

Blackbox Optimization for Free-space Quantum Key Distribution

by

Alexandra Maiercan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2024

© Alexandra Maiercan 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis contains work that was done in collaboration with others but to which I made the major contribution. These contributions are listed below.

In **Chapter 1**, Alexandra Maiercan wrote the Introduction.

In **Chapter 2**, Alexandra Maiercan conducted a literature review and compiled the overview of the CS, GPS, MADS, and MBTR algorithms, as well as summarized the procedure of benchmarking BBO algorithms.

In **Chapter 3**,

Simulation:

Alexandra Maiercan created the satellite orbital simulation in Ansys Systems Toolkit (STK) with guidance from Dr. Brendon Higgins, Brian Moffat, and Prof. Thomas Jennewein. When Vinodh Raj Rajagopal Muthu and Dr. Brendon Higgins implemented the QEYSSat MATLAB Simulator's latest format for the time of flight interpolation, Alexandra Maiercan conducted the orbital simulations to support that work.

The most recent version of the QEYSSat MATLAB Simulator has been developed by Ian DSouza, Dr. Jean Philipe Bourgoin, Dr. Brendon Higgins, Vinodh Raj Rajagopal Muthu, Brian Moffat, and Prof. Thomas Jennewein, with minor contributions from Alexandra Maiercan. Alexandra conducted the validation of the eavesdropper module, developed the parameter recording functionality, and implemented the dependency on excess bias voltage for the two parameters: detector efficiency and dark count rate. Prof. Jennewein conceived the initial idea and process for deriving the analytic expressions of these parameter dependencies and Alexandra Maiercan refined this analysis to come up with appropriate models and concrete coefficients.

QKD Analysis Software:

Dr. Higgins and Vinodh Raj Rajagopal Muthu developed and supported the QKD Analysis software.

Simulator automation:

Alexandra Maiercan developed the MATLAB code and graphical user interface for the QEYSSat Simulator automation software with input from Henri Morin, and used it to collect data when she implemented the GPS algorithm. Alexandra Maiercan created a version of the QEYSSat MATLAB Simulator which allows the GUI to access and update its internal parameters for each new simulation.

Optimization implementation:

Alexandra Maiercan wrote the code that implemented the Latin Hypercube Sampling, OrthoMADS, COBYLA, and LINCOA algorithms, with consultations from Prof. Norbert Lütkenhaus and Prof. Thomas Jennewein. Alexandra Maiercan ran the OrthoMADS, COBYLA, and LINCOA optimization algorithms. Alexandra Maiercan collected and processed the data to produce the performance profile graphs.

Alexandra Maiercan wrote the conclusion in **Chapter 4** with guidance from Prof. Thomas Jennewein.

Abstract

The Quantum Encryption and Science Satellite is an experimental proof-of-concept of free-space quantum key distribution.

As part of the mission conception, it is imperative to have an optimal design of all elements of the communication protocol. That is, given all of the possible parameters, we ask which combination will achieve the most secure link? This thesis explores the answer to this question for some specific parameters.

A detailed model of the mission is simulated in MATLAB to obtain data points describing the security of the link over the possible set of parameter values. Then, applying appropriate optimization algorithms, we seek to maximize the key rate and minimize the quantum bit error rate. Mathematically, this task reduces to an optimization problem where the objective is a 2-tuple of key rate and quantum bit error rate, which are indicators of protocol security; and the search space is the set of parameter values input to the simulation. The MATLAB simulation cannot be described analytically and thus represents an oracle or blackbox objective function.

In this thesis, a comparison of optimization protocols is conducted between Mesh Adaptive Direct Search (MADS), and Model-based Trust Region (MBTR) methods.

The investigation of optimization algorithm performance is applied to a subset of real Quantum EncrYption and Science Satellite (QEYSSat) parameters: excess voltage supplied to the detectors, and the mean photon number per pulse for the signal.

Acknowledgements

First and foremost, I would like express all my gratitude to my co-supervisors Prof. Thomas Jennewein and Prof. Norbert Lütkenhaus, who not only gave me the opportunity to learn under their care, and thoughtfully guided my work, but who also gave me an important opportunity for professional development.

As well, a big thank you to Dr. Brendon Higgins, Vinodh Rajagopal, Brian Moffat, for the day-to-day teachings, discussion, and help; and to all researchers and students who came before my time and managed the development/implementation of the QEYSSat simulation – the work in this thesis stands on their shoulders.

The effort that went into completing this research is broad and required a veritable village of friends and family for support. I must express my gratitude to the two University of Waterloo families at the Quantum Photonics Laboratory (QPL) and Optical Quantum Communications Theory Group (OQCT) who consistently brought out the fun in all the time that we spent and all that we did together. And finally, a sincere and heartfelt thank you goes out to my loving parents, Maxim and Ekaterina, as well as my brilliant husband, Serban. It is hard to put into words all that you have done to get me here.

Table of Contents

Author's Declaration	ii
Statement of Contributions	iii
Abstract	v
Acknowledgements	vi
List of Figures	ix
List of Tables	x
List of Algorithms	xi
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Outline	2
2 The Optimization Mathematics	3
2.1 Problem Definition	3
2.2 Blackbox Objective Function	4
2.2.1 BBO Challenges	5
2.3 Literature review	6
2.3.1 Mesh Adaptive Direct Search	7
2.3.2 Model-based Trust Region	28
2.3.3 Convergence	40
2.3.4 Bi-objective Optimization	41
2.3.5 Benchmarking	44
3 The Optimization Application	50
3.1 Overview	50
3.1.1 Free-space QKD	51
3.1.2 Satellite Link Simulation	52

3.2	Optimizing QKD	58
3.2.1	Objectives: Key Rate and QBER	58
3.2.2	Argument: Detector Voltage	65
3.2.3	Argument: Mean Photon Number	72
3.2.4	Methodology	73
3.2.5	Surrogate Models	82
3.3	Results	86
4	Conclusion	90
4.1	Closing Remarks	92
	References	93
	APPENDICES	105
A	Comparing Paths	106

List of Figures

1	Blackbox function.	4
2	Visualization of CS algorithm iterations.	9
2	Visualization of CS algorithm iterations...continued.	10
2	Visualization of CS algorithm iterations...continued and finalized.	11
3	Visualization of positive basis.	14
4	Visualization of GPS algorithm iterations.	20
4	Visualization of GPS algorithm iterations...continued.	21
4	Visualization of GPS algorithm iterations...continued and finalized.	22
5	Visualization of MADS algorithm iterations.	26
5	Visualization of MADS algorithm iterations...continued and fi- nalized.	27
6	Visualization of MBTR algorithm iterations.	29
7	MBTR Flowchart.	38
8	Contour plot.	47
9	Performance profiles.	49
10	Three current satellite QKD missions.	53
11	The four parts of the QEYSSat simulation.	55
12	Time varying link data for a selected pass.	57
13	Simplified example: obtaining finite-size Sifted Key.	63
14	Detector efficiency as a function of excess bias voltage.	68
15	DCR of APDs after irradiation.	70
16	Dark count rate: experimentally vs. computationally derived.	71
17	Set of starting points.	77
18	3D Surfaces for Key Rate and QBER from Grid Search.	78
19	Data flow of optimization software.	79
20	QKD Analysis software interface for key rate calculation.	83
21	Surrogate optimization: “shortened satellite pass”.	85
22	Key Rate vs. Voltage: pre- and post- error correction.	86
23	Final performance profiles.	88
24	Paths taken by the three algorithms.	107

List of Tables

1	Non-comprehensive list of simulation components and examples of their parameters.	59
2	Signal+Decoy+Vacuum Pulse Distribution.	72
3	Example data for performance profiles, $\tau = 50\%$	73

List of Algorithms

1	:	Coordinate Search (CS)	8
2	:	Generalized Pattern Search (GPS)	18
3	:	Mesh Adaptive Direct Search (MADS)	24
4	:	Model-Based Trust Region (MBTR)	36
5	:	Latin Hypercube Sampling (LHS)	75

1 Introduction

1.1 Motivation

Quantum Key Distribution (QKD) is a solution to one of the biggest challenges presented by the impending advent of the fault-tolerant quantum computer: encryption. Relying on the fundamental physics of quantum mechanics, QKD describes a method for securely sharing a secret encryption key between two distant parties. More generally, QKD achieves security of communication by alerting the parties when there is an eavesdropper.

The theory of QKD as a secure solution is robust and well-developed. Now, we are exploring how to implement the theory experimentally. One of the most promising versions of a solution is the satellite-based free-space (i.e., atmosphere and/or outer space) transmission architecture. It describes a system where the encryption key is sent as a string of single-photon pulses which act as bits of information, also called ‘qubits’. Note, the term *free-space* refers to the Earth’s atmosphere and outer space; this is in contrast to the more traditional means of sending photons/light which is through a fiber cable.

Building such a communication satellite necessitates an ultra-precise design and manufacturing process. To that end, prior to the start of the mission, all elements of the communication protocol must be simulated and optimized. Consider that the sender emits pulses (bits of information) from a laser that contain a single photon at a time. That is, the whole system must be optimized to precisely transmit a single photon particle from a ground station on Earth, to a chip-sized detector on board a satellite 500 km away. The detector alone has numerous parameters to consider, including its dark count rate, afterpulsing rate, timing jitter (error), and more. Incorrectly choosing any of these parameters can make or break the satellite link. A similar logic applies to any other component involved in the protocol.

The size and complexity of the QKD simulations requires careful consideration for which optimization algorithms would be most complementary.

This thesis aims to explore exactly that question – which algorithm is best suited for optimizing a simulation of a free-space QKD protocol. In the absence of a clear precedent for how to approach this, we implement and evaluate a variety of methods detailed in Chapter 2.

1.2 Thesis Outline

Chapter 2 presents the state-of-the-art optimization algorithms. Specifically, those aimed at solving blackbox problems. We explore methods for performance benchmarking for a handful of optimization methods appropriate for the QKD simulation.

Chapter 3 sees to the application of the optimization algorithms to the QEYSSat mission simulation. We endeavor to determine an optimal voltage level for the single photon detectors. As well, we work on determining the value of the average photon number per pulse for the QKD protocol that will maximize security, i.e., minimize quantum bit error rate and maximize key rate. Thirty-nine optimization tasks are carried out on the QEYSSat simulator to compare the performance of direct search, and model-based algorithms.

Chapter 4 concludes the thesis with a summary of the work presented, and perspectives for future work.

2 The Optimization Mathematics

2.1 Problem Definition

Optimization can be described as the mathematical study of determining optimal solutions for a variety of problems often derived from real world and physical challenges. In other words, making the best use of scarce resources. This includes engineering design (including bio-engineering), environmental management (agriculture, forestry, climatology, meteorology), investment portfolios, logistics, and more. For the interested reader, prominent examples of applications are cited: [1, 2, 3, 4, 5, 6, 7, 8]. Though in our case, we specifically explore an optimization problem as it applies to quantum communication satellite mission design.

More formally, an optimization problem can be defined as follows. Given a subset $\mathcal{S} \subset \mathbb{R}^n$ (solution space), aka the solution set, we seek to find the *minimum value* of a function $f : \mathcal{S} \rightarrow \mathbb{R}$, denoted:

$$y^* := \min_x \{f(x) : x \in \mathcal{S}\}. \quad (1)$$

Frequently, one is also interested in solving for the points where the minimum value of the function, y^* , is achieved. In this case, we say that we seek the *argument*:

$$x^* \in \arg \min_{x \in \mathcal{S}} f(x) := \{x \in \mathcal{S} : f(x) = y^*\}. \quad (2)$$

Note, an optimal function value may also be a maximum as much as a minimum. The two may be solved interchangeably, as

$$\max_x f(x) = -\min_x (-f(x)).$$

and

$$\arg \min_{x \in \mathcal{S}} f(x) = \arg \max_{x \in \mathcal{S}} (-f(x)).$$

Sources for further reading for introductory concepts in optimization include the introductory chapters of the following texts [9, 10, 11].

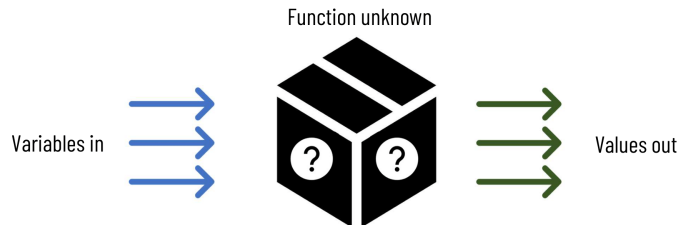


Figure 1: **Depiction of a blackbox whose contents are a mystery. Variables go in, and values come out, yet the transformation protocol within the box is not accessible.**

2.2 Blackbox Objective Function

In some cases, it happens that the objective function does not have a closed-form expression. This is called Blackbox Optimization. Here, *blackbox* refers to the objective function as a box where variables go in and values come out, but the function by which the variables are processed inside the box is not available — either not usable or more commonly, not obtainable.

There are three main reasons to implement BBO:

- the objective function takes the form of a computer simulation which cannot be expressed mathematically
- the objective function takes the form of a laboratory experiment
- the objective function is sufficiently noisy that the gradient information is not reliable.

Importantly, if gradient information is available, then in nearly all cases it would be less efficient to use derivative-free methods (DFO) and blackbox methods (BBO) than gradient-based methods. [10, 11].

Note that although often used interchangeably, the terms DFO and BBO are not the same. Rather, $\text{DFO} \subset \text{BBO}$. DFO generally focuses on continuous input variables, and methods that can be analyzed to prove convergence, and respective stopping criteria. BBO makes no assumptions about continuity, differentiability, or smoothness of the outputs. Also, it explores heuristic methods, which in contrast to derivative-free methods, often lack formal convergence analysis [12, 10, 13].

2.2.1 BBO Challenges

Pathological Functions

In order to conduct rigorous analyses of blackbox problems, it is often assumed that the objective and constraints are well-behaved (continuous, differentiable). However, in some implementations, the blackbox and/or constraints are not well-behaved and this presents a significant challenge [12]. One such example are Monte-Carlo simulations, where sequential evaluations of the objective function yield different values due to statistical variability. This is particularly relevant to the specific QKD simulation in this thesis which sees probabilistic sampling at various points within the simulation. We see in Section 3.3 that one way of addressing the problem is by running many optimization scenarios. A single optimization run may not give a sufficiently correct answer [14].

Computational Resources

In many cases, function evaluations are costly. That is, either running a single simulation/experiment is time intensive, or the time to obtain the aggregate data set is significant. Thus, a topic of interest in BBO methods is how to obtain the most information possible out of each simulation/experiment. [15]

Particularly in the case of Monte-Carlo simulations, one way to reduce the cost of evaluation is to use a surrogate optimization [16]. This method involves running the simulation for a shorter period of time, and is explored in more detail in Section 3.2.5.

2.3 Literature review

Foundational Methods

Historically, optimization problems which do not have an analytic expression for the gradient, have been tackled by finite difference methods to estimate gradients between discrete points in a domain [13]. This approach does not translate well to blackbox optimization problems because it requires far too many costly function evaluations to be practical [17]. Modern methods for addressing blackbox optimization are loosely divided into two classes: *direct search methods*, and *model-based methods*. Each of these classes addresses the problem in a fundamentally different way, and has its own advantages and disadvantages [12].

Direct search methods typically work by sampling the search space at carefully chosen points, and doing so without using derivatives. Established algorithms include: Coordinate Search (CS) [18], Generalized Pattern Search (GPS) [19], Mesh Adaptive Direct Search (MADS), and their variants [10, 11, 20]. Search methods are great when the optimization problem is non-smooth and a highly accurate local solution is sufficient, over say a global solution.

In comparison, model-based algorithms tend to address global search efficiently, but are challenged when the objective function is falsely assumed to be smooth. This is a rule of thumb. Exceptions exist, for example with [21] exploring direct search algorithms for global optimization, and [22] working on model-based methods for non-smooth objective functions.

More recently, there have been works published exploring more advanced methods that are model-based: using radial basis functions [23], Gaussian processes [24], weighted regressions [25], and even approaches for non-smooth functions [26, 27, 28]. For the purposes of this thesis, we will be exploring three specific approaches. Namely, Mesh Adaptive Direct Search (MADS), Model-based Trust Region (MBTR) using both linear and quadratic modeling. These are well-established methods rather than state-of-the-art. The choice of implementing well-established algorithms was made

for pragmatic reasons, as the goal of this project is to obtain *reliable* optimal values for parameters in the QEYSSat simulation, as they are intended to be implemented in the QEYSSat mission.

2.3.1 Mesh Adaptive Direct Search

The MADS algorithm is an evolution of the Generalized Pattern Search (GPS), which itself is an evolution of the Coordinate Search (CS) algorithm. To ease the reader into the theory of MADS, we start with the more intuitive CS algorithm and develop from there.

The coordinate search algorithm, as its name implies, is based on the idea that one can seek a minimizer by searching along the coordinate directions, $\pm e_i \in \mathbb{R}^n$, about an incumbent $x^{(k)}$ value. At every iteration, if an improvement is found, the algorithm samples the points in the immediate vicinity along the coordinate directions (for some defined radius $\delta^{(k)}$), otherwise it refines the search grid (reduces $\delta^{(k)}$ by some pre-determined factor). Algorithm 1 provides the pseudo-code. Figure 2 shows a sample visualization of the steps taken by the CS algorithm as it works towards a minimum. Note, the *poll set* is defined as $P^{(k)} := \{x^{(k)} \pm \delta^{(k)} e_i : i \in \mathbb{N}\}$. It is the set of points adjacent $x^{(k)}$ along the coordinate directions, which are $\delta^{(k)}$ distance away.

Algorithm 1 : Coordinate Search (CS)

Given $f : \mathbb{R}^n \mapsto \mathbb{R}$, and an initial point at $x^{(0)} \in \mathbb{R}^n$,

initialize

$\delta^{(0)} \in (0, \infty)$ step length

$\epsilon_{end} \in [0, \infty)$ termination tolerance

$k \leftarrow 0$ iteration counter

repeat

if $f(t) < f(x^{(k)})$ for some $t \in P^{(k)} := \{x^{(k)} \pm \delta^{(k)} e_i : i \in \mathbb{N}\}$ **then**

$x^{(k+1)} \leftarrow t$

$\delta^{(k+1)} \leftarrow \delta^{(k)}$

else

$x^{(k+1)} \leftarrow x^{(k)}$

$\delta^{(k+1)} \leftarrow \frac{1}{2} \delta^{(k)}$

end if

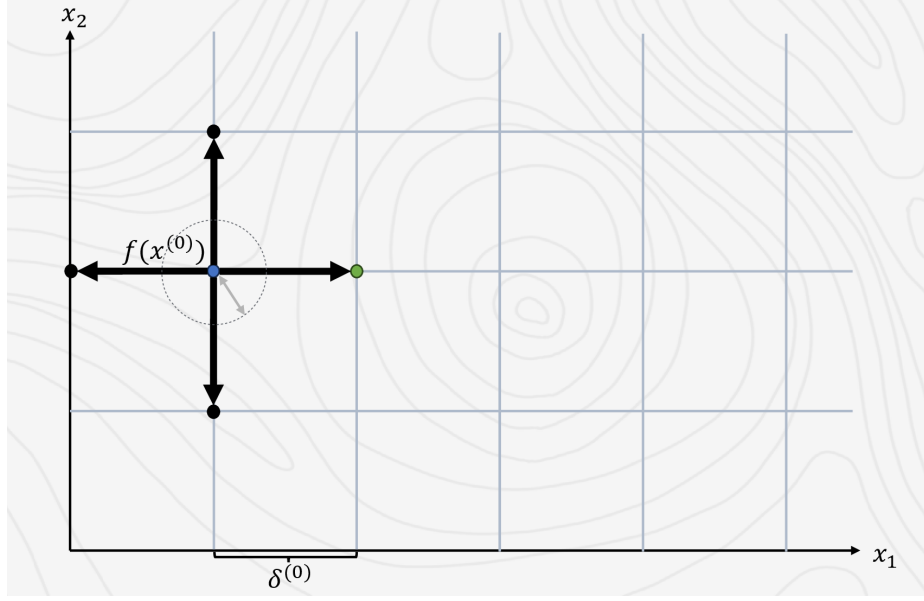
$k \leftarrow k + 1$

until $\delta^{(k)} < \epsilon_{end}$

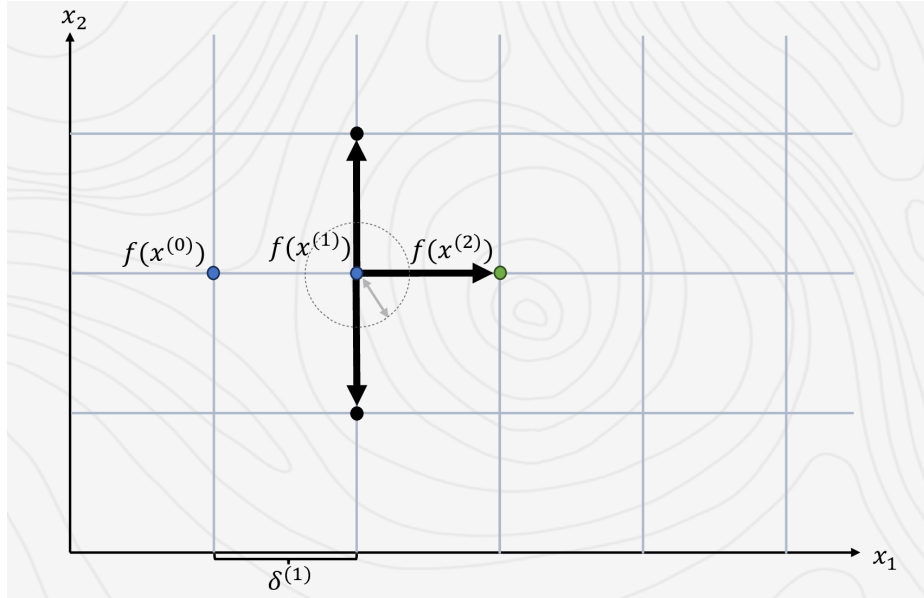
There are several improvements that the GPS algorithm makes to CS. Namely,

- Including an additional early (“search”) step in the algorithm to choose a better starting point. This turns out to be important, as the starting point in such a search algorithm can determine whether or not the algorithm succeeds.
- Including more varied directions of search beyond $\pm e_i \in \mathbb{R}^n$; a matter of efficiency. More specifically, replacing the poll set $P^{(k)} := \{x^{(k)} \pm \delta^{(k)} e_i : i \in \mathbb{N}\}$ with $\{x^{(k)} + \delta^{(k)} d : d \in \mathbb{D}^{(k)}\}$, where $\mathbb{D}^{(k)}$ is a finite set called the *poll directions*, and can vary from iteration to iteration.
- Including options for both refining and coarsening the search grid, $\delta^{(k)}$. Coarsening the grid after a successful iteration allows the algorithm to escape local

Figure 2: Visualization of CS algorithm iterations.

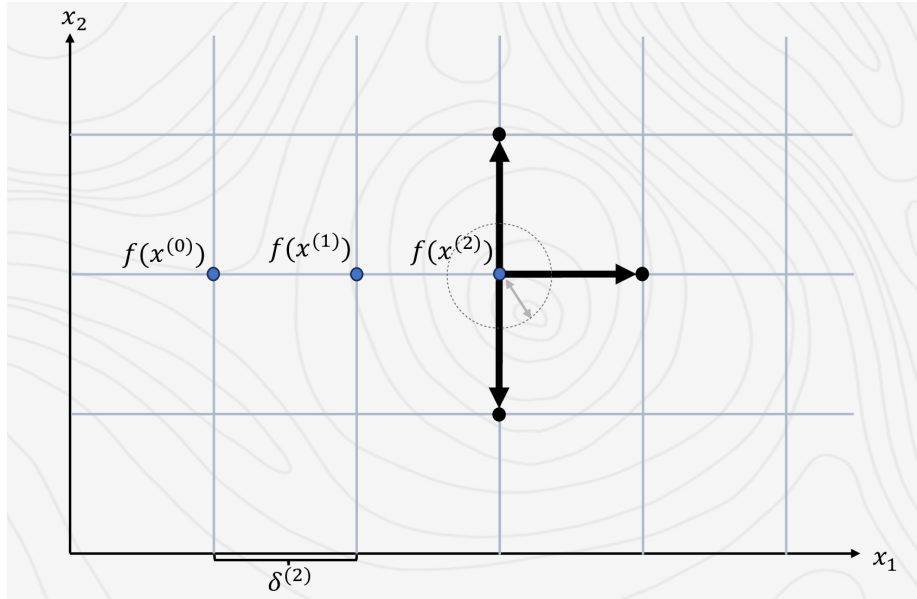


(a) CS algorithm. Iteration $k = 0$. Beginning the search for a minimizer at point $x^{(0)}$, the four coordinate directions are probed seeking a smaller objective value. One is found to the right of $x^{(0)}$. This is a “successful” iteration. Note, the gray circle about $x^{(0)}$ represents the stopping tolerance ϵ_{end} .

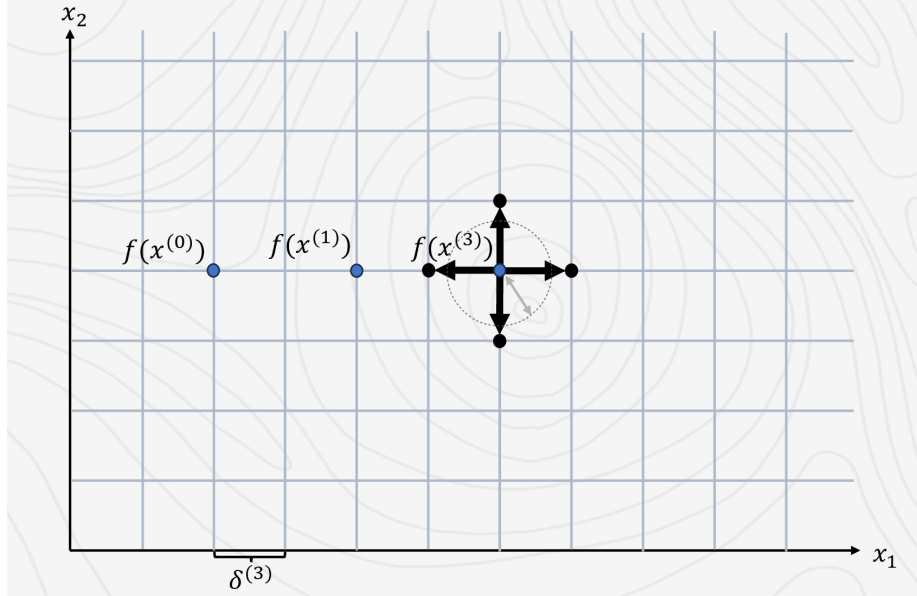


(b) CS algorithm. Iteration $k = 1$. After a successful iteration, the incumbent point is moved to the new minimal value. Coordinate search is repeated for the same grid parameter $\delta^{(k)}$; again successful.

Figure 2: **Visualization of CS algorithm iterations...**continued.

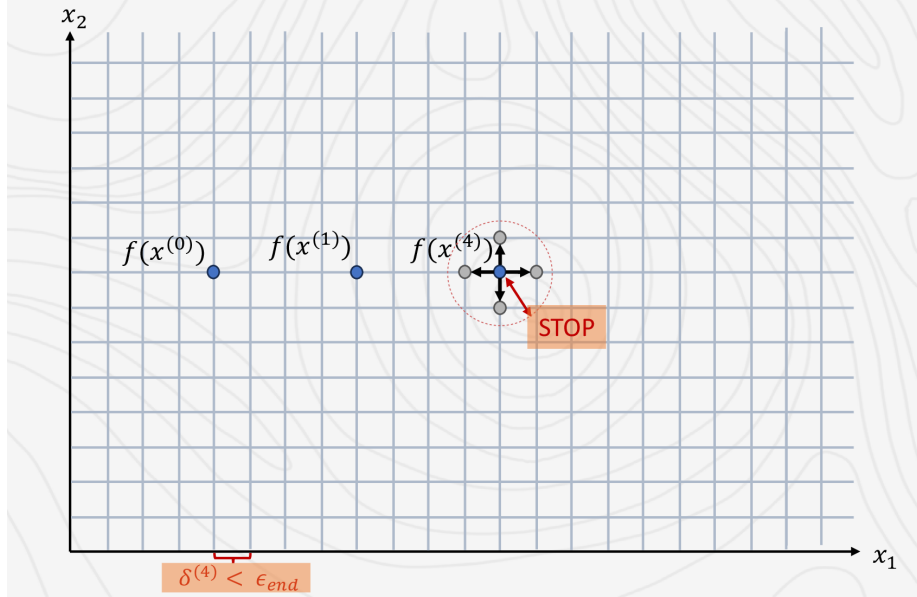


(c) CS algorithm. Iteration $k = 2$. No smaller value of the objective function is found at the three new points adjacent $x^{(2)}$; “unsuccessful” iteration. Next, the search grid is refined, $\delta^{(3)} \leftarrow \frac{1}{2}\delta^{(2)}$.



(d) CS algorithm. Iteration $k = 3$. No smaller value of the objective function is found at the four points adjacent $x^{(3)}$; “unsuccessful” iteration. Next, the search grid is refined, $\delta^{(4)} \leftarrow \frac{1}{2}\delta^{(3)}$.

Figure 2: Visualization of CS algorithm iterations...continued and finalized.



(e) CS algorithm. Iteration $k = 3$. The value of $\delta^{(4)}$ falls below the threshold ϵ_{end} , thus the algorithm is terminated.

minima.

Some definitions are needed to rigorously describe the enhanced variability of search directions offered by the GPS algorithm. The following three definitions can be thought of as analogous to the basic definitions presented in linear algebra for 1) the span of a set, 2) linear independence, and 3) a basis [29, 30, 31]. **These definitions will allow for the reduction in computational cost per optimization iteration because it will require fewer positive basis vectors to span the domain.**

Definition 2.1: Positive Spanning Set

Suppose $\mathbb{D} \subseteq \mathbb{R}^n$ is a finite discrete set. Then the positive span of $\mathbb{D}$, denoted $pspan(\mathbb{D})$, is the set of non-negative linear combinations of vectors in $\mathbb{D}$:

$$pspan(\mathbb{D}) = \left\{ \sum_{i=1}^m \lambda_i d^i : \lambda_i \geq 0, d^i \in \mathbb{D}, m \in \mathbb{N} \right\} \subseteq \mathbb{R}^n.$$

The set $\mathbb{D}$ is a positive spanning set for $\mathbb{R}^n \iff pspan(\mathbb{D}) = \mathbb{R}^n$. Note, $pspan(\mathbb{D}) \simeq cone(\mathbb{D})$, which is a more common notation, and constitutes a polyhedral cone.

Definition 2.2: Positive Linear Independence

The set $\mathbb{D}$ is called positive linearly independent $\iff d \notin pspan(\mathbb{D} \setminus \{d\})$, $\forall d \in \mathbb{D}$.

Definition 2.3: Positive Basis

The set $\mathbb{D}$ is called a positive basis $\iff$ it is both a positive spanning set for $\mathbb{R}^n$ and is positive linear independent.

Note, the distinction between spanning set and positive spanning set (p.s.s.), is that the coefficients of the linear combinations must be ≥ 0 in the p.s.s.. As an example, the coordinate vectors $\{e_1, e_2\}$ do not form a p.s.s. since their linear combination with non-negative coefficients does not span all of $\mathbb{R}^2$. On the other hand, the set

$\{(0, 1), (0.94, -0.33), (-0.94, -0.33)\}$ of three vectors evenly spaced in $\mathbb{R}^2$ is a p.s.s., see figure 3, but moreover, is a positive basis since the vectors are positive linearly independent. Removing any of the vectors would preclude the set from (positive) spanning $\mathbb{R}^2$.

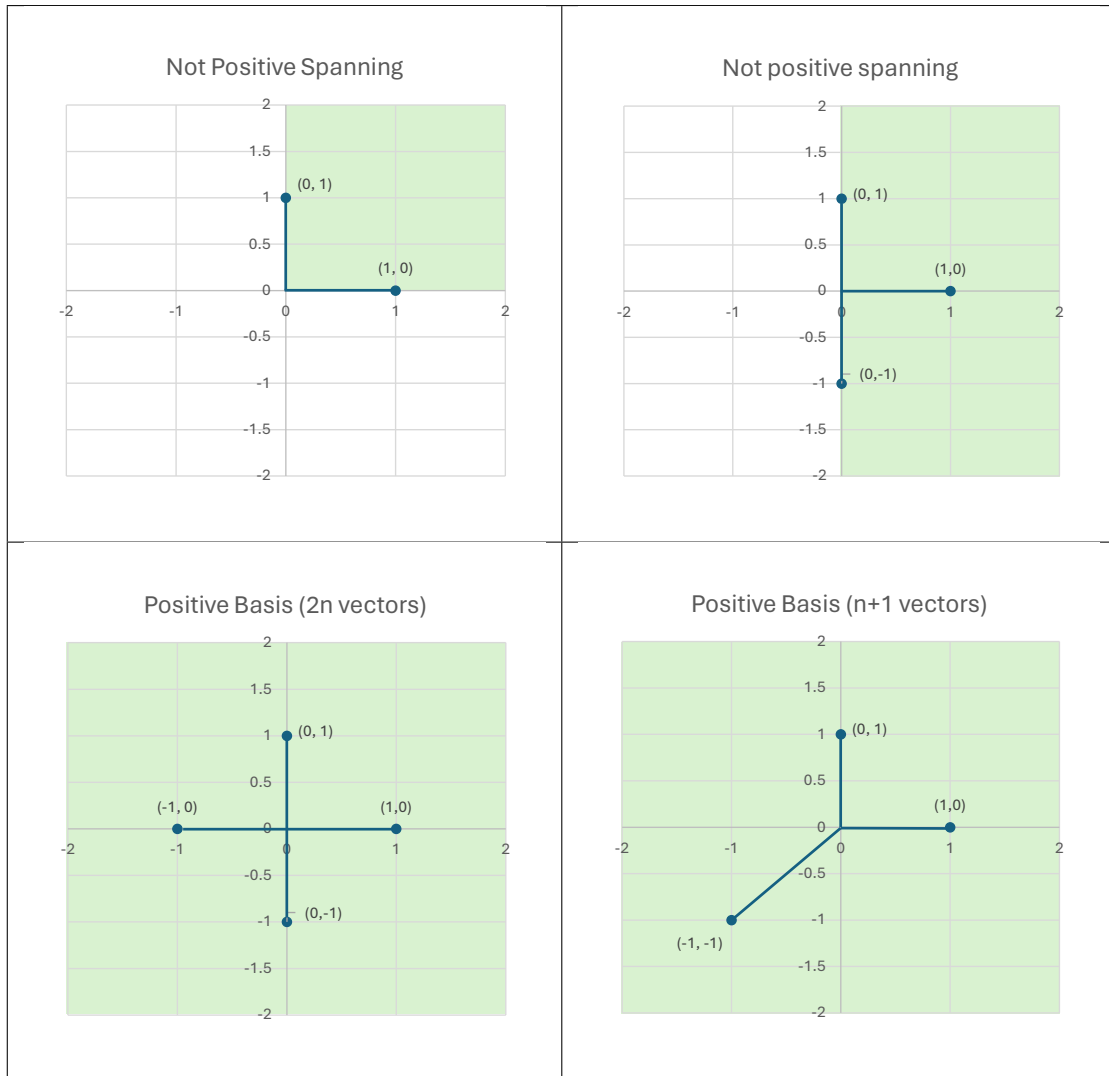
For example, using the coordinate directions to form a positive basis would require at least $n + 1$ vectors, and at most $2n$ vectors:

$$\mathbb{D}_{n+1} = \left\{ e_1, \dots, e_n, -\sum_{i=1}^n e_i \right\},$$

$$\mathbb{D}_{2n} = \{e_1, \dots, e_n, -e_1, \dots, -e_n\}.$$

It can be shown that $n + 1$ and $2n$ are the lower and upper limits for the cardinality of any positive basis. See lemma 2.1, also referenced in [32, 33]. Importantly, the advent of the positive basis saw a near halving of the computational cost per iteration as compared to the traditional basis/generating sets used in earlier pattern search methods [19, 29]. This is because a positive basis provides components that can access every half-space of the domain while not requiring that each coordinate axis be represented by two separate vectors (positive and negative). Beyond re-defining the search directions, the GPS algorithm also allows for the transformation of the search grid itself – now called *the mesh*.

Figure 3: The standard basis vectors $\{e_1, e_2\}$ do not span all of $\mathbb{R}^2$ by taking positive linear combinations; they only span the first quadrant. Using the standard basis vectors, it will require four of $\{e_1, e_2, -e_1, -e_2\}$ to create a positive basis. Alternatively, in the bottom right graph, we see how a positive basis can be generated with fewer than $2n$ vectors. The shaded green areas represent the positive span of the blue vectors.



Lemma 2.1

A positive basis of $\mathbb{R}^n$ has at least $n + 1$ elements and at most $2n$ elements.
Ref. [32].

Proof

Assume that $\mathbb{D} = \{d_1, \dots, d_p\}$ is a positive basis of $\mathbb{R}^n$.

We will show that for some k , the set $\mathbb{D} \setminus \{d_k\}$ has at least n elements, so that $\mathbb{D}$ (inclusive of $\{d_k\}$) must have at least $n + 1$ elements.

Since $\mathbb{D}$ is a positive basis $\implies \mathbb{D}$ is a p.s.s. $\iff pspan(\mathbb{D}) = \mathbb{R}^n$.

Fix some $k \in \{1, \dots, p\}$.

$\implies \exists$ nonnegative $(\lambda_1, \dots, \lambda_p)$ such that $-d_k = \sum_{i=1}^p \lambda_i d_i$.

Subtracting $\lambda_k d_k$ from both sides:

$$\begin{aligned} -d_k - \lambda_k d_k &= \sum_{i=1}^p \lambda_i d_i - \lambda_k d_k \\ -(1 + \lambda_k) d_k &= \sum_{i=1}^{k-1} \lambda_i d_i + \sum_{i=k+1}^p \lambda_i d_i \\ d_k &= \frac{-1}{(1 + \lambda_k)} \left[\sum_{i=1}^{k-1} \lambda_i d_i + \sum_{i=k+1}^p \lambda_i d_i \right]. \end{aligned}$$

$\implies d_k \in span(\mathbb{D} \setminus \{d_k\})$. Note this is the regular $span()$, not $pspan()$.

$\implies span(\mathbb{D} \setminus \{d_k\}) \supseteq span(\mathbb{D}) \supseteq pspan(\mathbb{D}) = \mathbb{R}^n$.

We conclude that the set $\mathbb{D} \setminus \{d_k\}$ spans $\mathbb{R}^n$ and thus has at least n elements, so that $\mathbb{D}$ (inclusive of $\{d_k\}$) must have at least $n + 1$ elements.

Next we would show that the maximal cardinality of a positive basis is $2n$.

This proof can take many pages when expressed completely, so it is omitted but a consise version can be found in [33]. We leave the reader with the following intuition: if $\mathbb{D} \subset \mathbb{R}^n$ is linearly independent then it contains at most n vectors. In the case of a positive basis, the coefficients in the linear combinations of the basis vectors must be nonnegative. Thus every basis vector would have to have its negative counterpart included in the positive basis set, leading to a cardinality of $2n$. $\square$

Definition 2.4: Mesh

Let $G \in \mathbb{R}^{n \times n}$ be a non-singular matrix.

Let $Z \in \mathbb{Z}^{n \times p}$ be a matrix such that its columns form a p.s.s. of $\mathbb{R}^n$.

We define the matrix $D = GZ$, whose columns represent the set of search direction, $\mathbb{D}$. Then the mesh generated by D centered at the incumbent solution $x^{(k)} \in \mathbb{R}^n$ of coarseness $\delta^{(k)} > 0$ is defined to be the discrete set of points

$$M^{(k)} := \{x^{(k)} + \delta^{(k)} Dy : y \in \mathbb{N}^p\} \subset \mathbb{R}^n,$$

where $\mathbb{N}^p$ is the set of p -tuples of natural numbers, and $\delta^{(k)}$ is the mesh size parameter.

From definition 2.4, we further define the set $\mathbb{D}$ to be the columns of matrix D which form a p.s.s. of $\mathbb{R}^n$ (see lemma 2.2 for a detailed derivation). Subsequently, $\mathbb{D}$ forms the possible search directions for the GPS algorithm.

Lemma 2.2

The columns of D form a positive spanning set (p.s.s.) of $\mathbb{R}^n$.

Proof

Recall the matrix $D = GZ$, where $G \in \mathbb{R}^{n \times n}$ is non-singular, and $Z \in \mathbb{Z}^{n \times p}$ whose columns (Z_{col}) form a p.s.s. of $\mathbb{R}^n$. We show that applying an invertible transformation (G) to a p.s.s. (Z_{col}) results in another p.s.s. (columns of D , aka $\mathbb{D}$). For any $x \in \mathbb{R}^n$, let $x' = G^{-1}x$.

Since $Z_{col} = \{z_1, \dots, z_p\}$ is a p.s.s. $\implies \exists \lambda_1, \dots, \lambda_p \geq 0$ such that

$$x' = \sum_{i=1}^p \lambda_i z_i.$$

Then multiplying by G we get

$$\begin{aligned} x = Gx' &= G \sum_{i=1}^p \lambda_i z_i \\ &= \sum_{i=1}^p \lambda_i (Gz_i) \end{aligned}$$

Since this holds $\forall x \in \mathbb{R}^n \implies \{Gz_1, \dots, Gz_p\} = GZ_{col} = \mathbb{D}$, i.e., $x \in pspan(\mathbb{D})$, and thus the columns of D form a p.s.s. in $\mathbb{R}^n$.

□

Algorithm 2 presents the pseudo-code for GPS, and Figure 4 visualizes some iterations.

Algorithm 2 : Generalized Pattern Search (GPS)

Given objective $f : \mathbb{R}^n \mapsto \mathbb{R}$, and an initial point at $x^{(0)} \in \mathbb{R}^n$,

initialize

$\delta^{(0)\dagger} = 1$ ▷ mesh size
 $D = GZ$ ▷ positive spanning matrix
 $\tau^\dagger = \frac{1}{2}$ ▷ mesh size adjustment
 $\epsilon_{end}^\dagger = 0.01$ ▷ tolerance for termination
 $k \leftarrow 0$ ▷ iteration counter

repeat

1. SEARCH STEP (Heuristic) ▷ candidate mesh points

if $f(t) < f(x^{(k)})$ for some $t \in S^{(k)} \subset M^{(k)}$ **then**

$x^{(k+1)} \leftarrow t$
 $\delta^{(k+1)} \leftarrow \frac{\delta^{(k)}}{\tau}$

else

2. POLL STEP (Directions search) ▷ neighboring mesh points

 select p.s.s. $\mathbb{D}^{(k)} \subseteq \mathbb{D}$

if $f(t) < f(x^{(k)})$ for some $t \in P^{(k)} = \{x^{(k)} + \delta^{(k)}d : d \in \mathbb{D}^{(k)}\}$ **then**

$x^{(k+1)} \leftarrow t$
 $\delta^{(k+1)} \leftarrow \frac{\delta^{(k)}}{\tau}$

else ▷ else $x^{(k)}$ is a local minimizer

$x^{(k+1)} \leftarrow x^{(k)}$
 $\delta^{(k+1)} \leftarrow \tau\delta^{(k)}$

end if

end if

$k \leftarrow k + 1$

until $\delta^{(k)} < \epsilon_{end}$

[†] Concrete parameter value set for the sake of example.

GPS has a search step which did not exist in CS. This additional step can be called at each iteration, or it can be implemented only once at the beginning of the protocol to establish a favourable starting point $x^{(0)} \in \mathbb{R}^n$. In Figure 4 it is only implemented once at the beginning for the sake of brevity. Figure 4 also illustrates how the generating and basis matrices, G and Z respectively, serve to define the mesh and polling directions. Although not explicitly defined in the pseudo-code, Ref. [20] describes how to define these matrices for a practical implementation of the MADS algorithm. As well, note that the search step is mentioned as “Heuristic”, referring to methods such as the Nelder-Mead algorithm, though alternatively a handful of line search iteration can work well instead. For more details, see Chapter 7 of Ref. [10].

The MADS Algorithm further extends the idea of variable search directions to include even more directions. Moreover, it affords better convergence by implementing an additional degree of freedom for adjusting the search grid – *the frame*.

Definition 2.5: Frame

Let $G \in \mathbb{R}^{n \times n}$ to be non-singular.

Let $Z \in \mathbb{Z}^{n \times p}$ to be such that its columns form a p.s.s. of $\mathbb{R}^n$.

Let $D = GZ$.

Fix the mesh size parameter $\delta^{(k)} > 0$, and define the frame size parameter $\Delta^{(k)} \geq \delta^{(k)}$.

Then the frame generated by D and centered at the incumbent solution $x^k \in \mathbb{R}^n$ of coarseness $\Delta^{(k)}$ is defined to be

$$F^{(k)} := \{x \in M^{(k)} : \|x - x^k\|_\infty \leq \Delta^{(k)}b\}.$$

Here, $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$.

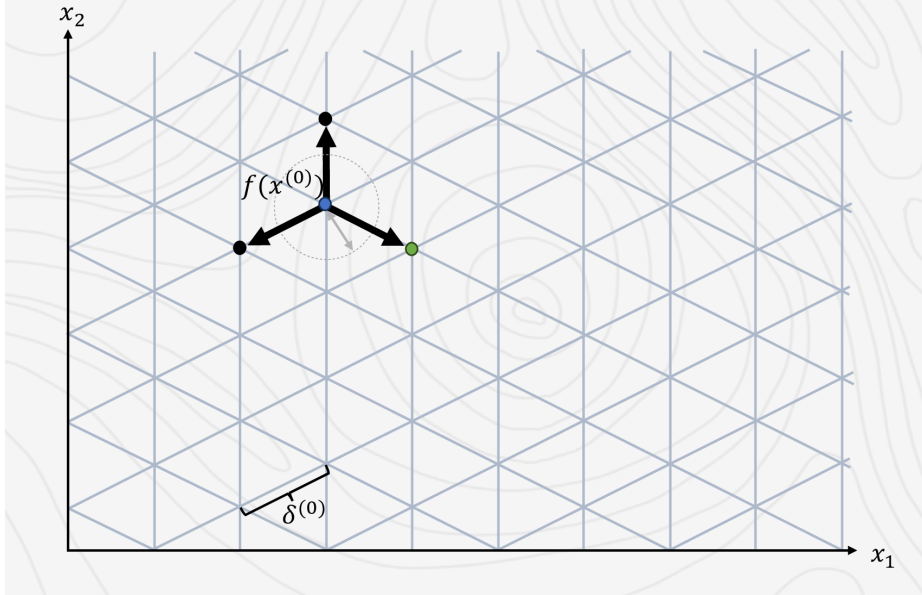
Recall, $\|x\|_\infty := \max\{|x_1|, \dots, |x_n|\}$.

The frame is a type of trust region.

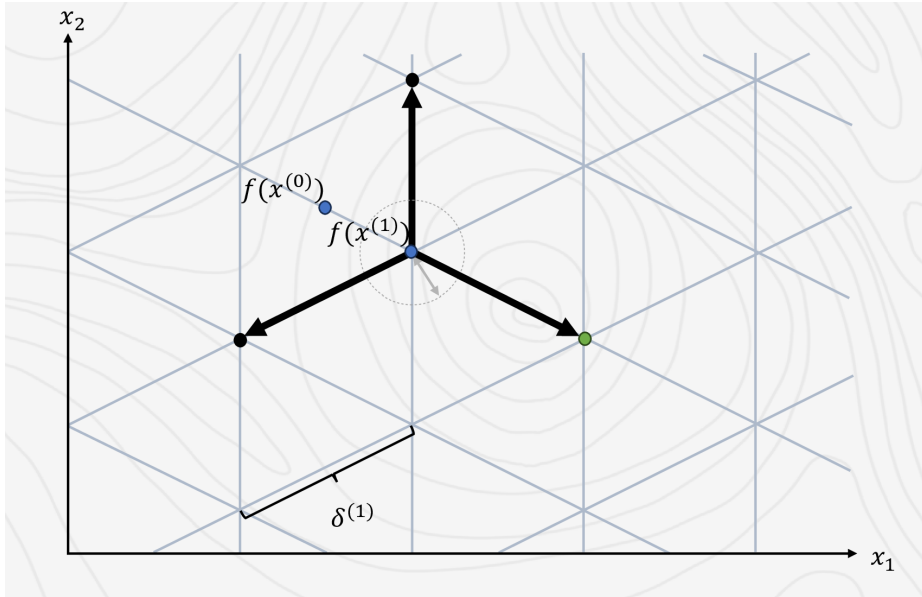
Not only is MADS using the mesh size parameter $\delta^{(k)}$, but it is also restricting the

Figure 4: **Visualization of GPS algorithm iterations.**

$$\mathbf{G} = \begin{bmatrix} \sqrt{3} & 0 \\ -1 & 2 \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

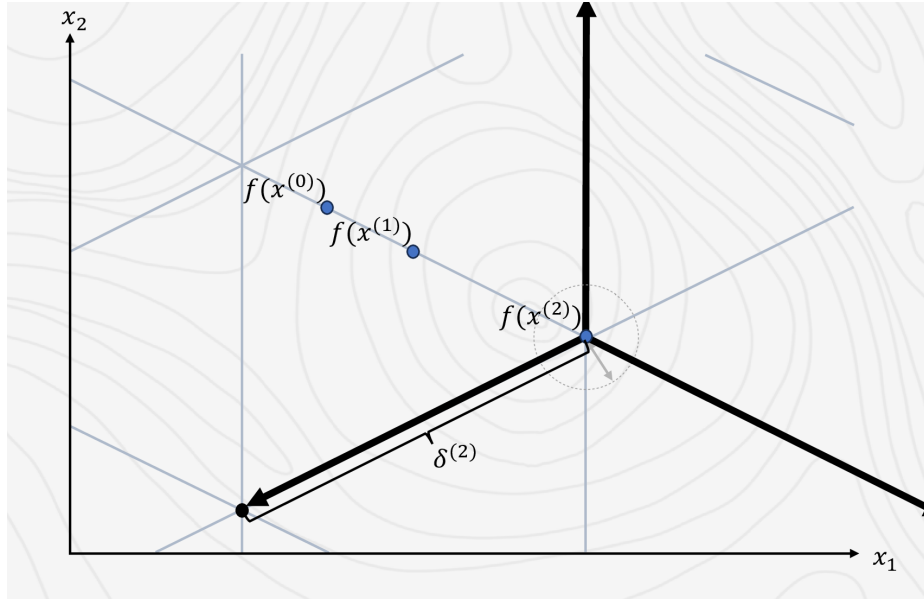


(a) GPS algorithm. Iteration $k = 0$. As part of the initialization, we run the latin hypercube sampling in the Search Step which is implemented to find an appropriate starting point, $x^{(0)}$. The Search Step is not re-visited in this example. Successful iteration.

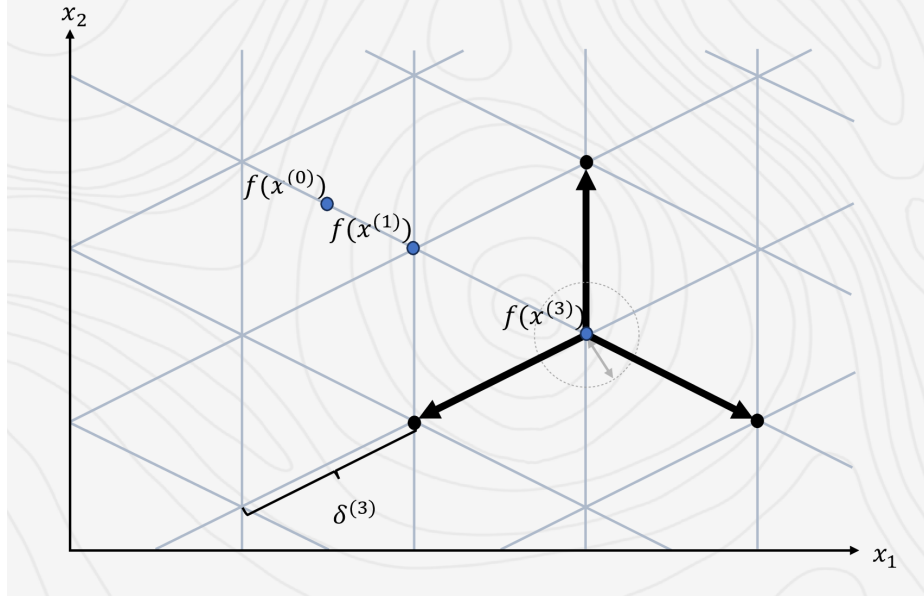


(b) GPS algorithm. Iteration $k = 1$. The mesh is coarsened. Successful iteration.

Figure 4: **Visualization of GPS algorithm iterations...continued.**

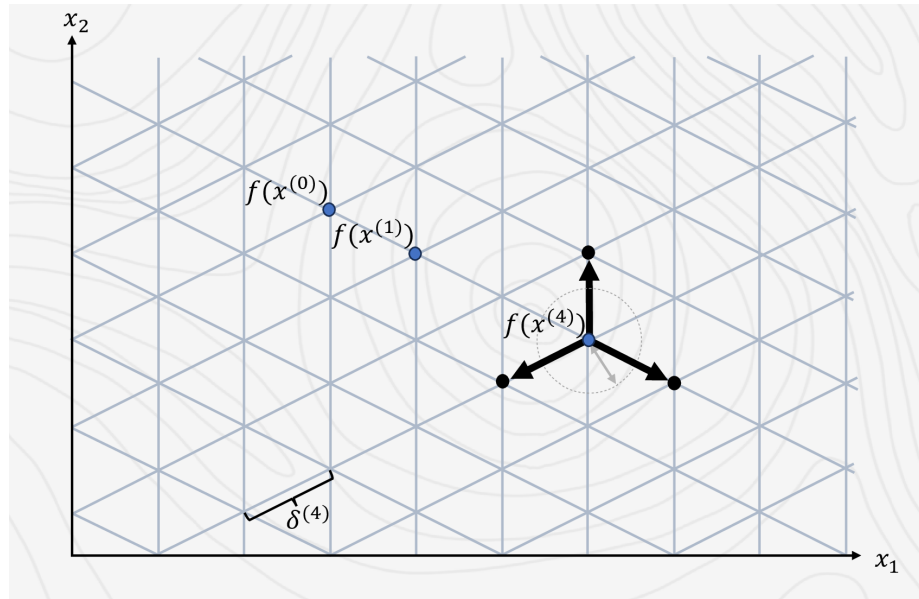


(c) GPS algorithm. Iteration $k = 2$. Mesh is coarsened. Unsuccessful iteration. This iteration demonstrates how the mesh coarsening feature could potentially release the search from a local minimum.

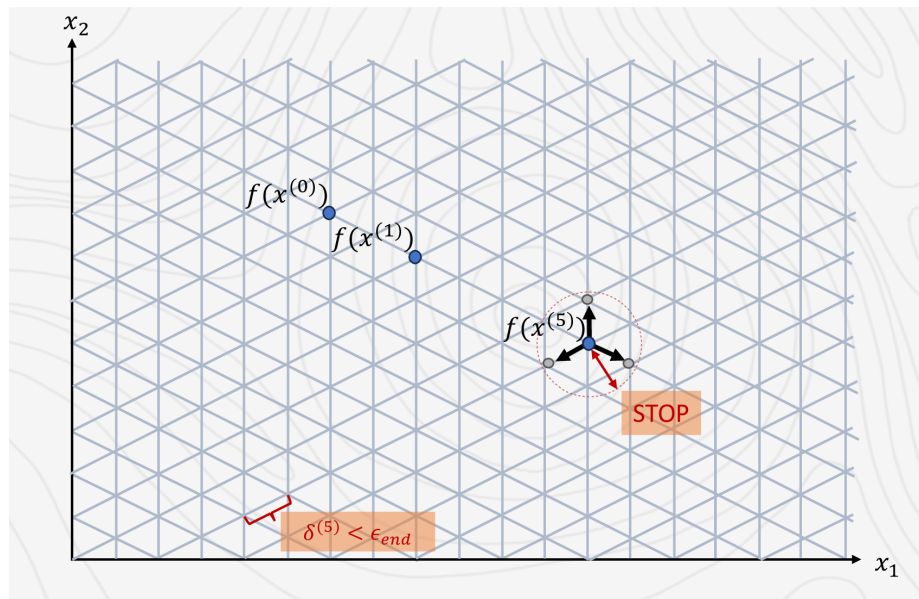


(d) GPS algorithm. Iteration $k = 3$. Mesh is refined. Unsuccessful iteration.

Figure 4: **Visualization of GPS algorithm iterations...continued and finalized.**



(e) GPS algorithm. Iteration $k = 4$. Mesh is refined. Unsuccessful iteration.



(f) GPS algorithm. Iteration $k = 5$. Mesh is refined. $\delta^{(5)}$ falls below the threshold ϵ_{end} , thus the algorithm is terminated.

search region using the frame size parameter $\Delta^{(k)}$. Importantly, when $\delta^{(k)}$ is refined at a faster rate than $\Delta^{(k)}$, MADS generates an asymptotically dense set of polling directions [20]. This is implemented in the algorithm 3 as $\delta^{(k)} = \min\{\Delta^{(k)}, (\Delta^{(k)})^2\}$. Figure 5 visualizes some MADS iterations.

Algorithm 3 : Mesh Adaptive Direct Search (MADS)

Given objective $f : \mathcal{S} \subset \mathbb{R}^n \mapsto \mathbb{R}$, and an initial point at $x^{(0)} \in \mathcal{S}$,

initialize

$\Delta^{(0)} \in (0, \infty)$ ▷ frame size
 $D = GZ$ ▷ positive spanning matrix
 $\tau \in (0, 1) \cap \mathbb{Q}$ ▷ mesh size adjustment
 $\epsilon_{end} \in [0, \infty)$ ▷ tolerance for termination
 $k \leftarrow 0$ ▷ iteration counter

repeat

1. MESH UPDATE

$\delta^{(k)} = \min\{\Delta^{(k)}, (\Delta^{(k)})^2\}$

2. SEARCH STEP

if $f_S(t) < f_S(x^{(k)})$ for some $t \in S^{(k)} \subset M^{(k)}$ **then**

$x^{(k+1)} \leftarrow t$

$\Delta^{(k+1)} \leftarrow \frac{\Delta^{(k)}}{\tau}$...and go to 4

else

3. POLL STEP

select p.s.s. $\mathbb{D}_\Delta^{(k)}$ such that $P^{(k)} = \{x^{(k)} + \delta^{(k)}d : d \in \mathbb{D}_\Delta^{(k)}\} \subset F^{(k)}$

if $f_S(t) < f_S(x^{(k)})$ for some $t \in P^{(k)}$ **then**

$x^{(k+1)} \leftarrow t$

$\Delta^{(k+1)} \leftarrow \frac{\Delta^{(k)}}{\tau}$

else

▷ else $x^{(k)}$ is a local minimizer

$x^{(k+1)} \leftarrow x^{(k)}$

$\Delta^{(k+1)} \leftarrow \tau \Delta^{(k)}$

end if

end if

$k \leftarrow k + 1$

until $\Delta^{(k)} < \epsilon_{end}$

Refer to GPS Algorithm 2 for example values of initialized parameters.

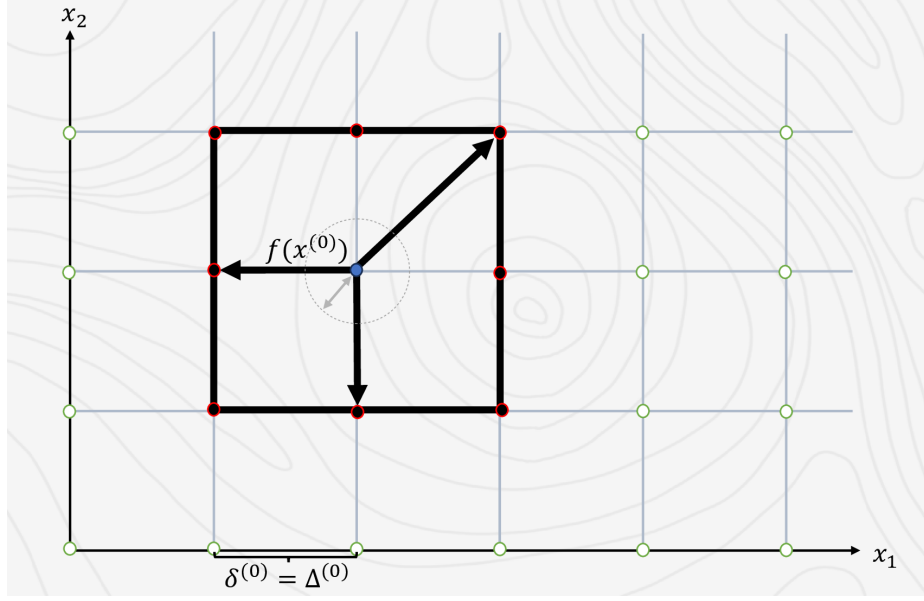
In algorithm 3 we define:

$$f_{\mathcal{S}}(x) := \begin{cases} f(x) & , x \in \mathcal{S} \\ \infty & , \text{otherwise.} \end{cases}$$

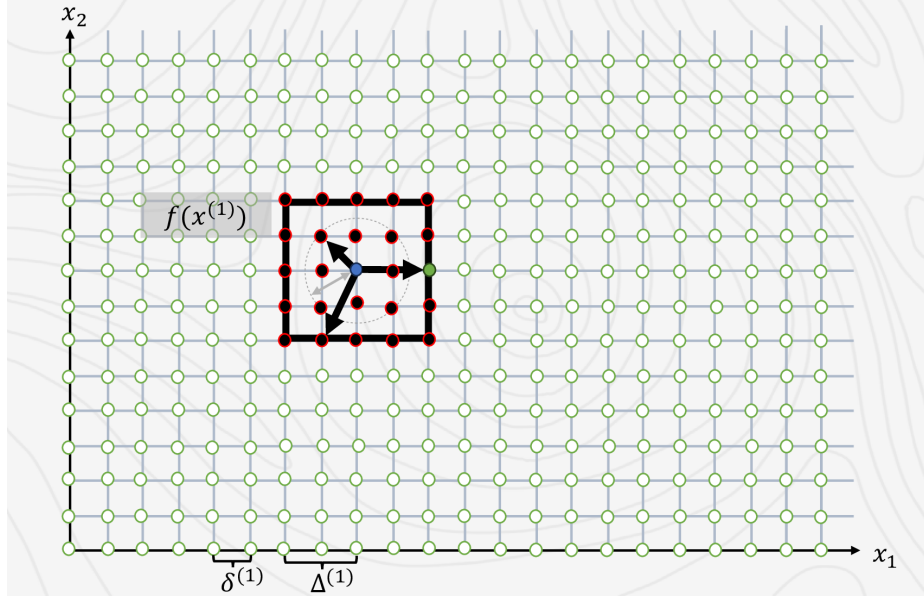
It is important to use $f_{\mathcal{S}}$ as it will restrict the optimization to the feasible set of points.

In our benchmarking work, we implement the more efficient OrthoMADS algorithm [34]. Its main differentiating feature being that it restricts the search directions to orthogonal ones and the choice of directions is made deterministically, allowing for reproducibility of results. This provides a stable environment for running the optimization. Alternatively, having to write the code from scratch may introduce unknowable errors, which could have immeasurable consequences given that the results in this thesis are aimed at being used on a real space mission. Reliability must be prioritized.

Figure 5: Visualization of MADS algorithm iterations.

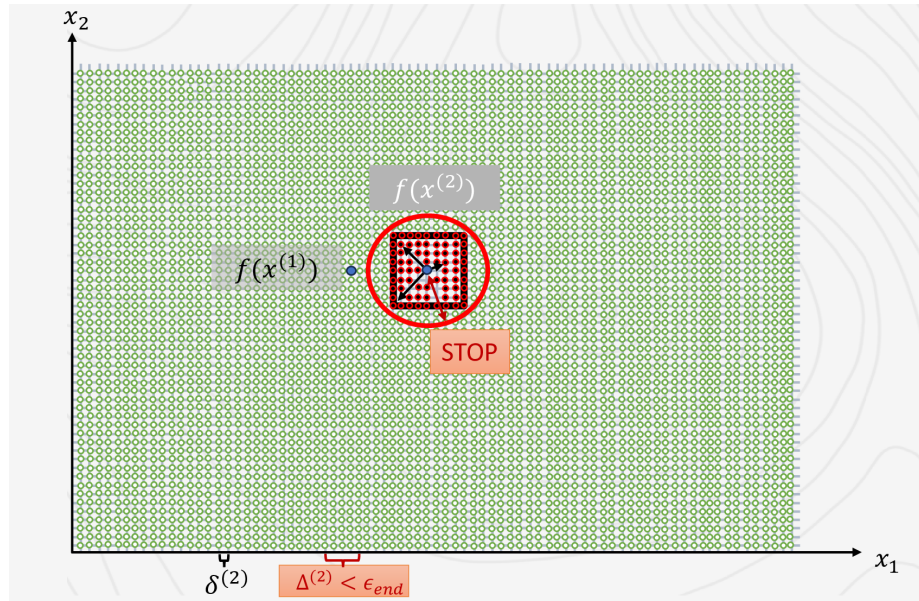


(a) MADS algorithm. Iteration $k = 0$. The white points circled in green represent the mesh points, and the black points circled in red represent frame points. As part of the initialization, we run the latin hypercube sampling in the Search Step which is implemented to find an appropriate starting point, $x^{(0)}$. The Search Step is not re-visited in this example. Unsuccessful iteration.



(b) MADS algorithm. Iteration $k = 1$. The mesh is refined and new search directions are chosen. Successful iteration.

Figure 5: **Visualization of MADS algorithm iterations...continued and finalized.**



(c) MADS algorithm. Iteration $k = 2$. Mesh is refined and incumbent x value moved to new point. $\Delta^{(2)}$ falls below the threshold ϵ_{end} , thus the algorithm is terminated.

2.3.2 Model-based Trust Region

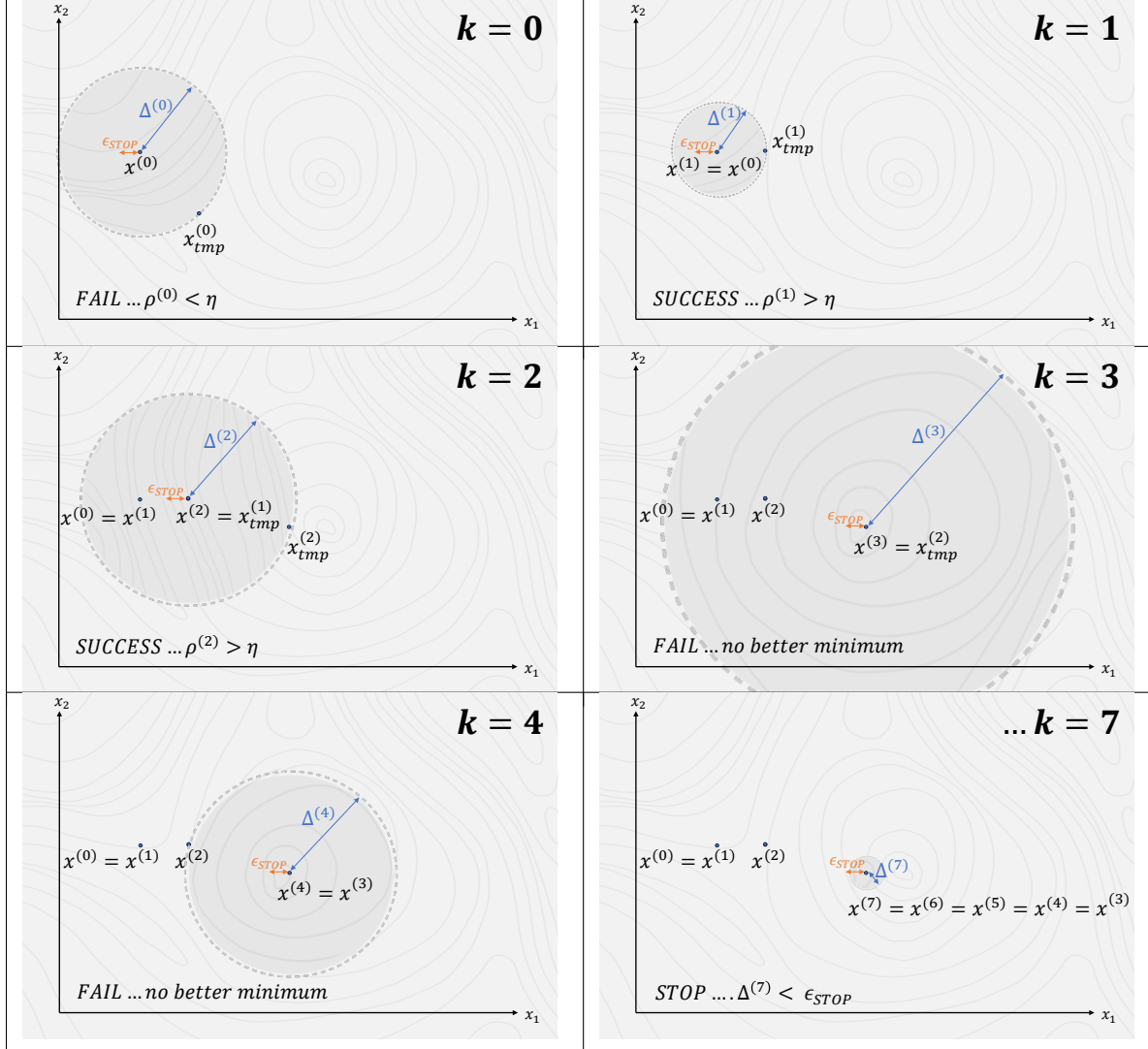
The second type of algorithm that we explore is the model-based trust region method (MBTR). If search-based methods (like MADS) iteratively step through the domain towards the solution, model-based methods rely on constructing local linear or quadratic models of the objective function as it seeks the solution. Figure 6 provides a visual of the iterations of the MBTR algorithm to aid the reader’s intuition as we step into a theory-intensive section. Note that this figure will be most clear after having completed reading of this MBTR section. Before revealing the MBTR algorithm, we define some terms to quantify what we mean by “local model”.

Models of multi-variate functions are categorized roughly into linear versus quadratic. There are various sub-types within each of these categories [10, 17]. For the sake of simplicity, in this text we restrict our view to fully linear models generated by linear interpolation, and quadratic interpolation.

In order to make models of the objective that converge to the objective function locally, we must be able to adjust/control the acceptable level of the model error - driving it to zero [35]. For the remainder of text, Δ will represent the user’s error tolerance in the model.

To that end, below we define *Fully Linear Models* as models which allow for user-controllable accuracy.

Figure 6: Visualization of MBTR algorithm iterations.



Definition 2.6: Class of Fully Linear Models

Given $f \in \mathcal{C}^1$, $x \in \mathbb{R}^n$, and $\bar{\Delta} > 0$, we say that

$\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$ is a *class of fully linear models* of f at x parameterized by Δ (abbreviated to FLM)

if there exists a pair of scalars $\kappa_f(x) > 0$ and $\kappa_g(x) > 0$ such that, given any $\Delta \in (0, \bar{\Delta}]$ the model $\tilde{f}_\Delta$ satisfies:

$$|f(y) - \tilde{f}_\Delta(y)| \leq \kappa_f \Delta^2 \quad \forall y \in \mathbb{B}_\Delta(x)$$

and

$$\|\nabla f(y) - \tilde{g}_\Delta(y)\| \leq \kappa_g \Delta \quad \forall y \in \mathbb{B}_\Delta(x)$$

where $\tilde{g}_\Delta = \nabla \tilde{f}_\Delta$.

In definition 2.6, the constants κ_f and κ_g are unknown to the user. The idea behind a class of fully linear models is that while the user may not know the exact level of error the model contains, they have access to the parameter Δ which can be used to drive the error to 0 in a predictable way. As $\Delta \rightarrow 0$, the upper bound on the difference between f and $\tilde{f}$ (in other words the model error) becomes tighter; likewise for ∇f and $\nabla \tilde{f}$.

The following example is borrowed from Audet and Hare [10] – it is used to illustrate the idea of fully linear models by demonstrating a single-variable case.

Example - FLM

Let $f(x) = \sin(\pi x)$ and $\bar{x} = 1$.

Then, the first-order Taylor approximation of f at $\bar{x}$ is

$$T(y) = f(\bar{x}) + \frac{d}{dx}f(\bar{x})(x - \bar{x}) = -\pi(x - 1).$$

Next, fix $\Delta > 0$.

Then, given any $y \in [1 - \Delta, 1 + \Delta]$, according to Taylor's Theorem, the error in this model satisfies:

$$\begin{aligned} |f(y) - T(y)| &\leq \frac{1}{2} \max_{\zeta \in [1-\Delta, 1+\Delta]} \left| \frac{d^2}{dx^2} f(\zeta) \right| |y - 1|^2 \\ &\leq \frac{1}{2} \max_{\zeta \in [1-\Delta, 1+\Delta]} |-\pi^2 \sin(\pi\zeta)| |y - 1|^2 \\ &\leq \frac{1}{2} \pi^2 \Delta^2, \end{aligned}$$

as $|y - 1| \leq \Delta$.

Next, we analyze the first-order error in this model.

$$\left| \frac{d}{dx} f(y) - \frac{d}{dx} T(y) \right| = |\pi \cos(\pi y) - (-\pi)| = |\pi \cos(\pi y) - \pi \cos(\pi)|.$$

Notice that for the function $g(x) = \pi \cos(\pi x)$, $\pi \cos(\pi)$ is the zeroth-order Taylor expansion of g at $\bar{x}$. As above, the error in this model satisfies:

$$\begin{aligned} |\pi \cos(\pi y) - \pi \cos(\pi)| &\leq \max_{\zeta \in [1-\Delta, 1+\Delta]} \left| \frac{d}{dx} g(\zeta) \right| |y - 1| \\ &\leq \max_{\zeta \in [1-\Delta, 1+\Delta]} |-\pi^2 \sin(\pi\zeta)| |y - 1| \\ &\leq \pi^2 \Delta. \end{aligned}$$

We can see that the model T is fully linear with constants $\kappa_f = \frac{1}{2}\pi^2$, and $\kappa_g = \pi^2$. This concludes the FLM example.

Next, we examine the two methods of generating FLMs which are implemented in the MBTR algorithms we compare in section 3.3:

- Linear Interpolation – constructs a linear function in $\mathbb{R}^n$ that passes through $n + 1$ points [36] – used by the COBYLA algorithm introduced in section 3.2.4.

- Quadratic Interpolation – constructs a quadratic function when there are more than $n + 1$ points for interpolation [37, 38] – used by the LINCOA algorithm introduced in section 3.2.4.

Linear Interpolation

A *linear* model of a function f can be expressed as

$$\tilde{f}_\Delta(x) = L(x) = \alpha_0 + \alpha^\top x,$$

where $\alpha_0 \in \mathbb{R}$, $\alpha = [\alpha_1, \dots, \alpha_n]^\top \in \mathbb{R}^n$.

Note: since we don't know the gradient $\nabla f(x)$, it is not possible to use Taylor polynomials.

Instead, consider the set of $m + 1$ interpolation points: $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$. One can think of interpolation points as the set of points in the domain where the objective function has been evaluated. Recall that in a blackbox optimization problem, we seek to limit as much as possible the number of objective function evaluations. Hence, working with a handful of “interpolation points” to locally model the objective is a realistic scenario.

A linear interpolation model can be built by finding values for α_0 and α that satisfy

$$L(y^i) = f(y^i) \quad \forall y^i \in \mathbb{Y} \dots$$

$$\Longleftrightarrow$$

$$L(y^i) = \alpha_0 + \alpha_1 y_1^i + \dots + \alpha_n y_n^i = f(y^i) \quad \forall y^i \in \mathbb{Y}. \quad (3)$$

In such a way, a system of linear equations with $n + 1$ unknowns and $m + 1$ equations is formed. For simplicity's sake, let's focus on the case where $n = m$. Then, equation 3 becomes:

$$\begin{bmatrix} 1 & (y^0)^\top \\ 1 & (y^1)^\top \\ \vdots & \vdots \\ 1 & (y^n)^\top \end{bmatrix} \cdot \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^n) \end{bmatrix}.$$

Introducing some new notation, we have:

$$f(\mathbb{Y}) := \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^n) \end{bmatrix} \in \mathbb{R}^n,$$

$$Y := [y^0 \ y^1 \ \dots \ y^n] \in \mathbb{R}^{n \times (n+1)}.$$

Again, we can re-express equation 3, this time using the new notation:

$$[\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y}),$$

where $\mathbf{1} \in \mathbb{R}^{n+1}$ is a vector of all ones. We seek a unique solution, thus $[\mathbf{1} \ Y^\top]$ must necessarily be non-singular. Definition 2.7 summarizes the linear interpolation function.

Definition 2.7: Linear Interpolation Function

Let $\mathbb{Y} = \{y^0, \dots, y^n\} \subset \mathbb{R}^n$ be such that $[\mathbf{1} \ Y^\top]$ is non-singular and $f : \mathbb{R}^n \mapsto \mathbb{R}$. Then the *linear interpolation function* of f over $\mathbb{Y}$ is

$$L_Y(x) := \alpha_0 + \alpha^\top x$$

where (α_0, α) is the unique solution to $[\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y})$.

Claim 1: The gradient of the linear interpolation function also forms a class of fully linear models.

Claim 2: Formally, the gradient of the linear interpolation function is called the *Simplex Gradient* for the simple reason that if $\mathbb{Y}$ is non-singular, then it necessarily also forms a simplex. The definition is fairly straight forward and follows directly from definition 2.7:

$$\nabla L_Y(x) = \alpha.$$

Both claims are proven in [10].

In the case of the linear interpolation model, the parameter Δ is used to control accuracy using what is called the *approximate simplex diameter*, denoted $\overline{diam}(\mathbb{Y})$. And since $\mathbb{Y}$ forms a simplex, we can set:

$$\Delta := \overline{diam}(\mathbb{Y}) = \max\{\|y^1 - y^0\|, \|y^2 - y^0\|, \dots, \|y^n - y^0\|\}.$$

We finish the linear interpolation section by expressing how the model is indeed fully linear (recall definition 2.6). Given $x \in \mathbb{R}^n$, there exists a bound $\bar{\Delta} > 0$, and a pair of scalars κ_f and $\kappa_g > 0$, such that if $\Delta := \overline{diam}(\mathbb{Y}) \in (0, \bar{\Delta}]$, then the linear interpolation model satisfies $\forall y \in B_\Delta(x)$:

$$|f(y) - L_Y(y)| \leq \kappa_f \Delta^2, \text{ and}$$

$$\|\nabla f(x) - \alpha\| \leq \kappa_g \Delta.$$

Quadratic Interpolation

Similarly to linear interpolation, in the case of the quadratic interpolation model, we are relying on a set of interpolation points to create a local model of the objective function, except this time, the model function has the form

$$Q_Y(x) = \alpha_0 + \alpha^\top x + \frac{1}{2} x^\top H x,$$

such that $Q(y^i) = f(y^i) \quad \forall y^i \in \mathbb{Y} = \{y^0, \dots, y^m\}$. Here, $H \in \mathbb{R}^{n \times n}$ is symmetric ($H = H^\top$), and the points y^i cannot all lie on a level surface of a quadratic function. Definition 2.8 formally summarizes this model.

Definition 2.8: Quadratic Interpolation Function

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, and $\mathbb{Y} = \{y^0, \dots, y^m\} \subset \mathbb{R}^n$ with $m = \frac{1}{2}(n+1)(n+2) - 1$. If the system

$$\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top H y^i = 0, i = 1, \dots, m$$

has a unique solution for α_0, α and $H = H^\top$, then the quadratic interpolation function of f over $\mathbb{Y}$ is

$$Q_Y(x) := \alpha_0 + \alpha^\top x + \frac{1}{2}x^\top H x,$$

where $(\alpha_0, \alpha, H = H^\top)$ is the unique solution to

$$\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top H y^i = f(y^i), i = 0, \dots, m.$$

With the minutia of generating local (linear and quadratic) models covered, we turn our attention to the Model-based Trust Region blackbox optimization algorithm. Algorithm 4 provides the pseudo-code for this method, and figure 7 consolidates the decision making process in a visual flow chart. Note, in algorithm 4, it is not indicated whether $\tilde{f}$ is found via linear or quadratic (or other [25, 39]) modeling means. That is because any one of these modeling methods can be used. Furthermore, after reviewing the pseudo-code, the reader is encouraged to revisit the earlier figure 6 graphs to solidify their intuition of the MBTR method. We proceed to describe the steps in the algorithm.

Algorithm 4 : Model-Based Trust Region (MBTR)

Given $f \in C^1$, an initial point at $x^{(0)}$, and fully linear models $\tilde{f}_\Delta$ of f

initialize

$\Delta^{(0)} \in (0, \infty)$ ▷ frame size

$\tilde{f}^{(0)}$ ▷ starting model

μ ▷ model accuracy

$\eta \in (0, 1)$ ▷ decrease test

$\gamma \in (0, 1)$ ▷ trust region update

$\epsilon_{end} \in [0, \infty)$ ▷ tolerance for termination

$k \leftarrow 0$ ▷ iteration counter

continued...

...algorithm 4 continued

repeat

1. MODEL

if $\Delta^{(k)} \leq \mu \|\nabla \tilde{f}^{(k)}(x^{(k)})\|$ **and** $\tilde{f}^{(k)}$ is an F.L.M. of f on $\mathbb{B}_{\Delta^{(k)}}(x^{(k)})$ **then**

2. TRUST REGION

obtain $x_{\text{temp}}^{(k)} \in \arg \min_x \{\tilde{f}^{(k)}(x) : x \in \mathbb{B}_{\Delta^{(k)}}(x^{(k)})\}$

3. CANDIDATE CHECK

evaluate $f(x_{\text{temp}}^{(k)})$ and compute $\rho^{(k)} = \frac{f(x^{(k)}) - f(x_{\text{temp}}^{(k)})}{\tilde{f}^{(k)}(x^{(k)}) - \tilde{f}^{(k)}(x_{\text{temp}}^{(k)})}$

if $\rho^{(k)} > \eta$, sufficient decrease found, **success then**

let $x^{(k+1)}$ be such that $f(x^{(k+1)}) \leq f(x_{\text{temp}}^{(k)})$

$\Delta^{(k+1)} \leftarrow \gamma^{-1} \Delta^{(k)}$

generate $\tilde{f}^{(k+1)}$ from $\tilde{f}^{(k)}$ and $x^{(k+1)}$

else failure

$x^{(k+1)} \leftarrow x^{(k)}$

$\Delta^{(k+1)} \leftarrow \gamma \Delta^{(k)}$

$\tilde{f}^{(k+1)} \leftarrow \tilde{f}^{(k)}$

end if

else

$\Delta^{(k+1)} \leftarrow \gamma \Delta^{(k)}$ and $x^{(k+1)} \leftarrow x^{(k)}$

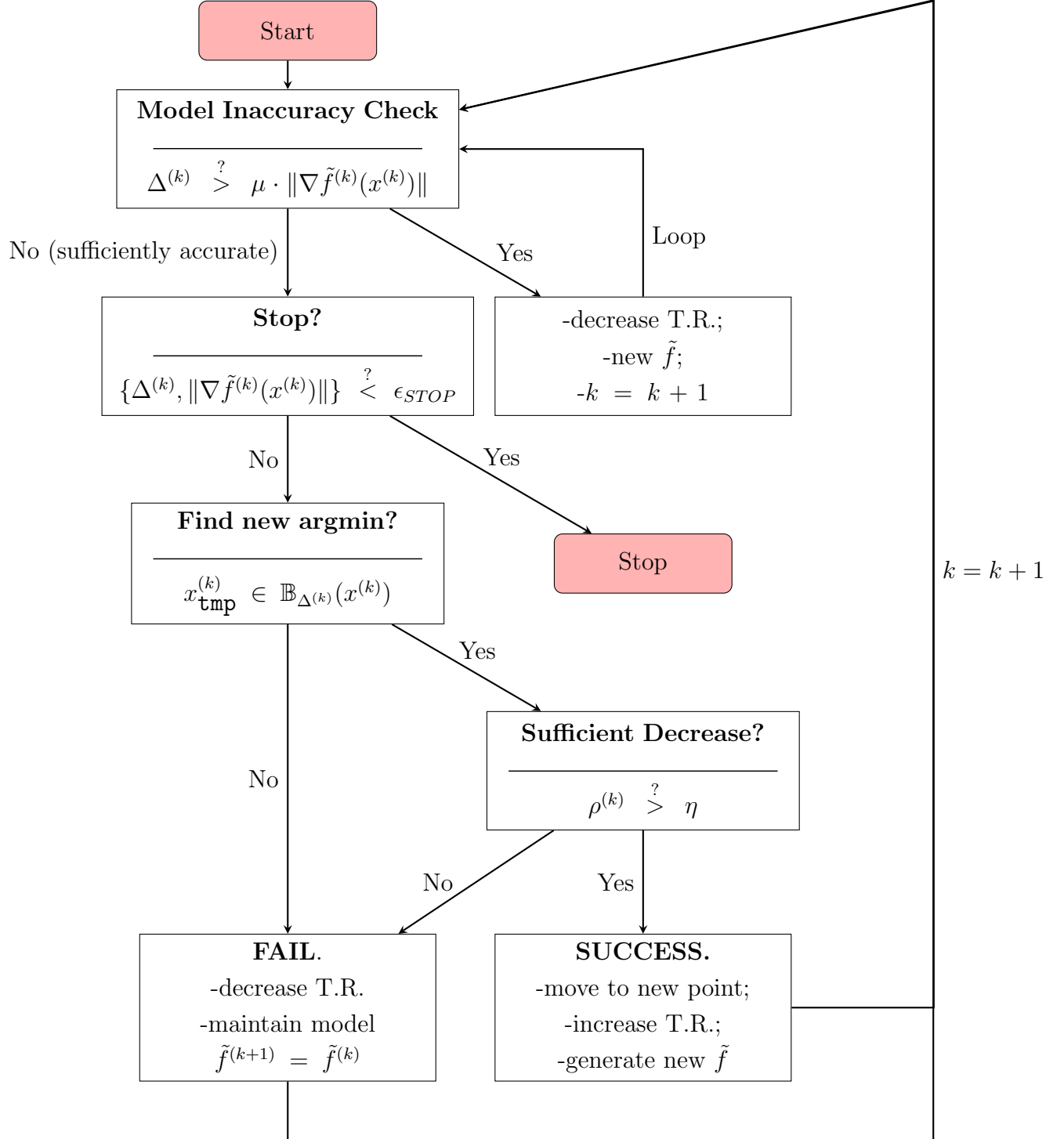
use $\tilde{f}_\Delta$ to generate F.L.M. $\tilde{f}^{(k+1)}$ of f on $\mathbb{B}_{\Delta^{(k+1)}}(x^{(k)})$

end if

$k \leftarrow k + 1$

until $\Delta^{(k)} < \epsilon_{\text{end}}$ **and** $\|\nabla \tilde{f}^{(k)}(x^{(k)})\| < \epsilon_{\text{end}}$

Figure 7: MBTR Pseudo-algorithm as a flowchart schematic.



Model Step

The MBTR algorithm takes fully linear models $\tilde{f}_\Delta$ of f as inputs. I.e., for any $\Delta > 0$ and for any x , the algorithm uses approximations of f and ∇f in a Δ -ball about x . Next, the algorithm seeks to validate the accuracy of the model. Meaning, it checks if the amount of deviation of the model from the true objective function within the Δ -ball is greater than the tolerance. If the model passes the accuracy check, then it is re-named $\tilde{f}^{(k)}$ to refer to the specific model at iteration k .

Trust region sub-problem

Once the model passes the accuracy check, it is used to analytically solve for the minimizer within the trust region, $\mathbb{B}_{\Delta^{(k)}}(x^{(k)})$,

$$x_{\text{tmp}}^{(k)} \in \arg \min_x \left\{ \tilde{f}^{(k)}(x) \quad : \quad x \in \mathbb{B}_{\Delta^{(k)}}(x^{(k)}) \right\}.$$

Candidate accuracy assessment

The candidate solution, $x_{\text{tmp}}^{(k)}$, is also checked for accuracy similarly to the earlier model for this iteration. The following ratio, denoted $\rho^{(k)}$, defines the accuracy:

$$\rho^{(k)} := \frac{f(x^{(k)}) - f(x_{\text{tmp}}^{(k)})}{\tilde{f}^{(k)}(x^{(k)}) - \tilde{f}^{(k)}(x_{\text{tmp}}^{(k)})}.$$

The user defines $\eta \in (0, 1)$ as the “sufficient decrease test parameter”. The ratio, $\rho^{(k)}$, compares actual reduction of the objective, to the reduction that the model estimates. Small values ($\rho^{(k)} \leq \eta$) can be interpreted as the objective function not having decreased as much as the the model estimated. In which case, the model needs improving and the candidate point is rejected and the trust region reduced.

Alternatively, had the candidate point passed the accuracy check, we would move the model to be centered at the new candidate point, and the trust region radius would be expanded. This ability to expand the trust region makes the MBTR algorithm dynamic since it has the ability to break free from local minimizers; similar to the MADS algorithm in section 2.3.1.

Note, setting η to a small value, e.g., 0.05, may allow more points to pass the candidate accuracy check, but the model may not approximate the local decrease as well as if η was set to a higher value like 0.5.

Stopping condition

Eventually, the algorithm is considered to have ended successfully if the radius of the trust region, $\Delta^{(k)}$ and the gradient of the model, $\nabla \tilde{f}^{(k)}$, have both fallen within the user-defined ϵ_{STOP} threshold.

2.3.3 Convergence

MADS

The convergence analysis for MADS is in two parts:

1. showing that the mesh gets infinitely fine:

$$\liminf_{k \rightarrow \infty} \delta^{(k)} = 0 \quad \text{and} \quad \liminf_{k \rightarrow \infty} \Delta^{(k)} = 0, \text{ and}$$

2. for any refined point, generalized directional derivatives at are non-negative.

This is discussed in detail in Chapter 8 of Ref. [10]. Generally speaking, MADS has a variable convergence rate which is dependent on the dimensionality of the search space, smoothness of the objective function, noise in the objective function, search space constraints (especially non-linear ones), among others; for further details see Ref. [20].

MBTR

Per Chapter 11 of Ref. [10], the convergence analysis for MBTR establishes that:

1. if $\nabla f(x^{(k)}) \neq 0$, the algorithm eventually produces a successful iterate, and
2. if $x^{(k)}$ remains constant after some finite number of iterations, then it must be that $\nabla f(x^{(k)}) = 0$, and
3. after an infinite series of *successful* iterations,

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^{(k)})\| = 0 \quad \Longleftrightarrow \quad \|\nabla f(x^{(k)})\| \rightarrow 0.$$

Similarly to MADS, the rate of convergence of MBTR is variable, but it depends on the method used in the step when the trust-region constraint is inactive. Specifically, MBTR has shown potential for quadratic convergence per Ref. [40] when implementing Newton's method.

2.3.4 Bi-objective Optimization

Often times, optimization problems wish to solve for two objectives simultaneously. Recall grade-school problems when students are tasked with maximizing the volume of a box while minimizing the cost of materials: expensive glass top, cheaper wood sides. This problem presents two objectives – volume and cost – and two arguments – area of wood material and area of glass materials. In future works, we may need to optimize for more than one objective simultaneously, e.g., maximizing key rate while minimizing the quantum bit error rate. More on this in section 3.2.

A bi-objective problem can be formally defined as:

$$\min_{\mathbf{x} \in \mathcal{S}} F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \tag{4}$$

where $F : \mathcal{S} \rightarrow \mathbb{R}^2$, and the feasible region is defined to be $\mathcal{S} \subseteq \mathbb{R}^n$.

Going back to the single-objective case, to compare two solutions, say (a_1, a_2) vs. (b_1, b_2) , it was sufficient to observe their respective objective values, e.g., $f(a_1, a_2) \leq f(b_1, b_2)$. However, in the bi-objective case, this is treated somewhat differently. We introduce the more general concept of *dominated points* in definition 2.9 to allow for comparing solutions in the case of two objective functions. This will be used in explaining how the bi-objective solution is ultimately obtained.

Definition 2.9: Dominated Points

Let $\mathbf{a}, \mathbf{b} \in \mathcal{S}$ be two feasible points. Then,

- $\mathbf{a}$ is said to *dominate* $\mathbf{b}$, denoted $\mathbf{a} \prec \mathbf{b}$, if and only if

$$f_1(\mathbf{a}) \leq f_1(\mathbf{b}), \quad f_2(\mathbf{a}) \leq f_2(\mathbf{b}) \quad , \text{ and}$$

$$f_i(\mathbf{a}) < f_i(\mathbf{b}) \quad \text{for at least one index } i \in \{1, 2\}.$$

- $\mathbf{a}$ is called *indifferent* to $\mathbf{b}$, denoted $\mathbf{a} \sim \mathbf{b}$, if and only if, $\mathbf{a}$ does not dominate $\mathbf{b}$, and vice versa, $\mathbf{b}$ does not dominate $\mathbf{a}$.

The simplest strategy for addressing a bi-objective problem given a blackbox, is to plot one objective against the other for all sampled points in the domain. Commonly, this set of points plots out a curve, representing a pattern of trade-offs between the two objective functions. This curve is referred to as a “Pareto-optimal frontier”, and in more recent publications is simply referred to as “Pareto front” [10, 41, 42]. The hope is that in the end, there will be a clear solution for an optimal combination of objective values, though all too often this is not the case.

Definition 2.10: Pareto-optimality

A feasible *point* $\mathbf{a} \in \mathcal{S}$ is said to be *Pareto-optimal* if and only if there is no $\mathbf{b} \in \mathcal{S}$ such that $\mathbf{b} \prec \mathbf{a}$.

We call the set of Pareto-optimal points (solutions) the *Pareto set*. It is denoted $\mathcal{S}_{\mathcal{P}}$, and it is the solution to equation 4.

The *Pareto-optimal frontier* is defined to be the set of values $F(\mathcal{S}_{\mathcal{P}})$, and is denoted $F_{\mathcal{P}} \subseteq \mathbb{R}^n$.

When a solution does not clearly stand out along the Pareto-optimal frontier, one strategy is to define and optimize a third function: a weighted linear combination of both objectives,

$$B_{\lambda} = \min_{\mathbf{x} \in \mathcal{S}} \lambda f_1(x) + (1 - \lambda)f_2(x). \quad (5)$$

Here, $\lambda \in [0, 1]$. To solve for B_{λ} , we vary λ across the interval $[0, 1]$. This method doesn't always solve the problem and more sophisticated methods can be employed [43].

2.3.5 Benchmarking

The *No free lunch Theorems*, originally described by Wolpert and Macready in 1997 prove that there is no single best algorithm performance-wise across all possible optimization problems, [44]. It is suggested that an improvement to an algorithm as it pertains to one set of problems, necessarily will make its performance worse for some other set of problems. As such, in this thesis we restrict our attention to comparing algorithm performance in the specific context of a *continuous multi-objective blackbox optimization problem*.

Continuous because our parameters of interest (detector voltage and mean photon number) and objective values (key rate, and quantum bit error rate, aka QBER) can take any real value as sampled from within a specified interval.

Multi-objective because we observe two different objectives: key rate and QBER.

And *blackbox* because the objective function is concealed by the QEYSSat simulator. In other words, there is no analytic expression for the transformation of parameter inputs to objective outputs.

In general, the exercise of benchmarking algorithms seeks to compare their *efficiency* and *accuracy*. In the setting of optimization problems, efficiency can take the form of run time complexity (CPU time) and/or computational complexity (number of fundamental function evaluations). In the BBO setting, algorithm efficiency focuses on comparing the number of required objective function evaluations, or *f-calls*, because that is consistently and by far the most time-intensive and computationally-intensive part of the process [45, 46]. Accuracy is measured as the *distance from the starting point to the final iteration* **relative to** the *distance from the starting point to the best known optimal value*. i.e., how far did you get in N iterations compared to how far you could have gone in unlimited iterations.

$$\text{efficiency} := \text{number of f-calls}$$

$$\text{accuracy} = f_{\text{acc}}^N := \frac{f(x^{(N)}) - f(x^{(0)})}{f(x^*) - f(x^{(0)})} \in [0, 1] \quad (6)$$

Here, $x^{(0)}$ and $x^{(N)}$ refer to the inputs on the initial f-call and the final, N^{th} iteration respectively. Figure 8 provides some intuition for this.

The procedure for any benchmarking task is in 3 parts:

1. selecting a data set
2. observing algorithm performance
3. analyzing results

Expanding on how we follow this procedure in order.

Prior to running the algorithms to obtain the data, we decide on a set of starting points for the algorithms to start running – all algorithms in the benchmarking exercise will run starting at these points. The data set is collected by running each of our chosen optimization algorithms over the QEYSSat simulator as many times as there are starting points. Every time an optimization is run, it iteratively chooses a sequence of steps in the domain (a path), and with each step evaluates the objective function and hopefully gets closer to the optimal value.

We denote by $\mathcal{A}$ the set of chosen algorithms for benchmarking, and let $a \in \mathcal{A}$ be an algorithm in that set. Let $\mathcal{P}$ be the set of starting points fed into an optimization algorithm – each of which should yield a path – with $p \in \mathcal{P}$ denoting a starting value in the set.

Note, we say an algorithm has “solved” an optimization if within N iterations, if $f(x^{(N)})$ attains a value that is within the user-defined interval of accuracy.

Next, to obtain algorithm performance information, we start by recording the basics – expected values for:

- time to reach solution
- solution accuracy and efficiency
- percentage of paths that reach solution for a fixed N .

Although insightful, these observables are not usually conclusive. So, we seek less obvious metrics for algorithm performance. Namely, we build *performance profiles* (read informative plots). The remainder of this section is dedicated to deriving an understanding of the performance profile.

Each performance profile corresponds to a fixed accuracy tolerance level, $\tau \in [0, 1]$. This value determines when to stop the algorithm. In other words, how close an iteration $f(x^{(N)})$ has to be to $f(x^*)$ before considering it close enough to stop.

To determine whether or not algorithm a , with path p , achieves a solution that falls within τ of perfect accuracy, i.e. $f(x^*)$, we define a convergence test, $T_{a,p}$.

$$T_{a,p} = \begin{cases} 1 & , f_{acc}^N \geq 1 - \tau \\ 0 & , \text{otherwise.} \end{cases}$$

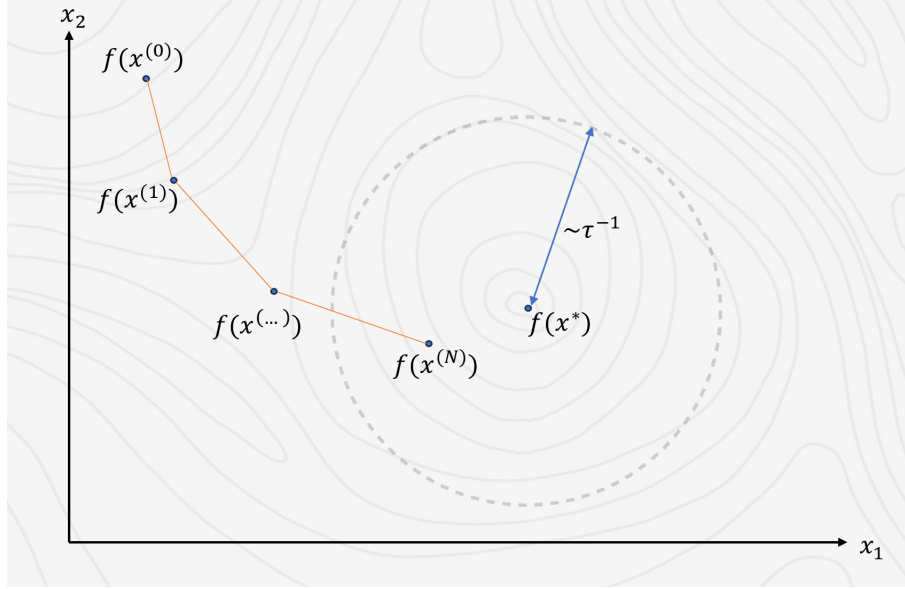
So, $\overbrace{\text{whether or not you get within } \tau}^{T_{a,p}}$ $\overbrace{\text{accuracy in } N \text{ iterations}}^{f_{acc}^N}$ will depend on your $\underbrace{\text{algorithm}}_a$ and respective $\underbrace{\text{starting point in the domain}}_p$.

Q: How do we tell which algorithms in $\mathcal{A}$ are best?

A: The best ones will consistently get the largest portion of p 's to within τ accuracy for a fixed N (budget of f-calls).

Importantly, the “best” algorithm can be different for different allowances of iterations, N . Thus, to benchmark the efficiency and robustness (fraction of solved problems) of a set of optimization algorithms, we would like to compare the percentage of paths that arrive at a solution for a fixed budget of f-calls. Then, varying N ,

Figure 8: Example of a contour plot for a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, and a possible path (in orange) generated by iterations of an optimization algorithm. Although inaccurate, for intuition, one may think of τ as related to the radius of a ball about $f(x^*)$ which defines the region of acceptable accuracy for a solution.



observe if the relative percentages of the different algorithms change – this is what we try to represent in a performance profile.

We are now ready to formally define the performance profile. [10, 47].

Definition 2.11: Performance Profile

A performance profile is a mapping:

$$\rho_a(M) = \frac{1}{|\mathcal{P}|} \cdot |\{p \in \mathcal{P} : N_{a,p} \leq M\}|,$$

where, $|\cdot|$ represents the cardinality of the set, and $N_{a,p}$ denotes the earliest iteration such that $f(x^{(N_{a,p})})$ is within the required tolerance.

Figure 9 is an example of two performance profiles resulting from the research in this thesis – explored in more detail in section 3.3.

Next, we propose a re-parameterization of the variable on the x -axis as a *ratio* dependent $N_{a,p}$:

$$r_{a,p} = \begin{cases} \frac{N_{a,p}}{\min_{\tilde{a} \in \mathcal{A}} \{N_{\tilde{a},p} : T_{\tilde{a},p} = 1\}} & , f_{\text{acc}}^N \geq 1 - \tau \\ \infty & , \text{otherwise} \end{cases}$$

This ratio, $r_{a,p}$, is used to make a fair comparison of algorithm performance. One can think of this as a kind of normalization, where $r_{a,p}$ compares the number of steps it takes for an algorithm to arrive at a solution compared to the best case algorithm for that same path (starting point in the domain).

It can be observed that $r_{a,p} = 1$ corresponds to the algorithm which attains the minimum number of f-calls for a solution. Subsequently, $r_{a,p} = 2$ corresponds to an algorithm which needed twice as many f-calls as the best algorithm (for the same p). So, the higher $r_{a,p}$, the further you are from the best performing algorithm in terms of efficiency. Now we can adjust the definition of the performance profile, ρ_a as follows:

$$\rho_a(M) = \frac{1}{|\mathcal{P}|} \cdot |\{p \in \mathcal{P} : r_{a,p} \leq M\}|.$$

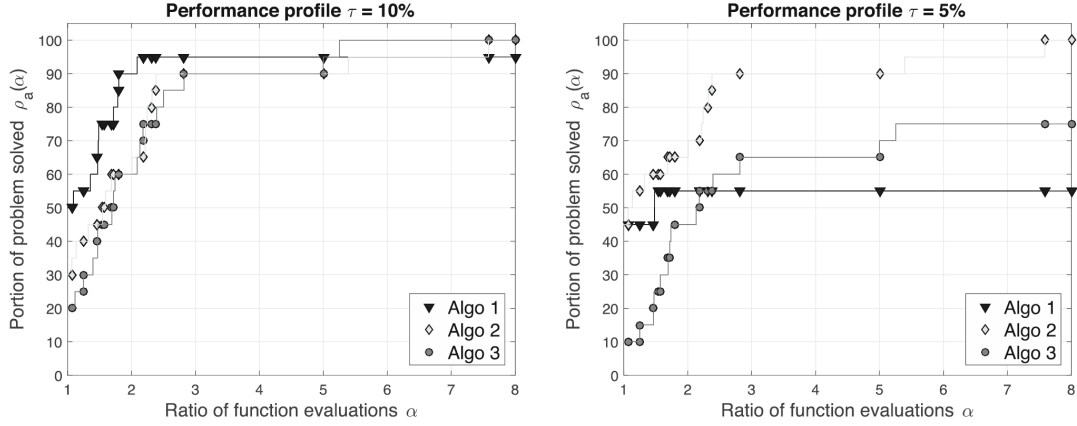


Figure 9: **Examples of a performance profiles with $\tau = 10\%$ and $\tau = 5\%$, comparing the efficiency and robustness of some algorithms a_1 , a_2 , and a_3 . Reprinted from [10].**

To summarize, we compare algorithms using performance profile plots. If we want the solution to be within τ of perfect accuracy, then different algorithms will get you there at different speeds ($N_{a,p}$ iterations or f-calls). This $N_{a,p}$ will also depend on where in the domain you start your optimization, point p . The higher the proportion of paths with $T_{a,p} = 1$, the better the algorithm is considered to perform. By varying the budget of f-calls, N , one may observe that some algorithms perform better than others. Constructing these profiles for different accuracy tolerances (τ) may reveal patterns in the relationships of the algorithms in question. For fair comparison of performance, algorithms are all compared against the best known algorithm. As such, the plot of the profile has on its x -axis, the relative number of f-calls, $r_{a,p}$ instead of the absolute $N_{a,p}$.

3 The Optimization Application

3.1 Overview

What is QKD?

Quantum Key Distribution (QKD), is a secure communication protocol. More specifically, it is one step in the chain of events required to send an encrypted message: the key distribution step. Supposing two distant parties wish to share an encrypted message, they will then need a **key** to encrypt and de-crypt the message to mask the contained information from potential interceptors. The key in our scenario is a string of 1s and 0s (bits), and the encrypted bits (qubits) are pulses of photons generated by a laser.

The security in the QKD protocol comes from its use of quantum phenomena, namely, entanglement and/or superposition (depending on the protocol). Moreover, the No Cloning Theorem [48, 49, 50] proves that unknown pure quantum states cannot be perfectly copied. The consequence of this theorem is that an eavesdropper cannot make copies of the encrypted message while it is in transit even if they wanted to.

Why single photons?

In the QKD field of study, much effort is spent predicting how to break the encryption protocol, and subsequently devising methods to thwart such attacks. Above, we mentioned that QKD takes advantage of quantum phenomena to guarantee protocol security – this involves physically working at the single particle level.

In practice, it is not (yet) possible to guarantee that a laser source will consistently generate single-photon pulses. Sometimes, it will inevitably emit multi-photon pulses. However, the specific QKD protocol implemented by the QEYSSat mission aims to enhance the protocol’s security by including an additional degree of freedom in the light pulse: its intensity (aka the number of photons in the pulse) [51, 52]. The sender and receiver monitor how many photons are sent and received in each pulse. If there are too many inconsistencies, they can assume there is an eavesdropper.

This intensity-enhanced transmission method is called Vacuum + Weak Decoy State [53, 54], and it is based on the original QKD protocol called BB84, developed by Bennett and Brassard in 1984 [55, 56].

3.1.1 Free-space QKD

Physical implementations of QKD have been around since the year 1989, when the authors of BB84 built a proof-of-principle experiment for their own protocol [57]. Since then, we have seen academic and industrial [58, 59] interests push fiber-based [60, 61, 62, 63] and free-space network architectures [64, 65] to the limits of their operational range. To date, fiber-based systems have shown functionality over 1000 km [66] (albeit not BB84), compared to the 144 km for ground-based atmospheric free-space systems [64]. In both implementations, operational distance is the fundamental barrier to building a secure global network. Possible solutions include setting up signal repeaters, analogous to those used in classical telecommunications. As yet, this technology is still not fully secure and requires further development [67]. Another, less direct solution is post-quantum cryptography (PQC), which is a set of algorithms that overcome the need for quantum hardware altogether. However, there is no guarantee that someday there won't be a better algorithm to undermine the existing PQC [68]. The quantum mechanical laws underpinning the security of QKD hardware may not be so easily bested [69]. So, we turn back to QKD, where a terrestrial channel is range-limited, but repeaters are not sufficiently mature.

The latest experimentally proven approach to expanding QKD globally is a satellite-to-ground free-space link. This satellite-to-ground approach is less affected by the transmission losses suffered by communications established strictly within the atmosphere [70]. The more atmosphere the photons have to pass through, the more likely the photon is to get absorbed by the atmosphere and not reach the receiving party. The first major proof-of-principle satellite experiment was the Micius operation launched in 2016 [71]. Importantly, their mission established a QKD link, as well as tested Bell's inequality, thus paving the way for broader satellite networks, but also,

alternative architectures [71, 72, 73, 74, 75]. Micius has an **uplink** and **downlink** configuration. A **downlink** means that the source and transmission equipment reside on the satellite, and the receiver components, on the ground. This is in contrast to an **uplink** configuration, where the source and transmission equipment are on the ground and receiver equipment, on the satellite. Although an uplink connection would suffer from comparatively higher signal losses due to atmospheric effects influencing the beam earlier in the transmission channel (called the shower curtain effect), an uplink has the particular advantage of housing the more cumbersome (transmitter) equipment on the ground and accessible to the science team. Given that satellite-based QKD is still in its developmental phase, having the ability to swap out key components to conduct varied experiments is an important capability at this stage [70]. Enter the Quantum Encryption and Science Satellite (QEYSSat) [76, 77].

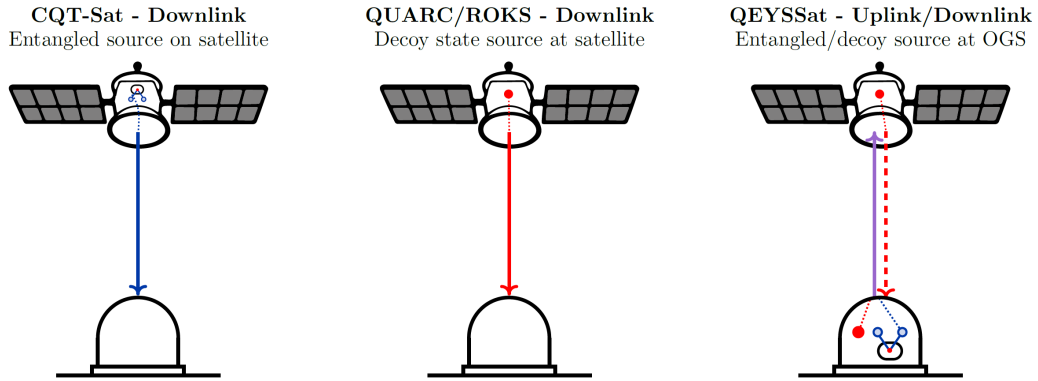
3.1.2 Satellite Link Simulation

The Quantum Encryption and Science Satellite (QEYSSat) is an experimental QKD mission funded by the government of Canada and set to launch in the year 2025 [77]. The original idea was to try an alternative and more experimentation-friendly architecture (uplink) compared to the ones being proposed by similar projects at the time (downlink) [71, 78, 79, 80, 81].

Since then, the Micius mission has launched and demonstrated the feasibility of a quantum connection from space to ground for both entanglement-based and prepare-and-measure QKD schemes. And the QEYSSat mission has since broadened its scope to include a downlink (RefQ project) [82] in addition to the originally proposed uplink and its three different sources – weak coherent pulse (WCP), spontaneous parametric down-conversion (SPDC), and quantum dot (QD). Figure 10 from [78] neatly lays out some upcoming missions and their respective schemes.

All these missions abound, there have been numerous efforts to model the expected link quality and its corresponding security, as well as to optimize certain parameters therein. Early works focused on estimating link losses. Notably, our own research

Figure 10: **Reprinted from [78]. Quantum channel configuration for three different SatQKD missions.** Each mission implements a different combination of QKD protocols and quantum channel configurations between an OGS and an orbiting satellite. The Singaporean CQT-Sat mission (left) implements the entanglement-based BBM92 protocol (blue arrow) in a downlink configuration. For this mission, one of the photon pairs is measured on board and the other is transmitted to the OGS. The UK QUARC/ROKS mission (middle) implements the weak coherent pulse (WCP) decoy-state BB84 protocol (red arrow) in a downlink configuration. The Canadian QEYSSat mission (right), implements both the decoy BB84 and BBM92 protocols (purple arrow) in an uplink configuration and intends to also incorporate a decoy BB84 downlink.



group who published results in 2013 modeling beam propagation and atmospheric effects for a QKD link [70]. The model combined simulated satellite pass geometries – distance and position as a function of time generated in the Ansys Systems Toolkit – with MODTRAN simulations of atmospheric effects, including diffraction, turbulence, absorption, and transmittance. In the end, the orbital path, and atmospheric effects were combined with a MATLAB simulation of beam divergence with distance. This 3-part simulation yielded a value for *transmissivity*, which is the probability that a photon sent by the transmitter would arrive at the receiver. Transmissivity, as such, can be used to calculate a key rate using a key estimation formula, and a corresponding error or noise probability. This work however did not allow for generating a bit sequence; that work came later.

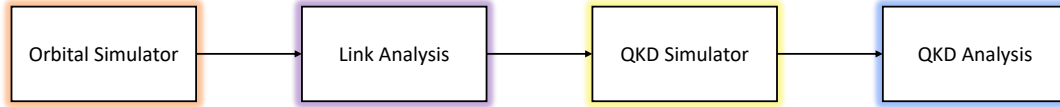
Subsequent efforts from other groups took the simulation a step further, discretizing and optimizing a numerical simulation of the satellite pass. In which case the simulation included many more parameters such sender/receiver basis bias, and a blackbox optimization for mean number of photons at the source [83]. Other groups yet, took a completely different approach. In some cases opting for a wholly analytic paradigm [84] and continuous optimization methods. In other cases, introducing machine learning algorithms to estimate optimal configurations for crucial components [85].

Our QEYSSat simulation, in its current state is a 4 part computer model, see Figure 11. It works by generating a bit string representing the key that the sender wishes to emit, and then adding or subtracting or otherwise altering the qubits according to functions modeling the various phenomena that affect the transmission process.

At a high level, the simulation operates as follows:

- Using the Ansys Systems Toolkit, an orbital satellite pass is simulated, recording the angle, position and distance with respect to the ground station.
- Then, the Link Analysis program, a MATLAB script, uses that geometry to compute coefficients for atmospheric effects corresponding to the specific pass.
- Third, the QEYSSat Simulator, a MATLAB program, generates a key (bit se-

Figure 11: **The QEYSSat simulation is organized in four parts. This figure depicts the flow of data between software parts. Reprinted from [86].**



quence) from the sender, and a key measured by the receiver, incorporating all errors and losses that are known to affect such a connection (adding/altering/removing bits), outputting sender and receiver time tag files.

- Lastly, the QKD Analysis software processes the two sender/receiver time tag files from the QEYSSat Simulator, and produces a final key rate as well as a variety of other outputs. This last software is written in C and C#.

The parameters over which we aim to optimize the link security are defined in the QKD Simulator, so we take a closer look at how that part of the overall simulation works.

There are four major modules in the QEYSSat Simulator: (1) the sender, (2) the eavesdropper, (3) the channel, and (4) the receiver.

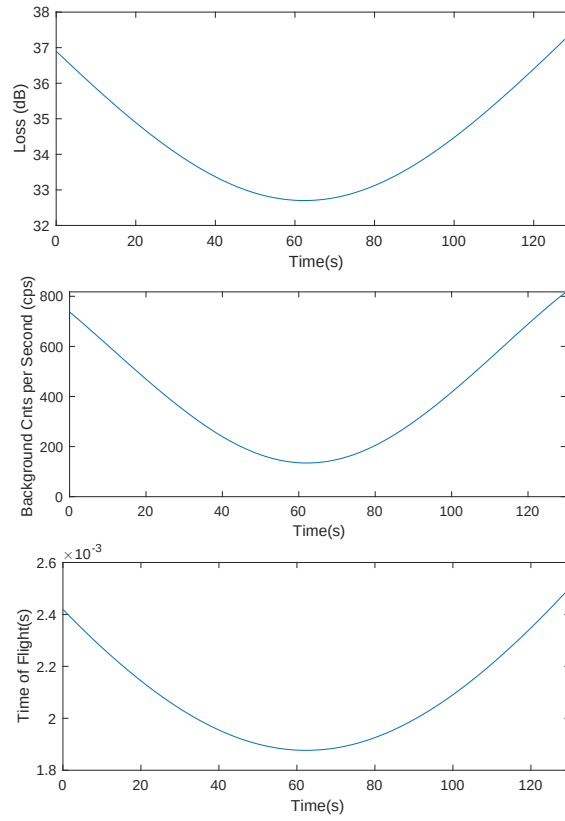
1. **The sender** uses a single photon source to generate a sequence of randomly polarized qubits (pulses) in the X- and Z-bases with variable photon counts (Poisson distributed; more on this in Section 3.2.3). The pulses are time tagged.
 - Errors are applied to the pulses: laser jitter, timing jitter, etc.
 - For quality assurance, the sender retains some of the emitted pulses, and measures them with detectors which also have error. This is better described in the receiver's description below.

2. **The eavesdropper** exists along the channel between the sender and receiver and has three defining properties: whether or not they're present, the proportion of pulses they are measuring, and the time delay added to every pulse as a result of their interference.
 - If present, the simulator generates a time tag file for the eavesdropper at the end of the protocol.
 - The presence of an eavesdropper changes the detection statistics on the receiver's end.
3. **The channel** is defined by (1) the channel loss (subtracting photons), (2) the background counts (adding photons), and (3) the time of flight at each time step. See Figure 12. It also includes polarization distortion (altering photons).
4. **The receiver** is mostly characterized by the detectors they have in their setup. The errors induced by the detectors includes afterpulses (adding photons), dead-times (subtracting photons), dark counts (adding photons), and jitter (altering photons). More details in Subsection 3.2.2. Detections are time tagged.

Notes:

- The sender has the choice of using a weak coherent pulse (WCP) source, or an entangled source, or a quantum dot source. In the optimization work that follows, I used the WCP source.
- All detectors are part of passive detection schemes. Meaning, each incoming pulse is sent through a 50:50 beam splitter towards the X- or Z-basis detector pairs, representing the receiver's choice of measurement basis. Then the pulse is sent through a polarizing beam splitter to determine its polarization.
- The downlink RefQ project referenced earlier in this section will be implementing the 4-state 6-state communication protocol [82, 87]. As such, the receiver would have two additional detectors (and polarizations correspondingly).

Figure 12: For a selected pass: a) Time varying channel loss. b) Time varying background counts per second. c) Time of flight for varying GPS positions. Reprinted from [86].



3.2 Optimizing QKD

The intention in this section is to uncover how the optimization of our specific applied problem was handled. We discuss which objectives were chosen to be optimized, and which arguments they were optimized over. Then, we detail how these arguments were implemented in the simulation. In the the end, we briefly discuss some challenges involved in running the required optimization algorithms as well as corresponding simulations, and some practicalities for how these issues were are addressed.

To successfully implement a satellite-based QKD connection, a significant physical challenge must be overcome: choosing the correct configuration of components. A list is provided in Table 1, including major components in our model, and some select parameters with brief descriptions. Our focus will be on the *detector’s applied excess bias voltage*, and the weak-coherent pulsed source’s (laser’s) setting for *average photon number per pulse* – our two arguments. The optimal values for both of these arguments should maximize the security of the protocol, and security is quantified by the *Key Rate*, and to a lesser degree by the *Quantum Bit Error Rate (QBER)* – these two will be our objectives.

3.2.1 Objectives: Key Rate and QBER

When implementing a QKD protocol, the encryption key that the communicating parties end up using for their secret message, is almost certainly some shorter version of the originally sent bit-string. That’s because not all the bits sent end up being received (missing bits). And sometimes, the receiver will record bits that were never sent in the first place (extra bits). Accounting for, and correcting for this mismatch in recorded bits between sender and receiver is called QKD analysis; and it’s the source of much toil for academics and cryptographers working to come up with secure key-sharing protocols.

The goal is always to be able to use as much of the originally sent key as possible. The proportion of “used key to sent key” is called the *Secret Key Rate*, or Key Rate

Table 1: Non-comprehensive list of simulation components and examples of their parameters.

Component	Parameters	Description
Detector	Afterpulsing Probability $[\in (0, 1)]$	Probability of generating an afterpulse.
	Dark Count Rate [Hz]	Rate of dark counts per second.
	Dead Time [s]	Dead time before detector re-ups.
	Efficiency $[\in (0, 1)]$	Probability of making a detection.
	Jitter [s]	Average detection timing error.
	Recharge Time [s]	Recharge time factor.
Channel	Background Rate [Hz]	Mean background counts per second for a fixed link.
	Depolarization $[\in (0, 1)]$	Depolarization factor.
	Optical Transmission Probability	Optical transmission probability for a fixed link.
Laser	Jitter [s]	Average emission timing error.
	Start Pulse Jitter [s]	Mean start pulse emission timing error.
	Mean Photon Number (signal), $\mu \in (0, 1)$	Setting for mean number of photons for a signal pulse.
	Mean Photon Number (decoy), $\nu \in (0, 1)$	Setting for average number of photons for a decoy pulse.
	Frequency [Hz]	WCP laser source pulse frequency.
GPS	Jitter [s]	Average timing error in serial signals.
	Serial Rate [s/byte]	Rate of serial character signals.
	Time Offset [s]	Offset between pulse per second and serial signals.
Timetagger	Resolution [s/tick]	Self-explanatory.
	Tick Drift $[\in \mathbb{R}]$	Drift scale of time-tagger clock relative to simulator time.

(KR) for short. At some points in this section we will also refer to it as the final key rate.

When photons are sent as bits of the key, they carry information in the form of polarization. For each bit to be transmitted, the sender must make a random choice of polarization basis to use for encoding information into the photon – horizontal/vertical polarization, or diagonal/anti-diagonal polarization. At the other end of the transmission channel, the receiver must also make a random choice between the two polarization bases every time they measure a photon with their detector.

When all bits have been transmitted, the sender and receiver begin the process of narrowing down which bits can and can't be used for their key. This is done by publicly revealing some agreed-upon portion of their bits. More specifically, they disclose what was the basis choice for each bit. Due to the aforementioned randomness of basis choice, there is inevitably some mismatch between the sender and receiver's disclosed bases, and when they don't match, those bits are discarded. The remaining reduced-size key is called the *Sifted Key*. Finally, for the remaining bits, the sender and receiver compare the bits' polarizations, implementing error correction and privacy amplification protocols to finally obtain the final key. Figure 13 illustrates a simplified version of the key sifting process intuition's sake, demonstrating how the sifted key is obtained from a sender (Alice) and receiver's (Bob's) raw keys (bit-strings sent and measured that coincide in time and can reasonably be considered as corresponding to the same photon).

To formally describe the key rate, we must establish some definitions. Specifically, what is yield, Y_n , gain, Q_μ , and QBER, E_μ .

Definition 3.1: Yield

Yield is defined as the *conditional* probability of the receiver making a detection given that the sender transmits an n -photon state.

In a typical optical implementation, yield, denoted Y_n , can be expressed as

$$Y_n = Y_0 + \eta_n - Y_0 \eta_n \approx Y_0 + \eta_n ,$$

where Y_0 is the background rate, and η_n is the n -photon transmittance.

Definition 3.2: Gain

Gain is defined as the *joint* probability that the sender transmits an n -photon state and the receiver records a detection.

In our model, the overall gain can be expressed as:

$$Q_\mu = \sum_{n=1}^{\infty} Q_n .$$

It is composed of the contributions from different photon numbers, Q_n :

$$Q_n = Y_n \frac{\mu^n}{n!} e^{-\mu} ,$$

where Y_n is the yield, and μ is the average photon number setting on the source.

Definition 3.3: Quantum Bit Error Rate (QBER)

Quantum bit error rate (QBER) is defined as the *joint* probability of observing a detection and having an error. In other words, it is the fraction of errors in the set of signals sent.

In our case, QBER, denoted E_μ can be modeled as a mixture of photon numbers:

$$E_\mu = \frac{1}{Q_\mu} \sum_{n=0}^{\infty} e_n Q_n ,$$

where e_n is error rate of an n -photon state and is expressed as:

$$e_n = \frac{e_0 Y_0 + e_d \eta_n}{Y_n} .$$

Here, $e_0 = \frac{1}{2}$ is the error rate of the background, e_d is the probability that a photon hits an erroneous detector.

Formally, in the asymptotic case for the BB84 protocol, the key rate, denoted R , can be expressed as

$$R \geq 1 - H_2(E_Z) - H_2(E_X) ,$$

where

$$H_2(x) = -x \log_2(x) - (1-x) \log_2(1-x)$$

is the binary Shannon entropy, and E_Z and E_X represent the QBERs in the Z - and X - bases respectively, see Refs. [88, 89] for more details.

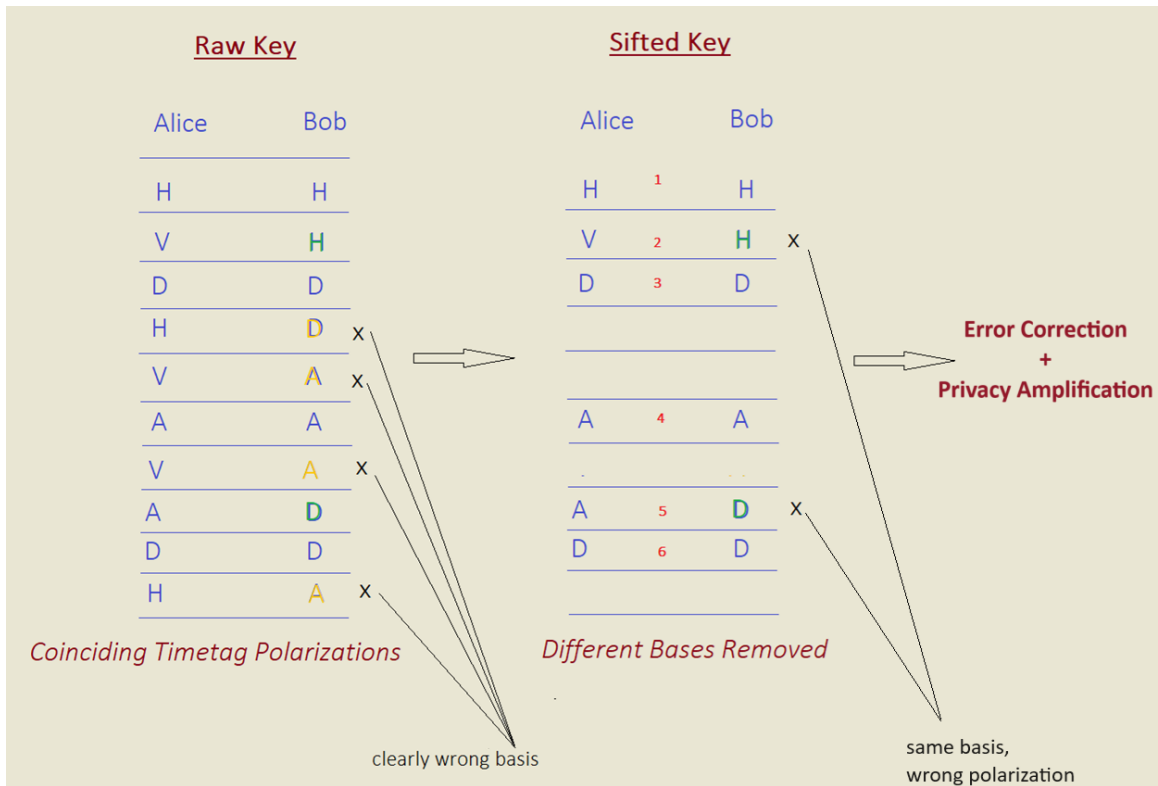
The QEYSSat mission is implementing the Vaccum+Weak decoy state protocol (a variation of the BB84), [90, 91, 92]. In this case, the more accurate key rate expression is:

$$R \geq q \{ -Q_\mu f(E_\mu) H_2(E_\mu) + Q_1 [1 - H_2(e_1)] \} ,$$

where q is set to $\frac{1}{2}$ for BB84 (or 1 for efficient BB84 [93]), and $f(\cdot)$ is an error correction efficiency function. See Ref. [94] for further reading.

In practice, when comparing their keys, the sender and receiver can account for mismatched bases due to random chance. But if after filtering out mismatched bases, they find bits with differing polarizations, then the communicating parties can assume one of two things occurred:

Figure 13: **Simplified example calculating the Sifted Key from the Raw Key.**



1. either the sent photon got lost in the channel (e.g., absorbed in the atmosphere, succumbed to misalignment or device imperfections), and simultaneously the receiver made an erroneous detection due to thermal noise or afterpulsing, or
2. there is an eavesdropper in the channel.

Either way, all errors are attributed to the presence of an eavesdropper. Hence, we use the proportion of mismatched polarizations in the sifted key (aka QBER) to quantify the damage to security.

Besides the fact that key rate is ultimately a function of QBER, another way of describing the relationship is: an improving QBER means improving the KR. As far as our optimization problem is concerned, the two objectives are proxies for security. Thus, in this thesis we consider both QBER and KR as objectives, the idea is to investigate if one of them may be a better function to optimize over than the other. Desirable properties for optimizability include:

- Simpler function.

This would reduce computational complexity. E.g., if the one function is a lower order polynomial than the other, or even more fundamentally, if one is constant in some coordinates where the other is not.

- Noise.

This refers to the variance of the objective at any given input point, from simulation to simulation. The lower the variance (conversely, the more consistent the output), the more accurate each iteration calculation will be in the optimization, requiring fewer iterations.

- Smoothness.

This refers to the variability of neighbouring points, and is related to noise.

Beyond the basic calculation for KR and QBER described above, there is a whole theory of post-processing to error-correct and amplify the privacy of the key exchange.

For further reading, see [95, 96].

Ultimately, the KR and QBER are each functions of observables in the communication protocol. In this thesis, we aim to optimize the KR and QBER over some observables. Recalling section 2.1, our task is to find the arguments (values of observables) that will yield the optimal KR and/or QBER values. As such, we make a simulation incorporating the observables and use that to maximize the KR.

3.2.2 Argument: Detector Voltage

Within both Alice and Bob’s setups, there are single-photon detectors (SPD). SPDs come in many varieties including photomultipliers, Geiger counters, and single-photon avalanche diodes (SPAD). Our simulation specifically considers silicon-avalanche photodiode detectors (SiAPD) – a type of SPAD which have relatively high efficiency and relatively low dark-count rates, and most importantly do not require cryogenic cooling as is often the case for other types of single-photon detectors [70]. For all of these reasons, SiAPDs are an appropriate choice for satellite use. These detector components are a focus in the simulation since they are particularly sensitive to environmental factors such as temperature and ambient light. Also, they have numerous parameters exhibiting statistical variation.

Generally, an SiAPD is a semiconductor-based photodiode, which, when charged with an electric current has the capacity to react to incident photons. When operated at a relatively high reverse voltage, also called Geiger Mode, absorbed photons are accelerated in the semiconductor so that they can generate additional charge carriers (electrons and holes). This amplifies the photocurrent by a significant factor, making for a particularly sensitive photodetector [97]. A more detailed explanation follows.

To operate the SiAPD, the user must apply a voltage to the semiconductor, called the **bias voltage**. There is a minimum bias voltage (aka **breakdown voltage**) that must be applied before the detector can conduct and have charge carriers flow. If the bias voltage is larger than the breakdown voltage and there is a mobile charge carrier

in the semiconductor junction, the electric field therein can cause it to displace other charge carriers out of the neighbouring atoms, which in turn displace more charge carriers – this is called an **avalanche**. Note, an avalanche can only occur when the bias voltage is above the breakdown voltage. Once an avalanche is triggered, the SiAPD cannot be used for another detection until the avalanche current is reduced to zero. This reduction to zero is called a **quenching process** and the time it takes is referred to as **dead time**.

SiAPDs do not perfectly detect incident photons, or not detect absent photons. Avalanches can occur for reasons other than incident photons. For example, **dark counts** are avalanches that are thermally triggered. They occur due to defects in the crystal layer and their rate increases linearly with temperature. Their rate also increases with **excess bias voltage** (bias voltage minus breakdown voltage), denoted V_E . Another source of spare avalanches, is called afterpulsing which also occurs due to defects the semiconductor. When an avalanche occurs (from a detected photon or otherwise), an electron can get trapped in the substrate. Once released, it can trigger an extra avalanche – an **afterpulse**. On the other hand, in the presence of incident photons, often the detector fails to trigger an avalanche and thus fails to make the detection. This is partly attributable to the SPAD structure, partly attributable to excess bias voltage, among other variables. [98]. The percentage of correctly detected photons is called **efficiency**, denoted P_b .

Many of the detector parameters mentioned above are integrated into the device and can only be observed but perhaps not controlled, at least not easily [99]. One of those that *can readily* be controlled is excess bias voltage. We explore its effects in the following subsection.

Although excess bias voltage is not explicitly modeled in the QEYSSat simulation, two simulated parameters (dark count rate and efficiency) are dependent on excess voltage. The QKD simulator is thus designed to incorporate these dependencies.

Efficiency

Efficiency, P_b , increases with the excess bias voltage, V_E , according to the following expression:

$$P_b = 1 - e^{-(V_E/V_C)}, \quad (7)$$

where V_C is the characteristic voltage. V_C depends on the semiconductor depletion layer thickness and on a parameter called k_{eff} , which is a weighted average of the ratio of the ionization coefficient of electrons to that of holes. Typically, V_C takes on values of 1-2 V for SiAPDs with a thin depletion layer, and values of 6-16 V for SiAPDs with thick (20-35 μm) depletion layers [100, 101, 102]. The QEYSSat detectors have a manufacturer-described efficiency of 60%. This has been experimentally verified even when operated at -80°C (the expected temperature maintained in orbit) [103]. Using the expression for P_b above, this maps to a $V_C = 21.8271$ for the QEYSSat detectors, see box below for calculation.

Calculation of V_C

From eq. 7:

$$P_b = 1 - e^{-(V_E/V_C)}.$$

Next, we know that for the QEYSSat detectors $P_b = 0.6$ when the excess bias voltage is 20V. Solving for V_C :

$$\begin{aligned} 0.6 &= 1 - e^{-(20/V_C)} \\ 1 - 0.6 &= e^{-(20/V_C)} \\ \ln(0.4) &= -\ln(e^{(20/V_C)}) \\ -\ln(0.4) &= \frac{20}{V_C} \\ V_C &= \frac{20}{-\ln(0.4)} \\ &= 21.8271 \end{aligned}$$

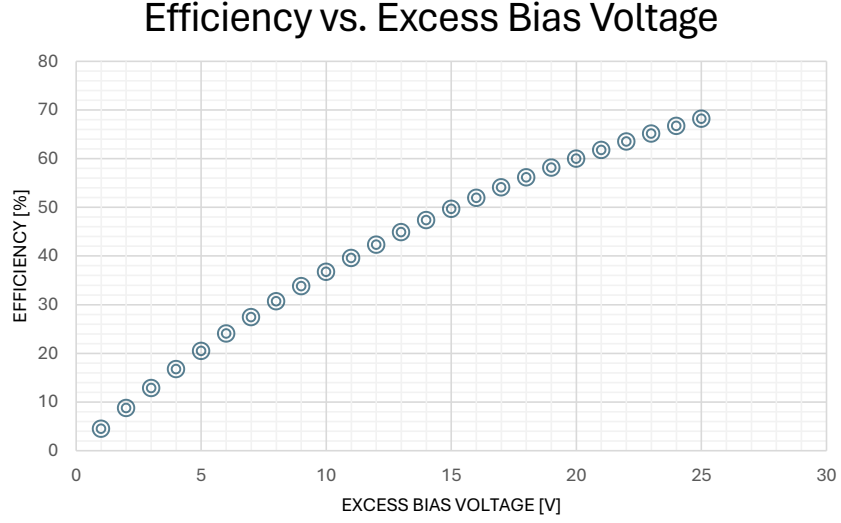


Figure 14: **Detector efficiency as a function of excess bias voltage.**

Finally, the exact expression used in our simulations for efficiency as a function of excess bias voltage is:

$$P_b(V_E) = 1 - e^{(-V_E/21.8271)}.$$

Figure 14 depicts this relationship graphically.

Dark Count Rate

Dark count rate as a function of excess bias voltage does not have an analytic expression derived, but in experiments demonstrates exponential increases with excess bias voltage [97]. Using the experimental results presented in [97], we were able to derive the following expression to model the relationship between dark count rate and excess bias voltage.

$$DCR = C_{\text{temp}} \cdot e^{(V_E/17)} \quad (8)$$

In Equation 8 above, C_{temp} is a scaling factor that accounts for temperature. Temperature is an important dependency to incorporate because dark counts are a function

of temperature. Also, this factor anchors the function at the observed DCR when $V_E = 20V$ (chosen because this is the value of excess bias which has been used in DCR tests at orbit temperatures).

Next I describe how I arrived at the expression for equation 8. First I performed a regression analysis over the experimentally derived data using various models (quadratic, exponential, etc.). The quadratic model yielded an $R^2 = 0.9943$:

$$DCR(V_E) = 0.0465 \cdot (V_E)^2 - 0.8122 \cdot (V_E) + 14.054,$$

and the exponential model yielded an improved $R^2 = 0.9973$:

$$DCR(V_E) = 5.2911 \cdot e^{0.0591 \cdot (V_E)} \approx 5.2911 \cdot e^{(V_E/17)} \quad (9)$$

Next, since the expression for Equation 9 is modeled from an experiment conducted at room temperature, to use a similar expression for the QEYSSat detectors, we must find the analogous expression for the -80°C scenario and $V_E = 20$ [V]. That is, find the corresponding C_{temp} scaling factor. From [104], we know that after 6 months of radiation exposure at similar levels to that of a satellite in orbit, $DCR \approx 0.5$ kHz. See Figure 15. Thus, we can solve the following for C_{temp} :

$$DCR(20) = 0.5 \text{ kHz} = C_{\text{temp}} \cdot e^{(20/17)}$$

Finally, the expression that we used in our simulator is,

$$DCR(V_E) = 0.1542 \cdot e^{(V_E/17)}. \quad (10)$$

Figure 16a compares experimental [97] versus computational results for dark count rate as a function of excess bias voltage at room temperature. Taking a step toward more QEYSSat-relevant results, Figure 16b compares modeled dark count rates at room temperature and for an in-orbit satellite. It is important to note that over time, with consistent exposure to radiation, the detectors will experience increasing dark counts as demonstrated in Figure 16b from [104].

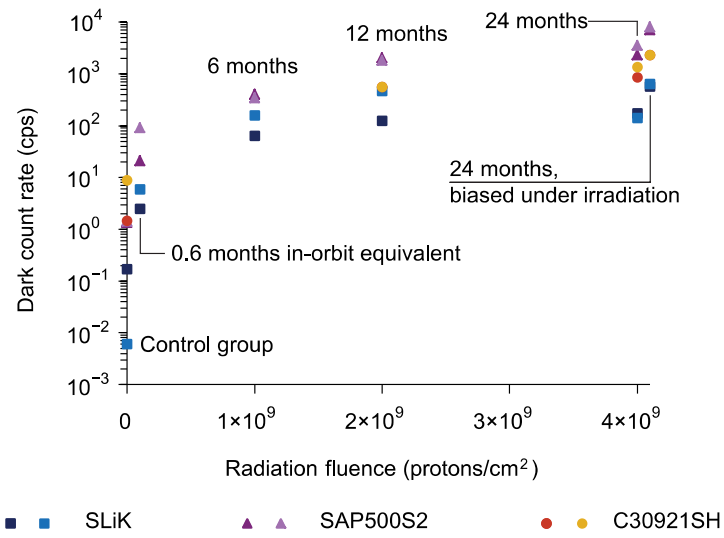
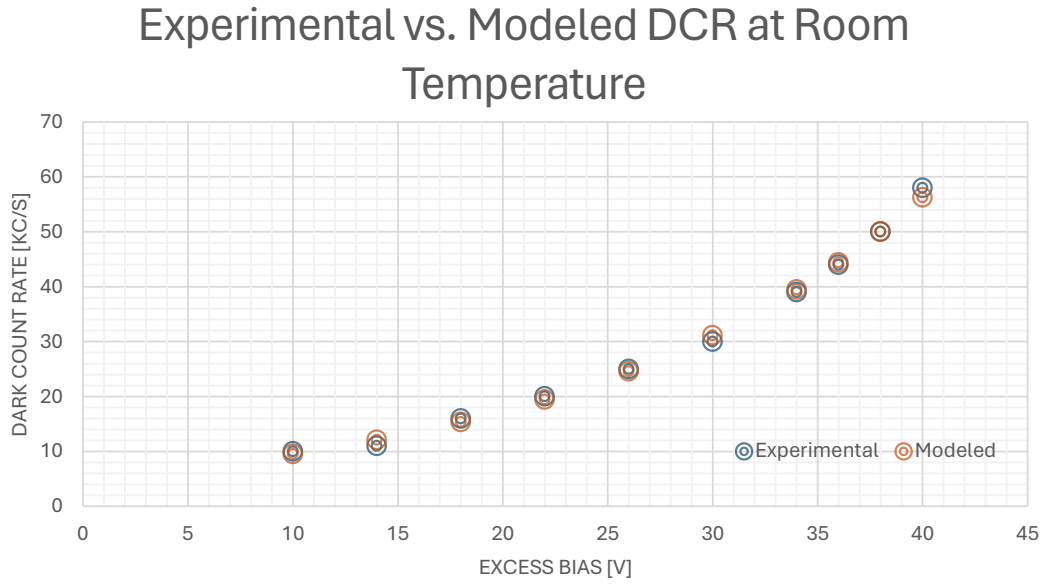
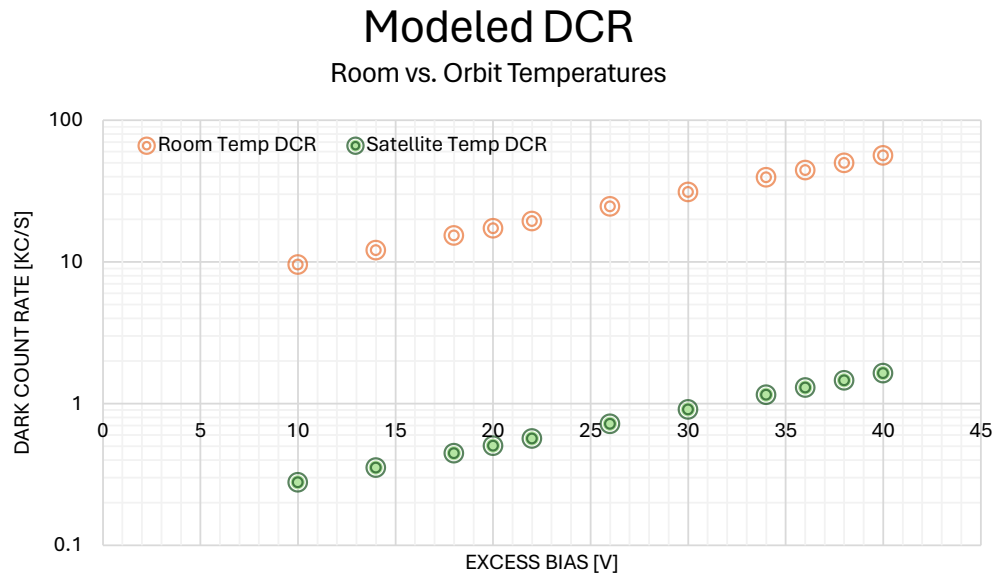


Figure 15: DCR of APD after irradiation, measurement taken at -86°C operation with APDs biased 20 V above their breakdown voltages. Reprinted from [104].

Figure 16: Dependence of the dark count rate on excess bias voltage.



(a) Experimentally obtained [97] versus computationally derived (Equation 9) DCR values at room temperature. This graph demonstrates the high level of accuracy in the model.



(b) A comparison of modeled DCR at different temperatures with a logarithmic DCR axis. See Equations 9 and 10.

3.2.3 Argument: Mean Photon Number

For the purposes of QKD, the sender needs a laser source capable of producing single photon pulses. In the QEYSSat simulator, we’re modeling a weak coherent pulsed source which emits pulses with a low, but varying number of photons at a frequency of 400MHz. That is, the laser emits 400 million pulses every second, and each pulse has a variable number of photons. The number of photons, n , per pulse varies probabilistically according to a Poisson distribution with a mean value denoted by λ :

$$P(n) = \frac{\lambda^n \cdot e^{-\lambda}}{n!}.$$

Below is an example to demonstrate its significance. Suppose we are implementing a vacuum+weak decoy state protocol, [53], with the parameter settings from Table 2.

Intensity	Occurrence	λ	P(empty pulse)	P(single photon)	P(> 1 photon)
Signal	80%	0.5	$P(0) = 61\%$	$P(1) = 30\%$	$P(> 1) = 9\%$
Decoy	14%	0.1	$P(0) = 90\%$	$P(1) = 9\%$	$P(> 1) = 1\%$
Vacuum	6%	0	$P(0) = 100\%$	$P(1) = 0\%$	$P(> 1) = 0\%$

Table 2: **Given the occurrence each type of pulse as a proportion of all emitted pulses, and the mean photon number for each, the table displays the distribution of photon events.**

Then, taking all three pulse intensities together, we would observe that 67.2% of all pulses are empty, 25.5% are single-photon pulses. and the remaining 7.3% represent multi-photon emissions.

In practice, the parameter for the mean value, λ , can be set by the user. As the user, we ask ourselves: what should the mean value be if we want to maximize security? On the one hand, increasing λ would increase the likelihood of the receiver detecting the pulse. On the other hand, the lower the λ , the less likely the source is to produce multi-photon emissions, thus reducing the probability of an eavesdropper

implementing a photon number splitting attack.

Unlike the excess bias voltage parameter, the mean photon number parameter is defined explicitly in the QEYSSat Simulator.

3.2.4 Methodology

I have implemented several blackbox optimization algorithms in MATLAB or Python, and compared their performance as it applies to the simulation of a satellite-based QKD link. There were two argument variables we were optimizing for: excess bias voltage in the detectors, and average photon number in the source. I used two objectives to quantify the security of the link: quantum bit error rate, and final key rate.

In Section 2.3.5, we explored the idea of performance profiles – graphs comparing the relative performance of optimization algorithms. See Figure 9. Here, we detail how these performance profiles are created in practice. First and foremost, the performance profile graph is derived from a table as follows (Table 3):

Table 3: **Sample data for performance profiles, $\tau = 50\%$. Values taken from data tables generated for this thesis. See Appendix for full tables.**

$\tau = 50\%$							
Path Number	Starting Point	OrthoMADS		COBYLA		LINCOA	
		$N_{a,p}$	$f_{acc}^{N_{a,p}}$	$N_{a,p}$	$f_{acc}^{N_{a,p}}$	$N_{a,p}$	$f_{acc}^{N_{a,p}}$
1	[0.56, 10.94]	1	0.9712	1	0.6575	?	?
2	[0.33, 6.77]	-	-	-	-	?	?
3	[0.43, 17.96]	3	0.6433	2	1.02	?	?
4	[0.54, 14.4]	11	0.7260	2	1.3151	?	?
5	[0.2, 5.75]	2	0.6433	-	-	?	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Each tuple of $(N_{a,p}, f_{acc}^{N_{a,p}})$ in the table for each algorithm is obtained by running the optimization from a starting point until it either reaches a satisfactory solution,

or until it elapses the f -call budget. I used an f -call budget of $N=15$. This value was not arbitrary, but rather an upper limit obtained as a result of multiple trials of the OrthoMADS algorithm. The trials showed that across 13 optimizations, if the algorithm reached the 15th iteration without achieving an accuracy of at least 90%, then it would reach the 48th iteration without achieving an accuracy of 90% (possibly higher, but I stopped at 48 which is the equivalent of 4 hours of f -calls). Since each iteration represents one run of the QEYSSat simulation (≈ 5 minutes), an f -call budget of 15 is the equivalent of setting an upper bound of 75 minutes on the time it takes to run one optimization (path).

Algorithms are compared against one another for their ability to efficiently generate a path to the minimum. For this reason, algorithms must generate paths from the same set of starting points. The set of starting points should ideally be somewhat evenly distributed in the domain. Though, one must be careful not to choose points in a way that may result in highly correlated values. Typical methods for partitioning the domain for a set of starting points include Latin Hypercube Sampling (LHS) [10, 105, 106] and low-discrepancy sequences [107] such as Sobol' sequences [108].

For our performance benchmarking task, we ran an LHS in Python using the **Quasi-Monte Carlo submodule**. At a high level, LHS partitions each dimension of the domain into evenly spaced intervals, producing a grid. Then it randomizes the distribution of points by randomly pairing intervals and points from a uniform distribution without replacement. The fundamental rule of LHS is that each interval contains only one point the length of the axis in any direction [109]. Algorithm 5 provides a more detailed presentation of these steps [10].

Algorithm 5 : Latin Hypercube Sampling (LHS)

Given $[l, u] \in \mathbb{R}^n$, a hyperrectangle, and a desired quantity of sample points $p \in \mathbb{N}$,

initialize

$M \in \{1, \dots, p\}^{n \times p}$ matrix where each row is a permutation of the set $\{1, \dots, p\}$.

$k \leftarrow 0$ counter for generated points.

$\mathcal{P} \leftarrow \{\}$ final set of points.

repeat

for $i \in \{1, \dots, n\}$ **do**

$x_i^k \leftarrow l_i + (u_i - l_i) \cdot \frac{M_{i,k} - r_{i,k}}{p}$, where $r_{i,k} \sim U[0, 1)$

$\mathcal{P}.\text{append}(x_i^k)$

end for

$k \leftarrow k + 1$

until $k = p + 1$

return $\mathcal{P}$

The Python code which generated the set of starting points using LHS:

```

1 from scipy.stats import qmc
2
3 # Establishing starting points
4 sampler = qmc.LatinHypercube(d=2)
5 sample = sampler.random(n=25)    # 25 starting points
6 l_bounds = [0.1, 1]             # [u lower bound, V lower bound]
7 u_bounds = [0.9, 25]            # [u upper bound, V upper bound]
8
9 # Output
10 print("LHS sampled 25 starting points")
11 print(sample_scaled)

```

I later manually re-assigned some of the points in the domain to exclude the values of $\mu = \{0.1, 0.9\}$ and $V = \{1, 2\}$, which consistently produced a zero key rate. The final set of starting points is graphed in Figure 17.

To observe how well each algorithm performs as it attempts to find the real optimum, we need to know what that optimum is from the get-go. Filling in Table 3 also requires us to know that value, (x^*, y^*) , since it is used to compute $f_{acc}^{N_{a,p}}$, recall Equation 6. Before starting to run the algorithms, I pinpointed the optimal value by running a quick Grid Search, see Figure 18, and manually refined the area of interest until I found the optimal point on the surface:

$$(x_{\mu}^*, x_V^*, y_{KR}^*, y_{QBER}^*) = (0.56, 17.69, 11800, 0.52).$$

Early security analyses for the decoy state protocol [94] provide validation for our result. They derive an optimal $\mu \approx 0.5$, close to our simulated 0.56.

Note, as a matter of practicality, this optimum is a point I chose *close* to max/min of the objectives, but not *at* the max/min. For example, in Figure 18, we can see that near the maximal Key Rate values, there is a steep cliff to zero. Recall that every time we run a simulation, we get slightly different values for the objectives. Simulations that are run with inputs at the edge of the cliff, will at times output an objective value at the top of the cliff surface (which would constitute an absolute

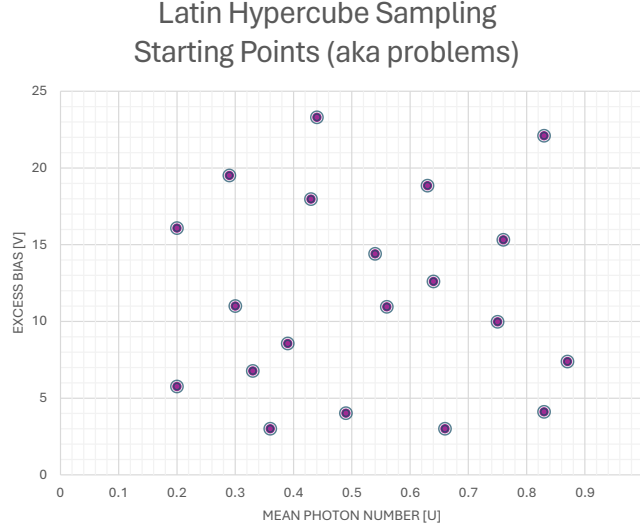


Figure 17: **Set of starting points generated by LHS.**

optimal value); and at other times objective values that are at the bottom of the cliff. The statistical variability within our QEYSSat (Monte Carlo) simulation makes it so that the edge of the cliff has a tendency to shift a little bit. Choosing an optimal Key Rate value, y^* , that is too close to the cliff may be dangerous because our algorithms may not find it if the simulation happens to output a value at the bottom of the cliff instead of at the top. To mitigate this risk, we choose a less optimal value for y^* , but one which perhaps has some buffer to the cliff so as to not likely be affected by the statistical variability.

A note about the objective surfaces in Figure 18. First, key rate counts are lowest in 2 neighbourhoods: (1) near low values of excess bias and/or mean photon number, but also (2) at high values of excess bias and mean photon number. The former can be accounted for by low counts, i.e., not enough coincidences due to low detections on the receiver's end. The latter can be accounted for due to too many detections, causing discarded coincidences.

Figure 18: 3D Surfaces for Key Rate and QBER from Grid Search.

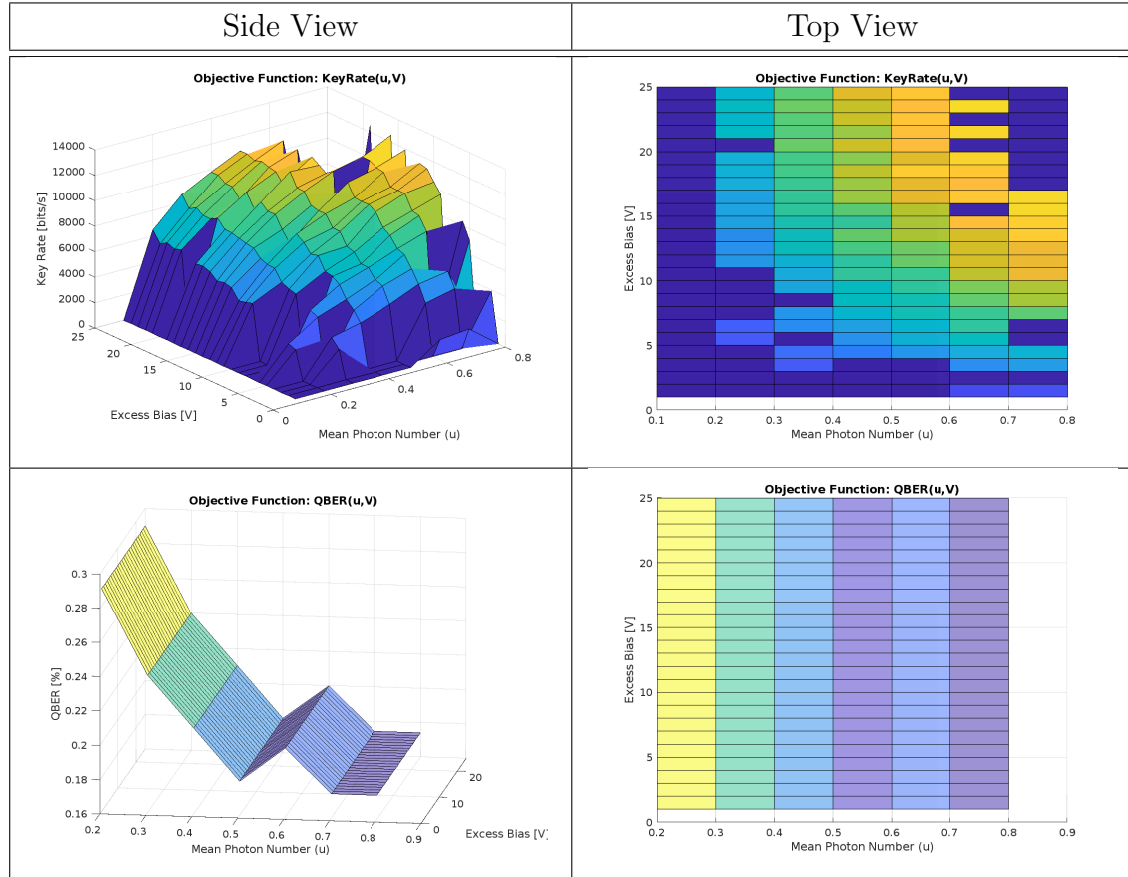
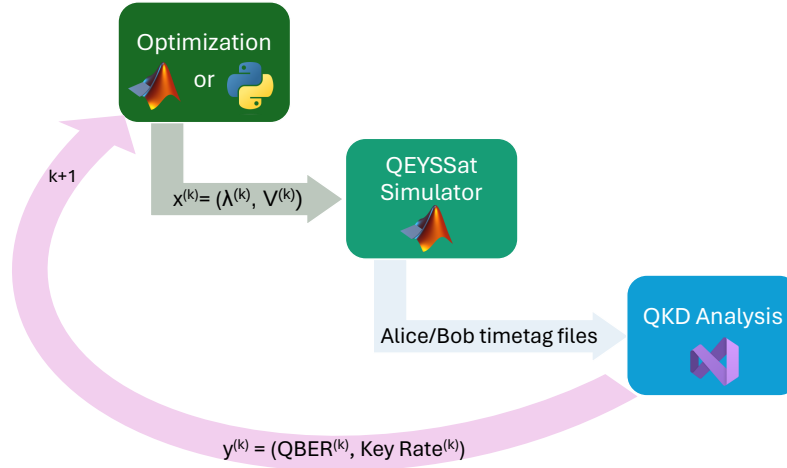


Figure 19: Data flow of optimization software.



A combination of three softwares worked in tandem to implement the optimization tasks. See Figure 19. For each starting point, the algorithms iterated through steps in the domain, seeking an optimum. I worked with one algorithm at a time; first MADS: OrthoMADS, then MBTR: COBYLA, then MBTR: LINCOA.

MADS was implemented in MATLAB using the `patternsearch()` function with the `Algorithm` parameter in `optimoptions` set to "nups-mads". As well, `PollMethod`="OrthoMADSPositiveBasisNp1", and `SearchFcn`="searchlhs".

OrthoMADS was developed in 2009 and implemented in MATLAB in the same year by its authors [34, 110]. It is a specific case of MADS where the search directions are necessarily orthogonal. This is in contrast to traditional MADS which has the capacity to generate a mesh with a non-orthogonal basis (this is possible due to the use of the *positive basis*, see Definition 2.3). For an example of a non-orthogonal mesh see Figure 4.

The mesh-adaptive direct search optimization algorithm code implemented in MATLAB is below.

```

1 disp("Remember to set starting point, x0.")
2

```

```

3 % Set patternsearch options.
4 options = optimoptions("patternsearch", Algorithm="nups-mads",
    PlotFcn=@psplotfunccount, @psplotmaxconstr, @psplotdistance,
    @psplotparetof, @psplotparetox, @psplotspread, @psplotvolume},
    PollMethod="OrthoMADSPositiveBasisNp1", SearchFcn="searchlhs",
    Cache="On", Display="iter");
5
6 % Set initial guess and bounds.
7 initialGuess = [0.66, 3.0]; % Initial guess for optimization [mu, V
    ]
8 lb = [0.2, 3]; % [u lower bound, V lower bound]
9 ub = [0.8, 25]; % [u upper bound, V upper bound]
10
11 % Run patternsearch.
12 fun=@objective;
13 [x, fval] = patternsearch(objective, x0, A, b, Aeq, beq, lb, ub,
    nonlcon, options)
14 [x, fval, exitflag, output] = patternsearch(fun, initialGuess, [],
    [], [], [], lb, ub, [], options);
15
16 % Display results.
17 disp('Patternsearch Results:');
18 disp(['Optimal x: ', num2str(x)]);
19 disp(['Optimal fval: ', num2str(fval)]);
20 disp(['Exit flag: ', num2str(exitflag)]);
21 disp(['Number of iterations: ', num2str(output.iterations)]);

```

Both COBYLA and LINCOA were implemented in Python where I used the PDFO: Powell's Derivative-Free Optimization solvers library, <https://www.pdf0.net/> [111]. Initially, I tried to implement PDFO in MATLAB, but ran into some difficulties with installing FORTRAN → MATLAB compilers which required legacy software versions.

LINCOA, or *Linearly Constrained Optimization Algorithm*, seeks the minimal value of a nonlinear function subject to linear inequality constraints while avoiding the use of derivatives. LINCOA is a model-based algorithm and follows the general

structure of MBTR outlined in Section 2.3.2. More specifically, it iteratively builds local quadratic models of the objective to estimate gradients and subsequently chooses the corresponding step direction. Although implemented in his original FORTRAN 77 solver, author M. J. D. Powell did not publish a paper to introduce the algorithm. An exposition of the trust-region subproblem from the solver can be found in [112], and a detailed presentation the algorithm can be found in [113].

COBYLA, or *Constrained Optimization by Linear Approximations*, on the other hand was formally introduced by M. J. D. Powell in the 1994 paper [36]. Similarly to LINCOA, it is an MBTR algorithm, but differs in a key way: the local models it builds are linear, not quadratic [113].

The model-based trust region optimization algorithm code is below – implemented in Python. This was used for both COBYLA and LINCOA by updating line 25.

```

1 import numpy as np
2 from pdfio import pdfio
3 from scipy.optimize import Bounds
4 import datetime
5
6 ## Optimal values used for computing accuracy:
7 ## mu*=0.56
8 ## V*=17.69
9 ## QBERstar=0.52
10 ## KRstar=-11800
11
12 # Define starting point.
13 print("Remember to set starting point, x0.")
14 print("If maximizing, objective values must be negative.")
15 x0 = [0.66, 3] # Starting point
16
17 # Build the constraints.
18 # Bounds([u lower bound, V lower bound], [u upper bound, V
19         upperbound])
20 bounds = Bounds([0.2, 3], [0.8, 25])

```

```

21 # Solve the problem.
22 options = {'maxfev': 15, 'scale': True}
23
24 ## NOTE: method={'cobyla', 'lincoa', 'uobyqa', 'newuoa', 'bobyqa'}
25 res = pdfo(objective, x0, method='cobyla', bounds=bounds, options=
    options)
26 res.x
27 res.nfev

```

My choice of algorithms (OrthoMADS, COBYLA, LINCOA) was based on the need to benchmark at least one instance of each of the two major algorithm groups: direct-search, and model-based. OrthoMADS was a simple-to-implement and well-established option of MADS. And luckily, the PDFO library offered multiple options for model-based algorithms, and I wanted to compare the performance of the linear (COBYLA) versus quadratic (LINCOA) modeling methods.

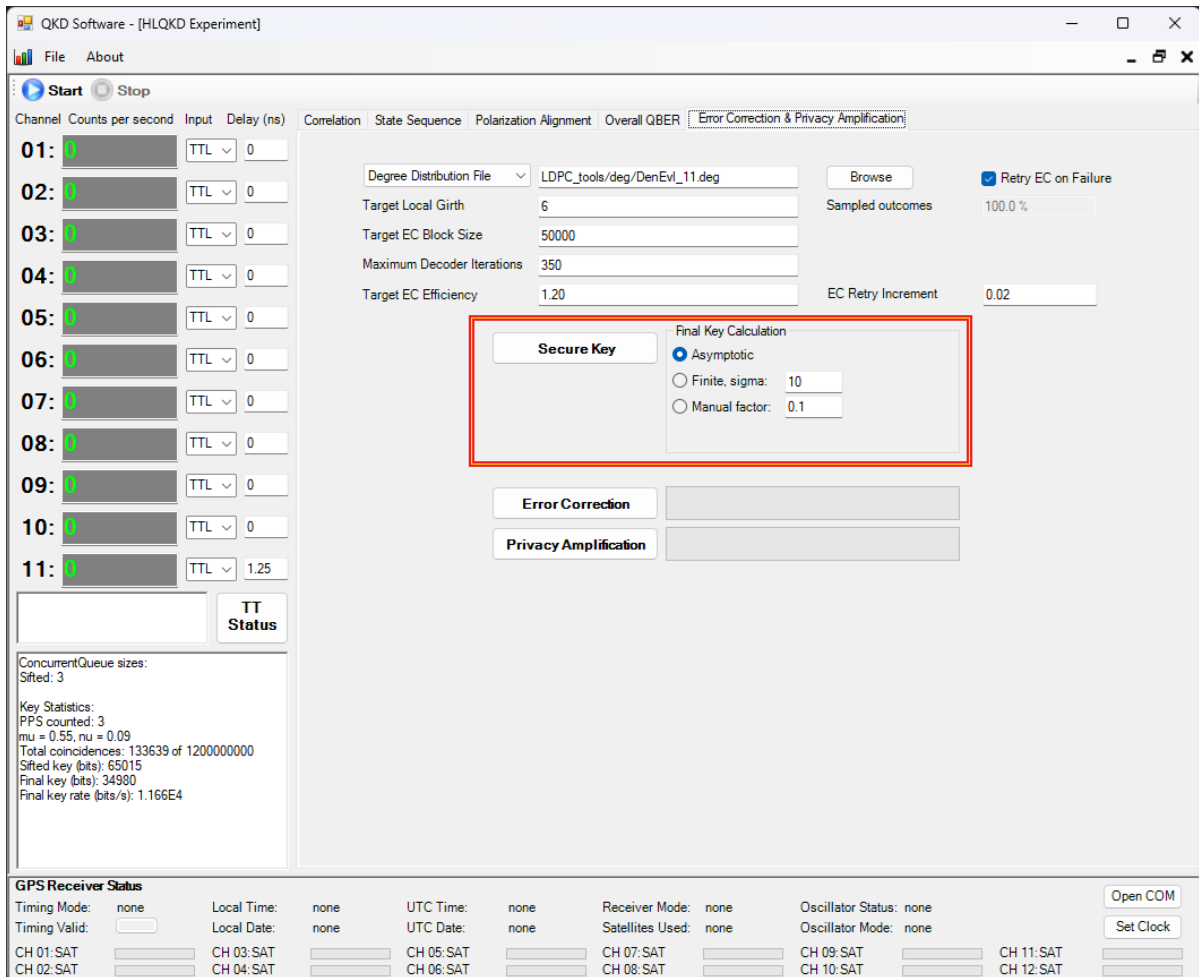
A comment about the QKD Analysis: a choice was made in the QKD Analysis software to calculate the Key Rate as an estimate of an asymptotic key rate, rather than a finite-size key rate. There are notable difference between the asymptotic and finite-size calculations which can be further explored in [95]. Figure 20 shows a screen shot of the window in the QKD Analysis software, highlighting the option to toggle between asymptotic and finite-size key rate calculations.

3.2.5 Surrogate Models

In the framework of BBO methods, the central theme is to limit the number of function evaluations since that is the most resource-intensive part of the process.

Another way of thinking about addressing the “function evaluation” bottleneck is instead of reducing the number of evaluations, to rather reduce the time it takes for each function evaluation by solving a similar problem that is analogous to the one you wish to solve, *but* which has a lower cost of evaluation. The analogous problem is referred to as a *surrogate* [114].

Figure 20: Options for key rate calculation in the QKD Analysis software interface.



A formal definition of the *surrogate* can be found in [10] but in practice, the data collection work conducted here made use of surrogates in two concrete ways [11, 16].

1. Shorter satellite passes simulated.
2. Error correction step not applied to key rate calculation.

First, although our software is capable of simulating a satellite pass of up to 120 seconds, a choice was made to shorten the simulation to 4 seconds (a little more than the minimum required of 3 seconds for the QKD analysis software to work well). In other words, our Monte Carlo sampled simulation was sampled using a significantly smaller set. As a satellite passes overhead, the angle at which it connects with the ground station has a direct impact on the quality of the link; the lower the satellite is to the horizon, the farther the signal travels through the atmosphere, experiencing quadratic losses [115]. As such, the simulation used the central 4 seconds of the satellite pass, illustrated in Figure 21.

Secondly, after conducting a grid search we were able to find that the results of the optimization were nearly identical whether or not we implemented the error correction protocol in the QKD analysis. Omitting the error correction step, halved the evaluation time on average. Figure 22 shows the similarity of the Key Rate for fixed values of Mean Photon Number $[\mu]$. Meaning that once found, the argument in the domain (μ, V) for the optimum will likely be the same whether or not error correction is applied. The caveat being that the absolute values for the objectives would not be accurate until put through the error correction step. It must be noted that Figure 22 was constructed from early data collected with the parameters for DCR and Efficiency having slightly different formulations than in the final data set.

Finally, a quick comment about the objective function that was optimized in the end. Going back to the visualization of the key rate and QBER surfaces, Figure 18; it can be seen from the top view graphs that an improving key rate implies an improving, or at least constant QBER. That is, improving the key rate will never degrade the QBER. This observation was used in the optimization tasks to reduce the data collection burden to only evaluate the optimization over the key rate.

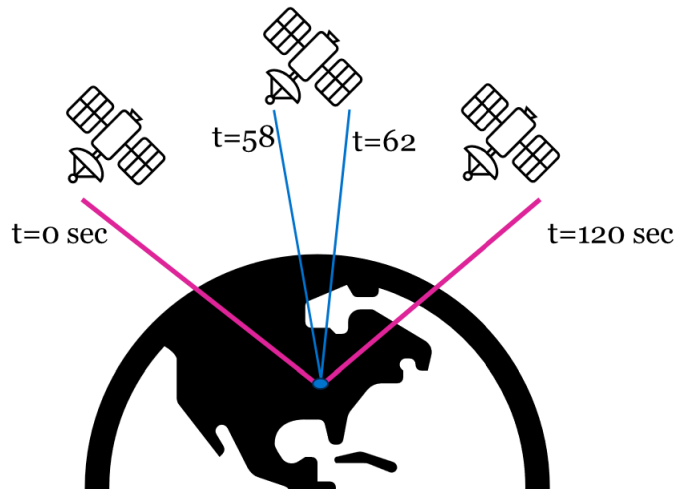
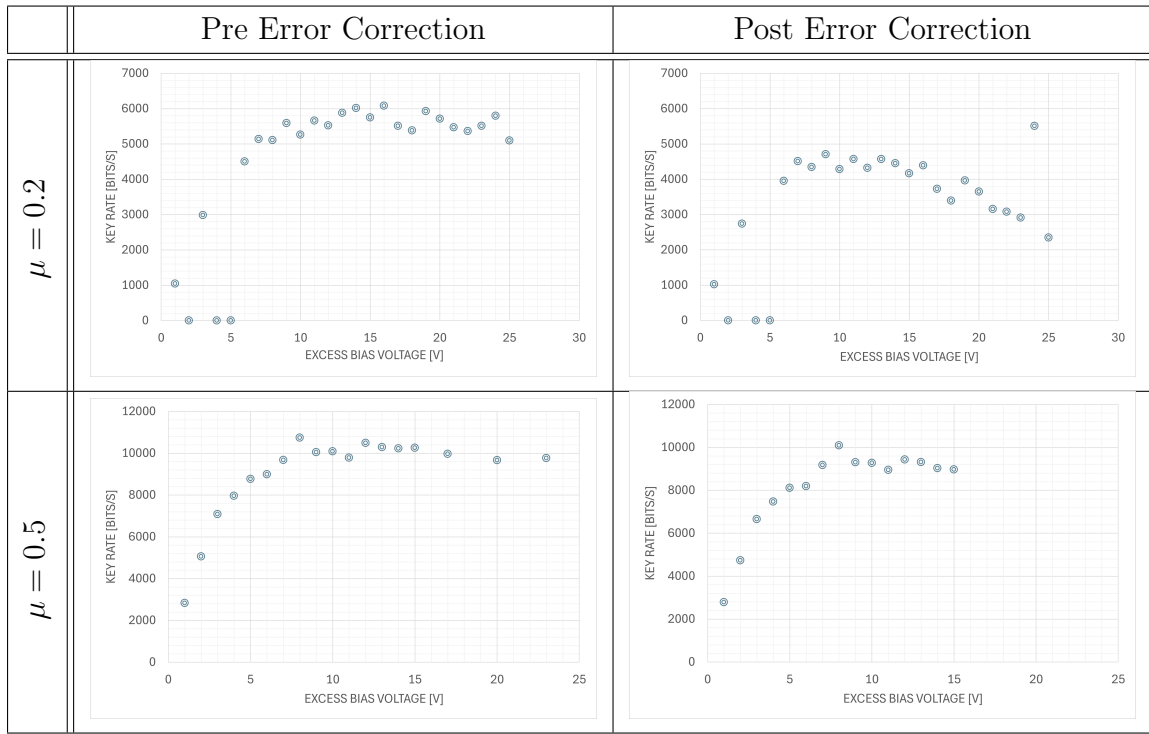


Figure 21: The “shortened satellite pass” surrogate used in the QEYSSat simulations. The interval between the pink radii represents the pass from 0 to 120 second as observed from the ground station (blue spot on North American continent), and the interval between the blue radii represents the surrogate 4 seconds.

Ultimately, if the goal is to reduce the computational cost, then optimizing QBER is a better choice since it has a more linear shape locally than the KR surface, which sports a higher order curvature. Alternatively, KR is better choice of objective if computational cost is less important than the precision of outcomes.

Figure 22: **Observed similarity of Key Rate versus Voltage pre- and post-error correction. Although there is some difference of the quality of the curve as μ increases from the top graphs to the bottom graphs, there is still an evident trend.**



3.3 Results

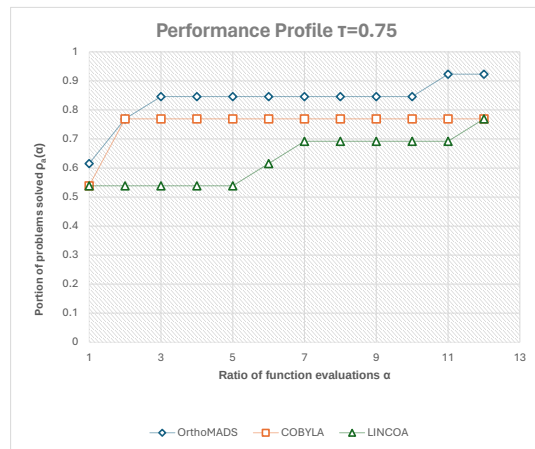
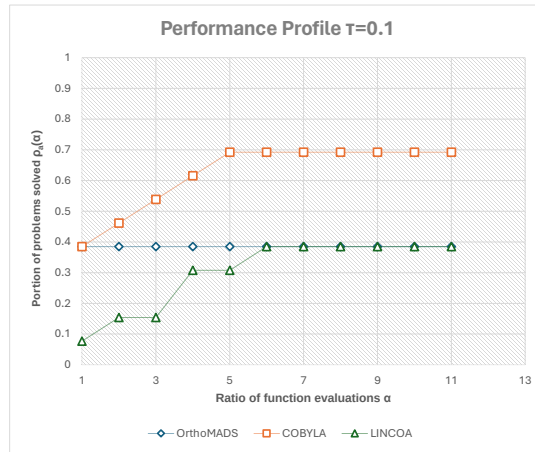
Of the three algorithms compared in the benchmarking, we conclude that the model-based COBYLA method performed best. Interestingly, this happens to be the same algorithm implemented by the University of Strathclyde group for their Satellite QKD

link simulation and optimization, SatQuMA [99, 83]. In the Appendix A, the reader can find a visualization of the paths taken by the three algorithms as they attempt to find the optimum from different starting points.

Recall from Section 2.3.5, that the ultimate benchmark of optimization algorithms is the performance profile graph. We generated these graphs and will now interpret their outputs. Note that for intuition’s sake, it is sufficient to think of the performance profile as “mapping the fraction of problems solved as a function of the allowable number of function evaluations” [12]. The performance profiles in Figure 23 show (at the highest recorded accuracy level $\tau = 0.1$) a significant improvement from LINCOA to OrthoMADS, and again from OrthoMADS to COBYLA. That being said, the COBYLA algorithm only solved 70% of problems with an objective accuracy (not f_{acc}) of 90% given a budget of 15 f -calls. Thus, running the optimization algorithm only once is not a reliable way of obtaining a result. Future optimizations should require the COBYLA algorithm to be run multiple times, each time starting at a different point in the domain, similarly to the way we did it for our benchmarking task. This last point about running the optimization from various points in the domain is crucial, since the algorithm’s success is highly dependent on the starting point in the domain [10, 99]. By running the numbers through a hyper-geometric distribution calculation, I found that **at least four optimizations** will have to be run to arrive within 10% of the optimum with 99% certainty.

For the optimization task at hand, with a 2-dimensional search space and 2-dimensional output-space, the brute force approach of implementing a Grid Search is a viable option for arriving at a result. However, when the optimization task grows any bigger, not to mention the full 49-parameter search space that the QEYSSat simulation offers, perhaps the sophistication of the direct search or model-based algorithms will justify the wrangle with implementation complexity. Future work includes re-evaluating the algorithms above for higher dimensional sample spaces, as well as evaluating the performance of the following list of new algorithms:

Figure 23: **Final performance profiles: OrthoMADS, COBYLA, LINCOA.**



Name	Type	Software Link
UOBYQA	MBTR	PDFO solvers library
NEWUOA	MBTR	PDFO solvers library
BOBYQA	MBTR	PDFO solvers library
Hooke-Jeeves	Direct Search	Iterative Methods: MATLAB
Implicit Filtering	Direct Search	Implicit Filtering
GPS guided by simplex derivatives	Direct Search	SID-PSM
Alternating directions methods	Direct Search	Matrix Computation Toolbox

4 Conclusion

In this thesis we explored the performance of three different kinds of BBO algorithms as they apply to a simulation of the QEYSSat mission. We optimized the objectives of Key Rate and QBER which track the level of security of the protocol, and sought the corresponding values of two arguments: excess bias voltage at the detector, and mean photon number at the source.

After outlining the thesis in Chapter 1, in Chapter 2 we reviewed how a variety of basic BBO algorithms are constructed. We broadly covered the two major groups of derivative-free algorithms: direct search (CS, GPS, MADS), and model-based search (MBTR). This chapter wrapped up with one of the most important sections in the text: how algorithms are benchmarked against one another using Performance Profile graphs.

In Chapter 3, we explored the context and implementation of the optimization task: what is QKD, what is QEYSSat and how QEYSSat was simulated. We explore what were the objectives and arguments, and finally, how we conducted the optimization task, and the result.

The QEYSSat simulator is the blackbox in our BBO scenario, and it fits into a larger suite of programs which cycle through (1) the optimization script, (2) the QEYSSat Simulator, and (3) the QKD Analysis software, see figure 19. We carefully unpacked the major parts of the simulator, which are loosely: the sender’s laser source, the receiver’s detectors, and the channel that connects the two parties. Of the two arguments in our optimization – voltage and mean photon number – only the latter had already been explicitly incorporated in the QEYSSat Simulator. Voltage was not defined as such. However, there were two other parameters in the simulation (detector efficiency and DCR) which are dependent on the voltage and thus were re-defined to include this dependency.

The efficiency parameter as a function of detector excess bias voltage was modeled by

an approximate analytic expression and made for a straight forward implementation of the voltage dependency. Dark count rate proved to be a more challenging function. Using results from separate experiments conducted at room temperature, and at -80°C , which characterized the relationship between excess bias voltage and DCR, we were able to formulate an analytic expression for the dependency and incorporate it into the QEYSSat Simulator. This derivation constitutes a novelty in this thesis.

In the end, we generated performance profiles to compare three optimization algorithms:

1. OrthoMADS (direct search)
2. COBYLA (model-based, linear)
3. LINCOA (model-based, quadratic),

of which, COBYLA performed best. We observed that the arguments that appear to optimize the KR and QBER were $x_{\mu}^* = 0.56$ and $x_V^* = 17.69$. These values are not far off from those estimated by initial developers of the QEYSSat simulator of $x_{\mu} = 0.5$ and $x_V = 20$.

With these results in hand, there is still more work to be done. The QEYSSat mission will require a complete optimization of its simulation over the entire 49-parameter search space. Efforts are underway to completely automate the optimization software suite and eventually conduct this comprehensive optimization task. Until then, we take baby-steps toward our goal and use the COBYLA algorithm to explore relationships between handfuls of arguments at a time – as can be handled by semi-automated simulation efforts. In the near term, we aim to solve for the optimal values of:

- uplink decoy source mean photon rate
- RefQ 4-state 6-state protocol
 - proportion of basis bias

- mean photon rate for signal source
- mean photon rate for decoy source

In the longer term, the hope is to make adjustments to the overall simulation. Presently, the link model gives average loss over time, though there have been observations of up to 10 dB of fluctuations in short term intervals. This work will involve putting together a finer grained loss table. Other updates to the simulator will include the completion of the quantum dot source model. The hope is to eventually conduct the optimization for both entangled and quantum dot sources as well.

In section 3.2.2, we saw that the DCR of the detectors was modeled as having a dependency on voltage and also temperature, though we focused on optimizing for voltage only. Currently, the QEYSSat satellite is designed to operate at -80°C . This design choice has not been optimized for. Once the satellite is in orbit, the DCR will necessarily increase over time due to radiation damage. It is possible that finding a more optimal temperature will decrease DCR and improve the longevity of the detectors. In the future, when the satellite is in orbit, we will receive measurements of the detector performance on a regular basis. Those performance measurements will provide valuable feedback for improving the accuracy of our DCR (and other parameter) models. Our vision going forward is to incorporate increasingly more parameters into our simulations and use the data collected by the satellite from space for validation and feedback.

4.1 Closing Remarks

Conclusions about any algorithm’s performance should always be made against the backdrop of a particular problem class or even particular problem instance. In this thesis we applied this philosophy and found an optimization algorithm that suits the needs of a particular blackbox problem: optimizing the parameters in a free-space satellite QKD simulation. That algorithm was COBYLA.

References

- [1] Mark A. Abramson, Thomas J. Asaki, J. E. Dennis, Kevin R. O'Reilly, and Rachael L. Pingel. Quantitative object reconstruction using abel transform x-ray tomography and mixed variable optimization. *SIAM Journal on Imaging Sciences*, 1(3):322–342, 2008.
- [2] Israel Negrellos-Ortiz, Antonio Flores-Tlacuahuac, and Miguel Angel Gutiérrez-Limón. Dynamic optimization of a cryogenic air separation unit using a derivative-free optimization approach. *Computers Chemical Engineering*, 109:1–8, 2018.
- [3] K.R. Fowler, J.P. Reese, and et al. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 2008.
- [4] Minsu Kim, Areum Han, Jaewon Lee, Sunghyun Cho, Il Moon, and Jonggeol Na. Comparison of derivative-free optimization: Energy optimization of steam methane reforming process. *International Journal of Energy Research*, 2023(Article ID 8868540):20, 2023.
- [5] Jiří Vala and Petra Jarošová. On a derivative-free optimization approach to some problems of civil engineering. *WSEAS Transactions on Mathematics*, 22, 2023(Art. 71):641–651, 2023.
- [6] S. Lucidi, M. Maurici, and L. et al. Paulon. A derivative-free approach for a simulation-based optimization problem in healthcare. *Optim Lett*, 10(Art. 71):219–235, 2016.
- [7] Alison L. Marsden, Jeffrey A. Feinstein, and Charles A. Taylor. A computational framework for derivative-free optimization of cardiovascular geometries. *Computer Methods in Applied Mechanics and Engineering*, 197(21):1890–1905, 2008.

- [8] David Echeverría Ciaurri, Tapan Mukerji, and Louis J. Durlofsky. *Derivative-Free Optimization for Oil Field Operations*, pages 19–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [9] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Ltd, 2000.
- [10] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer Cham, 2017.
- [11] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.
- [12] L.M. Rios and N.V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Glob Optim*, 56:1247–1293, 2013.
- [13] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.
- [14] L.M. Rios. *Algorithms for Derivative-Free Optimization*. Phd thesis, Department of Industrial and Enterprise Systems Engineering, University of Illinois, Urbana, 2009, May.
- [15] D.B. Serafini. *A framework for managing models in nonlinear optimization of computationally expensive functions*. Phd thesis, Rice University, 1999.
- [16] A.J. Booker, J.E. Dennis, and P.D. et al. Frank. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17:1–33, 1999.
- [17] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, NY, 2006.
- [18] R. Hooke and T. A. Jeeves. 'direct search' solution of numerical and statistical problems. *J. ACM.*, 8, No.2:212–229, 1961.

- [19] Virginia Torczon. On the convergence of pattern search algorithms. *SIAM journal on optimization*, 7(1):1–25, 1997.
- [20] Charles Audet and J. E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- [21] Stefano Lucidi and Marco Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, 13(1):97–116, 2002.
- [22] Charles Audet and Warren Hare. *Model-Based Methods in Derivative-Free Nonsmooth Optimization*, pages 655–691. Springer International Publishing, Cham, 2020.
- [23] Stefan M. Wild, Rommel G. Regis, and Christine A. Shoemaker. Orbit: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219, 2008.
- [24] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.
- [25] Stephen C. Billups, Jeffrey Larson, and Peter Graf. Derivative-free optimization of expensive functions with computational error using weighted regression. *SIAM Journal on Optimization*, 23(1):27–53, 2013.
- [26] Warren Hare and Julie Nutini. A derivative-free approximate gradient sampling algorithm for finite minimax problems. *Computational Optimization and Applications*, 56:1–38, 02 2013.
- [27] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Manifold sampling for ℓ_1 nonconvex optimization. *SIAM Journal on Optimization*, 26(4):2540–2563, 2016.
- [28] C. Audet, V. Béchar, and S.L. Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *J Glob Optim*, 41:299–318, 2008.

- [29] Robert Michael Lewis and Virginia Torczon. Rank ordering and positive bases in pattern search algorithms (final report). 1996.
- [30] I. Coope and C. Price. Positive bases in numerical optimization. *Computational Optimization and Applications*, 21:169–175, 2002.
- [31] J. M. Carnicer and J. M. Peña. *Total positivity and optimal bases*, pages 133–155. Springer Netherlands, Dordrecht, 1996.
- [32] C Audet. A short proof on the cardinality of maximal positive bases. *Optim Lett*, 5:191–194, 2011.
- [33] C Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76(4):733–746, 1954.
- [34] Mark A. Abramson, Charles Audet, J. E. Dennis, and Sébastien Le Digabel. Orthomads: A deterministic mads instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.
- [35] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.
- [36] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994.
- [37] Andrew Conn, Katya Scheinberg, and Ph Toint. On the convergence of derivative-free methods for unconstrained optimization. *Approximation Theory and Optimization*, pages 83–108, 01 1997.
- [38] Andrew Conn, Katya Scheinberg, and Ph Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79:397–414, 1997.
- [39] A.S. Bandeira, K. Scheinberg, and L.N. Vicente. Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization. *Math. Program.*, 134:223–257, 2012.

- [40] A. Conn, N. Gould, and P. Toint. *Trust Region Methods*, chapter 6. Global Convergence of the Basic Algorithm, pages 115–168. 2012.
- [41] Jiguan G. Lin. *Three Methods for Determining Pareto-Optimal Solutions of Multiple-Objective Problems*, pages 117–138. Springer US, Boston, MA, 1976.
- [42] Indraneel Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.
- [43] Indraneel Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14:63–69, 1997.
- [44] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [45] Jorge J. Moré and Stefan M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.
- [46] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *CoRR*, cs.MS/0102001, 2001.
- [47] Vahid Beiranvand, Yves Lucet, and Warren Hare. Best practices for comparing optimization algorithms. *Optim Eng*, 18:815–848, 2017.
- [48] W. Wootters and W. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.
- [49] D. Dieks. Communication by epr devices. *Physics Letters A*, 92(6):271–272, 1982.
- [50] Susan Loepp and William K. Wootters. *Protecting Information: From Classical Error Correction to Quantum Cryptography*. Cambridge University Press, 2006.
- [51] Brendon Higgins, Jean-Philippe Bourgoin, and et al. Detailed performance analysis of the proposed qeyssat quantum receiver satellite. In *Conference on*

- Lasers and Electro-Optics 2012*, page JW4A.118. Optica Publishing Group, 2012.
- [52] T. Jennewein, J. P. Bourgoin, and et al. QEYSSAT: a mission proposal for a quantum receiver in space. In *Advances in Photonics of Quantum Computing, Memory, and Communication VII*, volume 8997, page 89970A. International Society for Optics and Photonics, SPIE, 2014.
 - [53] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. Decoy state quantum key distribution. *Phys. Rev. Lett.*, 94:230504, Jun 2005.
 - [54] Hoi-Kwong Lo. Quantum key distribution with vacua or dim pulses as decoy states. In *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, pages 137–, 2004.
 - [55] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, 175, 1984.
 - [56] HK. Lo, H. Chau, and M. Ardehali. Efficient quantum key distribution scheme and a proof of its unconditional security. *J Cryptology*, 18:133–165, 2005.
 - [57] C. Bennett, H., and G. Brassard. Experimental quantum cryptography: the dawn of a new era for quantum cryptography: the experimental prototype is working]. *SIGACT News*, 20(4):78–80, 1989.
 - [58] idQuantique. www.idquantique.com/, 2008.
 - [59] evolutionQ. <https://www.evolutionq.com/>, 2015.
 - [60] Hua-Lei Yin, Teng-Yun Chen, and et al. Measurement-device-independent quantum key distribution over a 404 km optical fiber. *Phys. Rev. Lett.*, 117:190501, Nov 2016.
 - [61] Alberto Boaron, Gianluca Boso, and et al. Secure quantum key distribution over 421 km of optical fiber. *Phys. Rev. Lett.*, 121:190502, Nov 2018.

- [62] Jiu-Peng Chen, Chi Zhang, and et al. Quantum key distribution over 658 km fiber with distributed vibration sensing. *Phys. Rev. Lett.*, 128:180502, May 2022.
- [63] S. Wang, ZQ. Yin, and DY. et al. He. Twin-field quantum key distribution over 830-km fibre. *Nat. Photon.*, 16:154–161, 2022.
- [64] T. Schmitt-Manderbach, H. Weier, and et al. Experimental demonstration of free-space decoy-state quantum key distribution over 144 km. In *2007 European Conference on Lasers and Electro-Optics and the International Quantum Electronics Conference*, pages 1–1, 2007.
- [65] Christopher J. Pugh, Sarah Kaiser, and et al. Airborne demonstration of a quantum key distribution receiver payload. *Quantum Science and Technology*, 2(2):024009, jun 2017.
- [66] Experimental twin-field quantum key distribution over 1000 km fiber distance. *Physical review letters*, 130(21):210801–210801, 2023.
- [67] William J. Munro, Koji Azuma, Kiyoshi Tamaki, and Kae Nemoto. Inside quantum repeaters. *IEEE Journal of Selected Topics in Quantum Electronics*, 21(3):78–90, 2015.
- [68] D. Bernstein and T. Lange. Post-quantum cryptography. *Nature*, 549:188–194, 2017.
- [69] Valerio Scarani, Helle Bechmann-Pasquinucci, Nicolas J. Cerf, Miloslav Dušek, Norbert Lütkenhaus, and Momtchil Peev. The security of practical quantum key distribution. *Rev. Mod. Phys.*, 81:1301–1350, Sep 2009.
- [70] J-P Bourgoin, E Meyer-Scott, and et al. A comprehensive design and performance analysis of low earth orbit satellite quantum communication. *New Journal of Physics*, 15, Feb. 2013.
- [71] SK. Liao, WQ. Cai, and WY. et al Liu. Satellite-to-ground quantum key distribution. *Nature*, 549:43–47, 2017.

- [72] Imran Khan, Bettina Heim, Andreas Neuzner, and Christoph Marquardt. Satellite-based qkd. *Opt. Photon. News*, 29(2):26–33, Feb 2018.
- [73] Juan Yin, Yuan Cao, Yu-Huai Li, Sheng-Kai Liao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Bo Li, Hui Dai, et al. Satellite-based entanglement distribution over 1200 kilometers. *Science*, 356(6343):1140–1144, 2017.
- [74] JG. Ren, P. Xu, and HL. et al. Yong. Ground-to-satellite quantum teleportation. *Nature*, 549:70–73, 2017.
- [75] Sheng-Kai Liao, Wen-Qi Cai, and et al. Satellite-relayed intercontinental quantum network. *Phys. Rev. Lett.*, 120:030501, Jan 2018.
- [76] T. Jennewein, J. P. Bourgoin, B. Higgins, C. Holloway, E. Meyer-Scott, C. Erven, B. Heim, Z. Yan, H. Hübel, G. Weihs, E. Choi, I. D’Souza, D. Hudson, and R. Laflamme. QEYSSAT: a mission proposal for a quantum receiver in space. In Zameer U. Hasan, Philip R. Hemmer, Hwang Lee, and Charles M. Santori, editors, *Advances in Photonics of Quantum Computing, Memory, and Communication VII*, volume 8997, page 89970A. International Society for Optics and Photonics, SPIE, 2014.
- [77] H. Podmore, I. D’Souza, and et al. QKD terminal for Canada’s Quantum Encryption and Science Satellite (QEYSSat). In *International Conference on Space Optics — ICSO 2020*, volume 11852, page 118520H. International Society for Optics and Photonics, SPIE, 2021.
- [78] Tanvirul Islam, Jasminder S. Sidhu, Brendon L. Higgins, Thomas Brougham, Tom Vergoossen, Daniel K. L. Oi, Thomas Jennewein, and Alexander Ling. Finite resource performance of small satellite-based quantum key distribution missions, 2024.
- [79] Aitor Villar, Alexander Lohrmann, and et al. Entanglement demonstration on board a nano-satellite. *Optica*, 7(7):734–737, Jul 2020.
- [80] Kevin Günthner, Imran Khan, and et al. *Quantum-limited measurements of optical signals from a geostationary satellite*. *Optica*, 4(6):611–616, Jun 2017.

- [81] S. Nauerth, F. Moll, M. Rau, C. Fuchs, J. Horwath, S. Frick, and H. Weinfurter. Air-to-ground quantum communication. *Nature Photonics*, 7(5):382–386, 2013.
- [82] B. Schrier, P. J. Godin, B L. Higgins, V R. R. Muthu, N. Sultana, and T. Jennewein. Demonstration and analysis of quantum key distribution with the refq satellite quantum source. *Photonics for Quantum, SPIE Conference Proceedings*, 2024.
- [83] Jasminder S. Sidhu, Thomas Brougham, Duncan McArthur, Roberto G. Pousa, and Daniel K. L. Oi. *Satellite Quantum Modelling Analysis Software Version 1.1: Documentation*, 2022.
- [84] Openqkdsecurity platform. <https://github.com/nlutkenhaus/openQKDsecurity>. Accessed: 2024-02-25.
- [85] Wenyuan Wang and Hoi-Kwong Lo. Machine learning for optimal parameter prediction in quantum key distribution. *Phys. Rev. A*, 100:062334, Dec 2019.
- [86] Vinodh R. R. Muthu, Alex Maiercan, Dr. Brendon Higgins, Dr. Thomas Jennewein, and Brian Moffat. QKD Simulator Build and Performance Modelling Report, QSS-TN-007-IQC 9F064-170332/001/MTB. Technical report, Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada, 2023.
- [87] Ramy Tannous, Zhangdong Ye, Jeongwan Jin, Katanya B. Kuntz, Norbert Lütkenhaus, and Thomas Jennewein. Demonstration of a 6 state-4 state reference frame independent channel for quantum key distribution. *Appl. Phys. Lett.* 115, 211103, 2019.
- [88] Daniel Gottesman and John Preskill. Secure quantum key distribution using squeezed states. *Phys. Rev. A*, 63:022309, Jan 2001.
- [89] D. Gottesman, H.-K. Lo, N. Lutkenhaus, and J. Preskill. Security of quantum key distribution with imperfect devices. In *International Symposium on Information Theory, 2004. ISIT 2004. Proceedings.*, pages 136–, 2004.

- [90] Won-Young Hwang. Quantum key distribution with high loss: Toward global secure communication. *Phys. Rev. Lett.*, 91:057901, Aug 2003.
- [91] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. Decoy state quantum key distribution. *Phys. Rev. Lett.*, 94:230504, Jun 2005.
- [92] Xiang-Bin Wang. Beating the photon-number-splitting attack in practical quantum cryptography. *Phys. Rev. Lett.*, 94:230503, Jun 2005.
- [93] HK. Lo, H. Chau, and M. Ardehali. Efficient quantum key distribution scheme and a proof of its unconditional security. *J Cryptology*, 18:133–165, 2005.
- [94] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. Decoy state quantum key distribution. *Phys. Rev. Lett.*, 94:230504, Jun 2005.
- [95] Ramona Wolf. *Quantum Key Distribution: An Introduction with Exercises*. Springer Cham, 2021.
- [96] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2011.
- [97] S. Cova, M. Ghioni, A. Lacaita, C. Samori, and F. Zappa. Avalanche photodiodes and quenching circuits for single-photon detection. *Appl. Opt.*, 35(12):1956–1976, Apr 1996.
- [98] M Ghioni, Sergio Cova, Franco Zappa, and Carlo Samori. Compact active quenching circuit for fast photon counting with avalanche photodiodes. *Review of scientific instruments*, 67(10):3440–3448, 1996.
- [99] J.S. Sidhu, T. Brougham, and D. et al. McArthur. Finite key performance of satellite quantum key distribution under practical constraints. *Commun Phys*, 6:210, 2023.
- [100] Henri Dautet, Pierre Deschamps, Bruno Dion, Andrew D MacGregor, Darleene MacSween, Robert J McIntyre, Claude Trottier, and Paul P Webb. Photon counting techniques with silicon avalanche photodiodes. *Applied optics*, 32(21):3894–3900, 1993.

- [101] PP Webb. Properties of avalanche photodiodes. *RCA review*, 35:234, 1974.
- [102] Robert J McIntyre. The distribution of gains in uniformly multiplying avalanche photodiodes: Theory. *IEEE Transactions on Electron Devices*, 19(6):703–713, 1972.
- [103] Ian DSouza. *Silicon avalanche photodiodes for satellite based quantum communication*. Master’s dissertation, University of Waterloo, 2018.
- [104] E. Anisimova, B.L. Higgins, and J.-P. Bourgoin et al. Mitigating radiation damage of single photon detectors for space applications. *EPJ Quantum Technol.*, 4(10), 2017.
- [105] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.
- [106] Ishan Bajaj and M.M. Faruque Hasan. Effective sampling, modeling and optimization of constrained black-box problems. In Zdravko Kravanja and Miloš Bogataj, editors, *26th European Symposium on Computer Aided Process Engineering*, volume 38 of *Computer Aided Chemical Engineering*, pages 553–558. Elsevier, 2016.
- [107] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.
- [108] Josef Dick and Friedrich Pillichshammer. *Special constructions of digital nets and sequences*, page 263–284. Cambridge University Press, 2010.
- [109] Wei-Liem Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5):2058–2080, 1996.
- [110] Charles Audet, Andrea Ianni, Sébastien Le Digabel, and Christophe Tribes. Reducing the number of function evaluations in mesh adaptive direct search algorithms. *SIAM Journal on Optimization*, 24(2):621–642, 2014.

- [111] T. M. Ragonneau and Z. Zhang. PDFO: a cross-platform package for Powell's derivative-free optimization solvers. 2023.
- [112] M. J. D. Powell. On fast trust region methods for quadratic models with linear constraints. *Math. Prog. Comp.*, 7:237–267, 2015.
- [113] Tom M. Ragonneau. *Model-based derivative-free optimization methods and software*. PhD thesis, The Hong Kong Polytechnic University, 2022.
- [114] J. Søndergaard. *Optimization using surrogate models - by the space mapping technique*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2003. Supervisor: Kaj Madsen.
- [115] M. Mohageg, L. Mazzarella, and et al. The deep space quantum link: prospective fundamental physics experiments using long-baseline quantum optics. *EPJ Quantum Technol*, 2022.

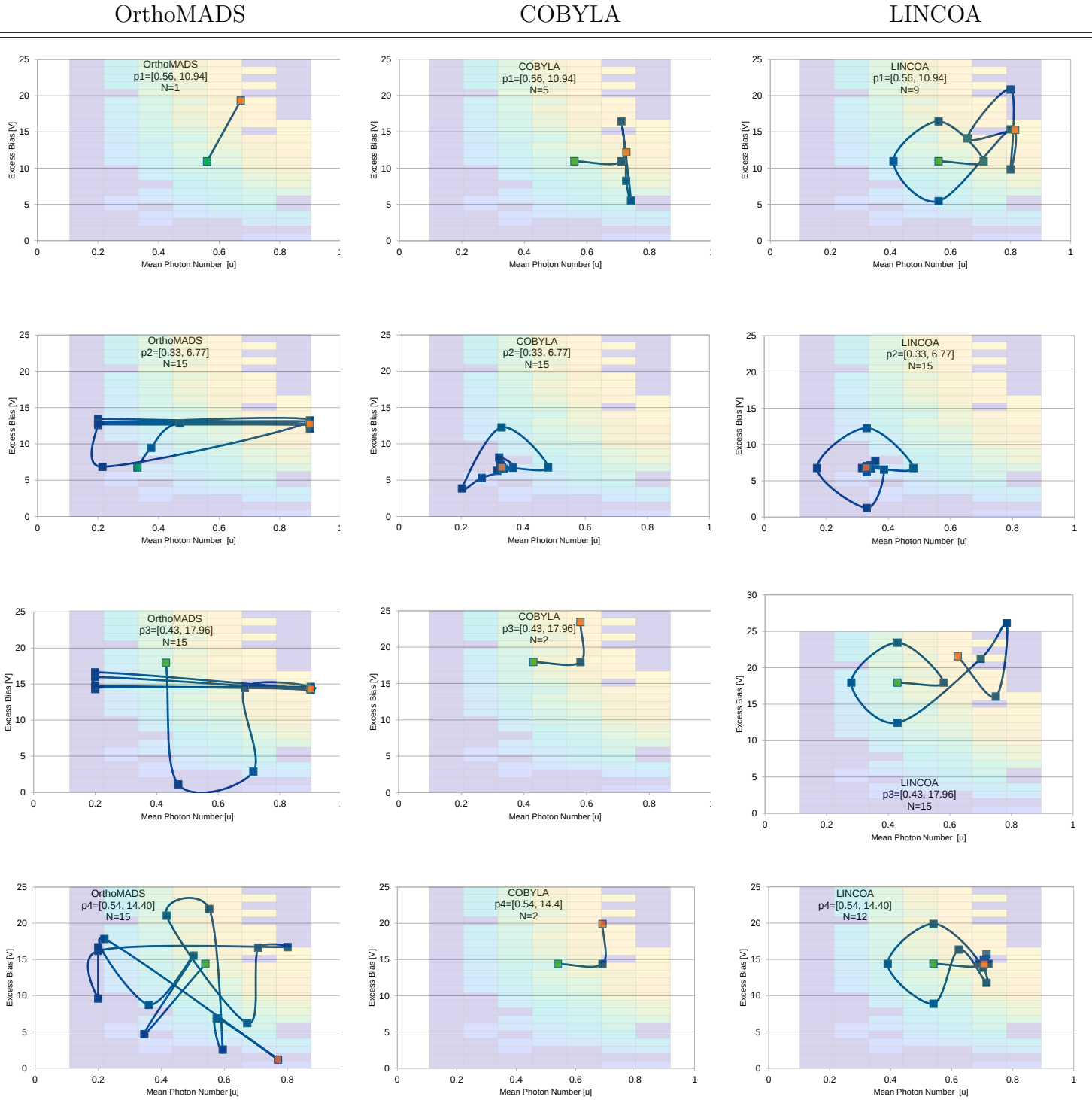
APPENDICES

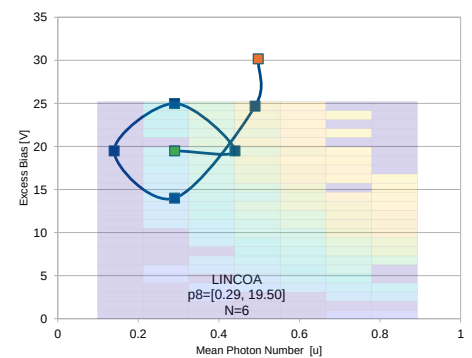
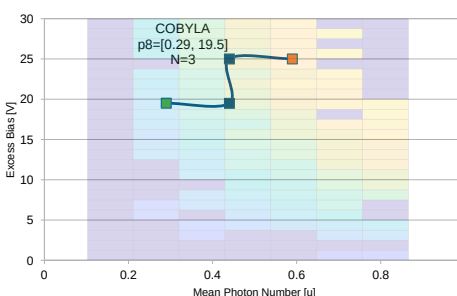
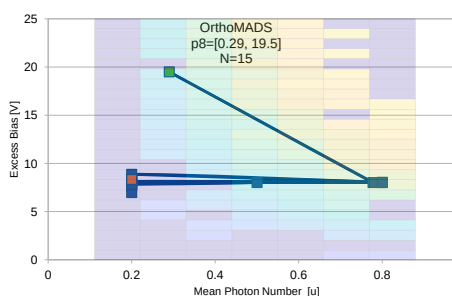
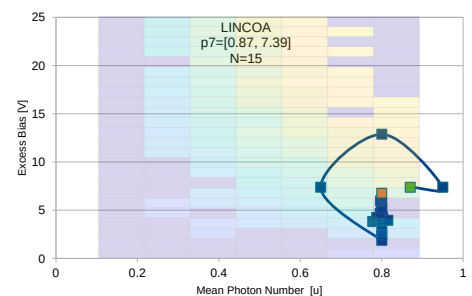
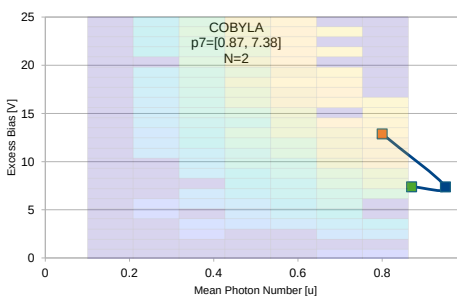
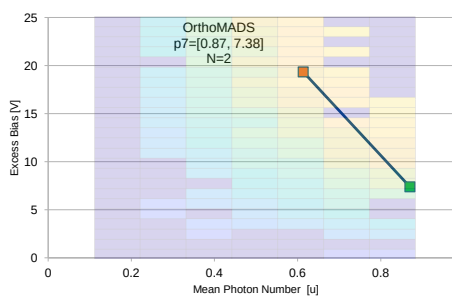
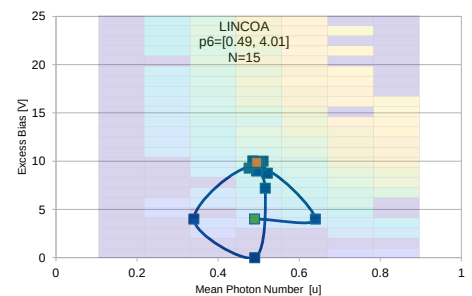
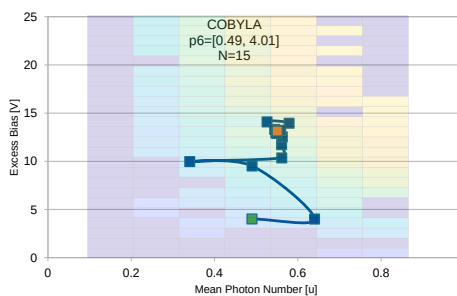
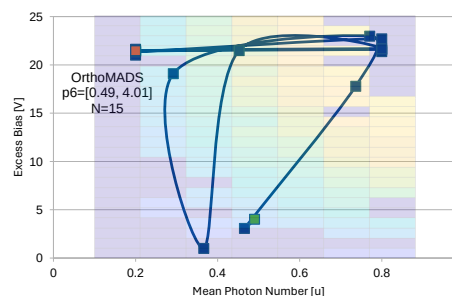
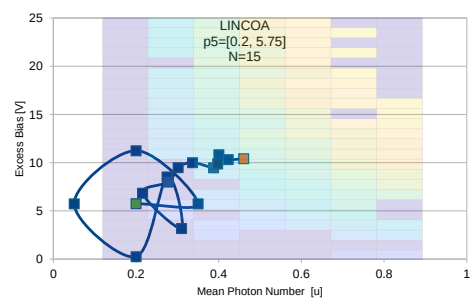
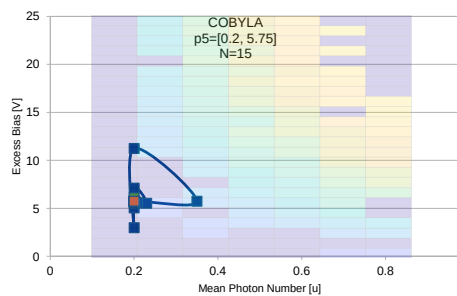
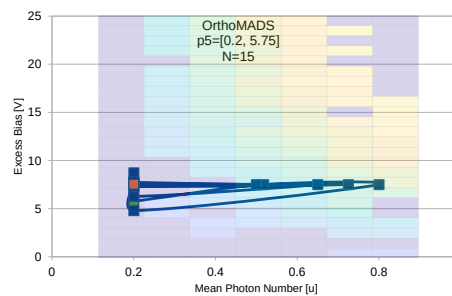
A Comparing Paths

The following table (Figure 24) shows graphs in three columns (OrthoMADS, COBYLA, LINCOA) for the first 12 starting points. Each row represents a starting point common to all three columns. The graphs depict the paths taken by the algorithms from the given starting point (in green), and ending at the stopping value (in orange).

The background in each graph is coloured according to the Key Rate surface gradient in Figure 18. This background is a semi-transparent version of the Key Rate surface top view.

Figure 24: Paths taken by the three algorithms.





OrthoMADS

COBYLA

LINCOA

