Aquila: What’s next for high-altitude connectivity?” In: (Oct. 2017). URL: <https://engineering.fb.com/2017/10/26/connectivity/aquila-what-s-next-for-high-altitude-connectivity/>.
- [44] Jared Miller. *umf-thesis*. <https://github.com/jarmillemich/umf-thesis>. 2021.
- [45] Philipp Oettershagen. *Conceptual Design and Analysis Tool for solar-powered UAVs*. 2018. URL: [https://github.com/ethz-asl/fw\\_conceptual\\_design](https://github.com/ethz-asl/fw_conceptual_design).

- [46] *Nova-233 G2 Outdoor TDD eNodeB*. Nova-233 G2. Baicells. 2020. URL: <https://www.doubleradius.com/site/stores/baicells/baicells-nova-233-gen-2-enodeb-outdoor-base-station-datasheet.pdf>.
- [47] *Technical Appendix - Application of Kuiper Systems LLC for Authority to Launch and Operate a Non-Geostationary Satellite Orbit System in Ka-band Frequencies*. Tech. rep. July 2019.
- [48] *SPACE X V-BAND NON-GEOSTATIONARY SATELLITE SYSTEM - Attachment A*. Tech. rep. Mar. 2017.
- [49] S. Schopferer et al. “Evaluating the energy balance of high altitude platforms at early design stages”. In: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2016, pp. 170–177. DOI: 10.1109/ICUAS.2016.7502577.
- [50] Will Holmgren et al. *pplib/pplib-python: v0.8.0*. Version v0.8.0. Sept. 2020. DOI: 10.5281/zenodo.4019830. URL: <https://doi.org/10.5281/zenodo.4019830>.
- [51] ns-3 project. *ns-3 Manual*. Jan. 2021.
- [52] Sharan Naribole. *Poisson Pareto Burst Process (PPBP) ns-3 traffic generator*. <https://github.com/sharan-naribole/PPBP-ns3>. 2020.
- [53] Kyle Klein and Julian Neira. “Nelder-Mead Simplex Optimization Routine for Large-Scale Problems: A Distributed Memory Implementation”. In: *Computational Economics* 43.4 (Apr. 2014), pp. 447–461. ISSN: 1572-9974. DOI: 10.1007/s10614-013-9377-8. URL: <https://doi.org/10.1007/s10614-013-9377-8>.
- [54] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.
- [55] Enrico Bertolazzi and Marco Frego. *A Note on Robust Biarc Computation*. 2017. arXiv: 1711.00935 [math.NA].
- [56] Richard Burden, Douglas Faires, and Annette Burden. *Numerical Analysis*. Boston, MA: Cengage Learning, 2014.
- [57] Sean Luke. *Essentials of Metaheuristics*. second. <http://cs.gmu.edu/~sean/book/metaheuristics/> Lulu, 2013.
- [58] John A Nelder and Roger Mead. “A simplex method for function minimization”. In: *The computer journal* 7.4 (1965), pp. 308–313.

- [59] Fuchang Gao and Lixing Han. “Implementing the Nelder-Mead simplex algorithm with adaptive parameters”. In: *Computational Optimization and Applications* 51.1 (Jan. 2012), pp. 259–277. ISSN: 1573-2894. DOI: 10.1007/s10589-010-9329-3. URL: <https://doi.org/10.1007/s10589-010-9329-3>.
- [60] Mark Drela. *XFOIL 6.9 User Primer*. Tech. rep. MIT Aero & Astro, Nov. 2001.
- [61] *Airfoil Tools*. URL: <http://airfoiltools.com/>.
- [62] *U.S. standard atmosphere, 1976*. Technical Memorandum NASA-TM-X-74335. NASA, Oct. 1976.
- [63] URL: <https://www.sagemath.org/>.
- [64] URL: <https://jupyter.org/>.