NTNU
Det skapende universitet

Modellering og implementering av optimaliseringsalgoritmer for styresystem for retningsantenner

Gunnar Andreas Skogvold

Master i kybernetikk og robotikk
Innlevert: januar 2015
Hovedveileder: Tor Engebret Onshus, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Master's Thesis

Modeling and Implementation of Optimization Algorithms for Control of Directional Antennas

Gunnar Andreas Skogvold

December 2014

Norwegian University of Science and Technology
Engineering Cybernetics

Supervisors

Professor Tor Onshus, NTNU
Ragnar Dag Wik, Kongsberg Defence Systems

Abstract

This report analyzes and tests out different search and optimization algorithms for a sub-problem of a larger automatic control system for two rotating high-gain directional antennas. The goal of the larger system is for the two rotating antennas to automatically find each other, optimize the signal between them and lock onto the optimal orientation. The sub-problem that this report will focus on is the optimization phase after the two antennas have found each other and established communication.

The optimization will be done through a searching algorithm called lobe search, that maps the signal strength around the initial connection point. It uses this information to feed initial values into the optimization algorithm.

This project explores three different gradient optimization algorithms: Steepest Descent Method, Newtons Method and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method, as well as a non-gradient optimization algorithm called the Nelder Mead Method. The tests are done through simulations of the system created in MATLAB, as well as a real life test with a single rotating parabolic antenna optimizing towards a stationary antenna. The goal of the optimization is for both antennas to be pointing directly at each other to ensure the highest possible received power level of the signal between them.

Results from the real life test shows that the best algorithm is a simple slow lobe search without any optimization algorithm afterwords. A slow lobe search is faster and results in better received power level then a quick lobe search running the Nelder Mead optimization afterwords. Gradient optimization algorithms is deemed useless because the information the algorithms are dependent on already contains the optimal point. These conclusions come from ideal test cases and the Nelder Mead algorithm could still be useful in more demanding environments.

Sammendrag

Denne rapporten analyserer og tester ut forskjellige søke- og optimaliseringalgoritmer for et delproblem av et større automatisk kontrollsystem for to roterende høyforsterkende retningsantenner. Målet med det større systemet er for de to roterende antennene å automatisk finne hverandre, optimalisere signalet mellom dem for å så låse seg til den optimale retningen. Delproblemet denne rapporten vil fokusere på er optimaliseringsfasen etter de to antennene har funnet hverandre og etablert kommunikasjon.

Optimalisering vil bli gjort gjennom en algoritme som heter lobesøk, som kartlegger signalstyrken rundt den initielle retningen, og deretter bruker denne informasjonen til å mate initielle verdier inn i en optimaliseringsalgoritme.

Dette prosjektet tester ut tre forskjellige gradient optimaliseringalgoritmer: Steepest Descent metoden, Newton's metode og Broyden-Fletcher-Goldfarb-Shanno (BFGS) metoden, samt et ikke-gradient optimaliseringsalgoritme som heter Nelder Mead metoden. Testene er utført ved å lage simuleringer av systemet i MATLAB, samt reelle tester med en enkel roterende parabolantenne som skal optimaliseres mot en stasjonær antenne. Målet for optimalisering er at begge antenner skal peke direkte mot hverandre for å sikre høyest mulig mottat effektnivå på signalet mellom dem.

Resultater fra de reelle testene konkluderte med at den beste algoritmen er en enkel treg lobesøk uten optimaliseringalgoritme etterpå. Et tregt lobesøk er raskere og gir bedre mottat effektnivå enn et raskt lobesøk med Nelder Mead optimalisering etterpå. Gradient optimaliseringsalgoritmene ble ansett for å være ubrukelig fordi informasjonen som algoritmene er avhengig av allerede inneholder det optimale punkt. Disse konklusjonene kommer fra tester under ideelle forhold og Nelder Mead algoritmen kan fortsatt være nyttig i mer krevende miljøer.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

λ	Wavelength
ϕ	Azimuth
θ	Elevation
f	Frequency
P	Power
R	Distance between two antennas
AS	Azimuth Span
BFGS	Broyden–Fletcher–Goldfarb–Shanno
dBi	Decibel Isotropic
dBm	Decibel-Milliwats
ES	Elevation Span
FLND	Fast Lobe Search and Nelder Mead Optimization
GPS	Global Positioning System
IMU	Intertial Measurement Unit
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SLGO	Slow Lobe Search and Gradient Optimization

1. Introduction

1.1. Project Background

Information sharing has become more and more integral in demanding applications like military applications. Bandwidth requirements for satellite and antenna communications grows with the use of high quality video surveillance in field operations. To compensate for the stricter requirements, antennas must be able to send more data over transmission links. Data rates are in general increased by increasing the signal-to-noise ratio at the receiver. As most radio relay systems are limited in available power due to power constrains in supplies and local heating, the antenna becomes an important element in the transmission budget. Antennas may increase the power in the transmitted direction more than 1000 times – and similar improve the received signal similarly.

Directional antennas are usually utilized in these types of situations, but with longer distances and higher gain requirements, the functional operational area has become quite small. The combination of a small search lobe with the demand of quick and reliable setup time for antenna nodes, automating the task from start to optimal connection between two high-gain directional antennas, will be a great help to setup reliable, high bandwidth networking in demanding locations.

An example use case is a quick field deployment of a video surveillance network. Fig. 1.1 shows an example where the goal of the operation is to positively identify and capture a wanted suspect through video surveillance. Intelligence has gathered that the suspect may travel across a given road, but when this happens is unclear. The network should then be deployed as quickly as possible as not to miss the time window, but also take into consideration that it may be necessary to deploy the network for several days. The proposed solution shows two remote stations, an ID station and a warning station, where the warning station is a simple video stream that alerts the base and ID Station to oncoming vehicles. When the suspect vehicle arrives, the ID Station is alerted and relays the high quality stream to base, as well as takes high quality pictures of the suspected vehicle. The commanding officer at base then checks the pictures and video to see if the suspect vehicle matches intelligence using for example the registration plate. If identification is positive, the base alerts the capture team over radio, and the suspect is captured.

The antenna control system is relevant in this case to quickly set up the high bandwidth link between the ID station and base, as video streams demand reliable high

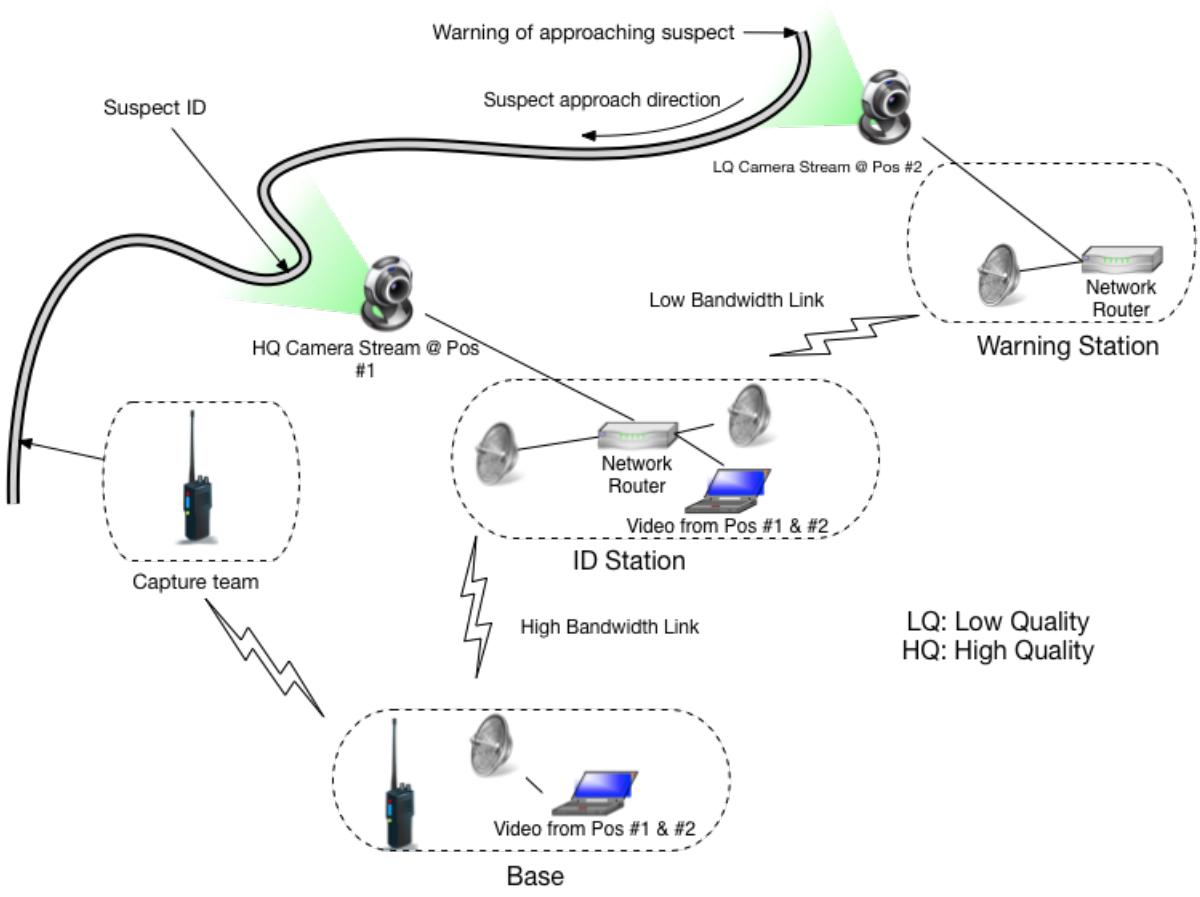


Figure 1.1.: Video surveillance field network

bandwidth connection that can only be achieved using well designed directional antennas. This link will transmit two video streams to base, where the commanding officer has complete overview of the operation, and can positively identify the suspect for the capture team through the high quality video stream.

1.2. Previous Work

A practical concept for automatic alignment of two antennas has been developed by summer students at Kongsberg Defense Systems. A solution for a system design was proposed, and a prototype was built.

1.2.1. System Design

The system proposed is shown in Fig. 1.2 with the following modules:

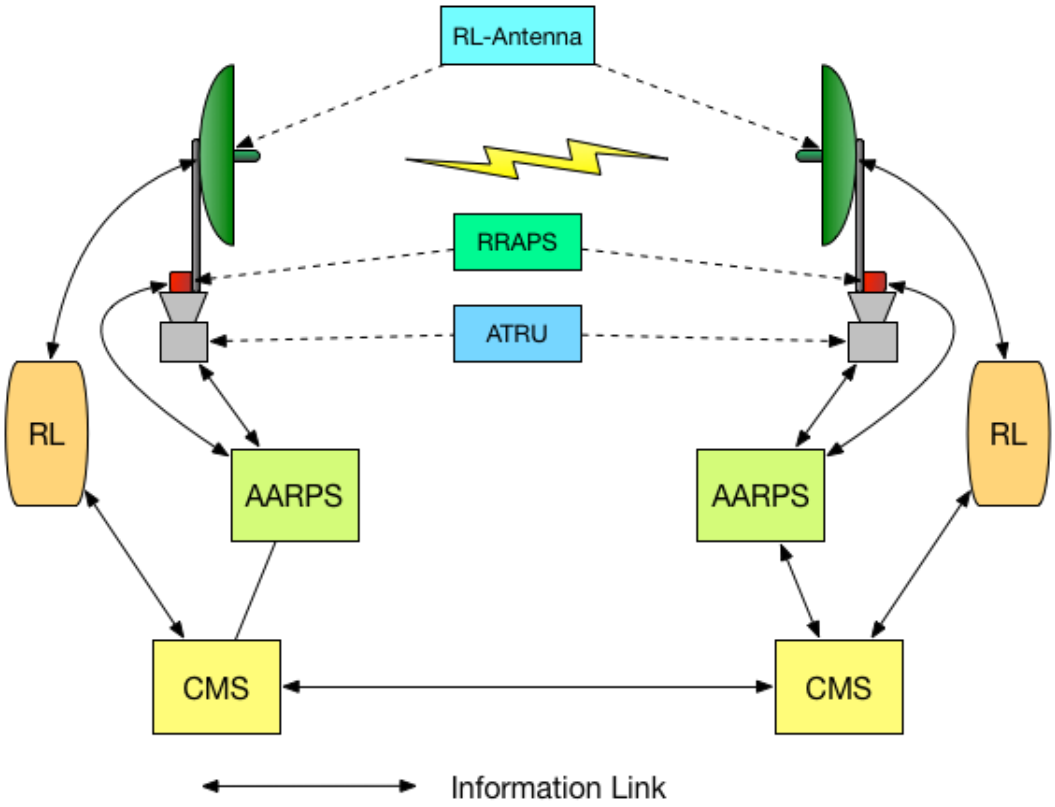


Figure 1.2.: System modules

- The Radio Relay (RL): This is Kongsberg equipment that translates data traffic to/from radio frequencies.
- The Antenna Tilt/Rotator unit (ATRU): This is a pan/tilt rotator that the antenna is mounted on.
- The Radio Relay Antenna Pointing Sensor (RRAPS): This is a sensor system with a Global Positioning System (GPS) and an Inertial Measurement Unit (IMU,) that reports the location and orientation of the antenna.
- The Communication Management System (CMS): This is a system that communicates system data between the two antennas, like data from the RRAPS.
- The Automated Radio Relay Pointing Software (ARRPS): This is the software that runs on an on-board computer that controls the antenna with data from the CMS, ATRU, RRAPS and RL.

Most of the software for this project is part of the ARRPS, as the ARRPS combines data from all the other modules so that the antenna systems can find each other, optimize the signal between them and lock on to the best possible positions. Fig. 1.3 shows the flow chart the ARRPS is built upon. Both systems start with running an independent "blind search," where the degree of automation is dependent on what

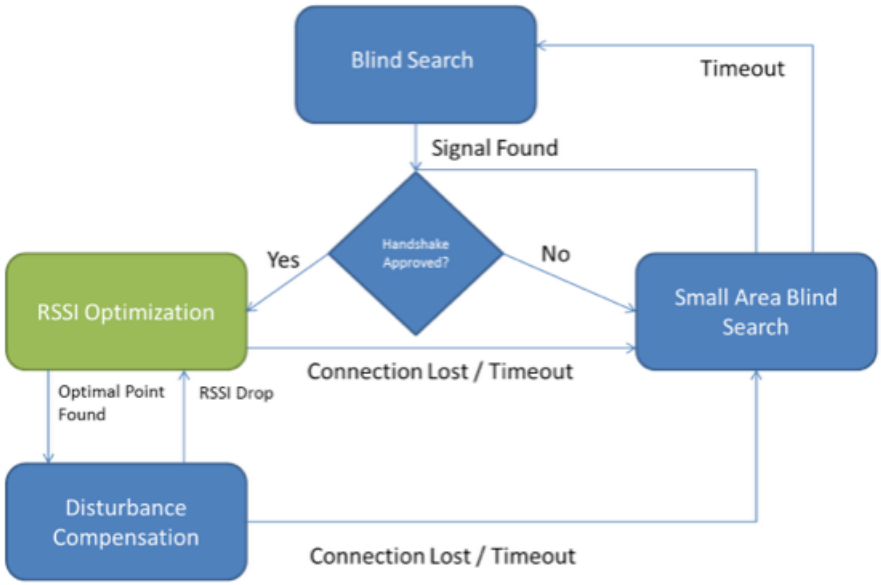


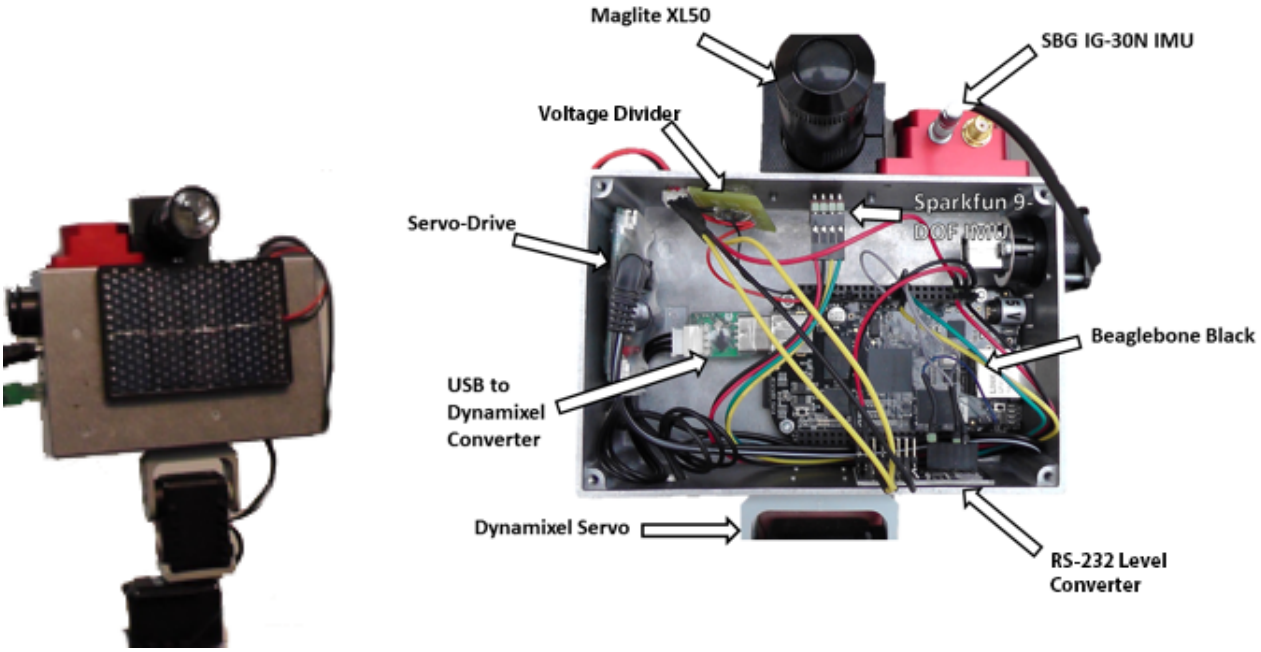
Figure 1.3.: System flow chart

input each antenna has. The best-case scenario is if the system is part of a larger pre-existing network, where a new high capacity link between two nodes that are already connected is needed. The two CMS modules can then share GPS positions between each other over the existing network, and using the Inertial Measurement Unit (IMU) each antenna can rotate towards each other. The more the system knows, the more efficient search algorithms. The best search uses trigonometry from GPS data to acquire the best orientation and the worst search is two completely random blind searches with no restrictions of search area.

When each system has found each other and issued a handshake, the systems start communicating and the next step is to optimize the signal between the two antennas. This will be done through a mix of search and optimization algorithms and state machines that ensure that only one antenna moves at a time. When each antenna is finished with optimization, the optimal orientation for each antenna is presumed found, and the system will lock into this position. This state is called the disturbance compensation state, where the rotator will work with the IMU to compensate for disturbances such as wind and snow and other factors that could move the antenna.

1.2.2. Prototype

A prototype was built to test out blind search algorithms as well as IMU solutions. The prototype was a small scale model of the hardware where the radio antenna was simulated with a flash light and a solar panel. The pan tilt rotator was two servos, the onboard computer was an embedded Beaglebone Black running Debian



(a) Front **(b)** Back

Figure 1.4.: Prototype from summer project

Linux. The IMU was either a low-cost hobby sensor kit from Sparkfun[1] or a high-end industrial grade IMU called SBG IG-30N[2]. Fig. 1.4 shows front and back pictures of the prototypes and the different components inside the metal container mounted on the servos. Two of these were built, one with the high-end IMU and one with the low-cost IMU. The two onboard computers communicated over an ethernet connection, and several search algorithms where tested, as well as disturbance compensation controllers.

A quick conclusion from the prototype is that blind searching is very slow and unreliable. The best way for systems like these to find each other is equipping them with GPS locations, and relaying these locations to each node through existing communication channels. There where also several problems with the prototypes that make them unsuitable for this project. Flash light and solar panels do not accurately simulate radio frequency antennas, especially in an optimization context where specific problems with antenna properties are addressed. The servos where also not powerful and precise enough compared to the sensors, and would often cause oscillations if the servos moved to fast.

[1] https://www.sparkfun.com/products/10724
[2] http://www.sbg-systems.com/products/ig-30-3d-compass

1.3. Project Scope

What this project will focus on is optimizing the signal between the two directional antennas after first contact. In context of Fig. 1.3, only the RSSI optimization state will be focused on. This project works from the following assumptions:

- The antennas start this state in a position where there is bi-directional communication between them for at least small amounts of data.
- The starting position is not the optimal position.

That the antennas can communicate is an important assumption, as only one antenna should be able to move at a time for optimization to make sense. If both move independently of each other the system is chaotic.

The optimization will be completely automatic, and the end result will be that both antennas point at each other with the optimal received signal strength possible for both of them. Either this communication is done directly with the antennas, as this state demands that they have already found each other, or the communication goes through a more reliable already existing connection, but with lower bandwidth.

1.4. Report Structure

This report has already introduced background information about the antenna pointing system, as well as the scope of this project. Chapter 2 will introduce relevant antenna system theory, while chapter 3 will introduce the numerical optimization algorithms implemented in the project. Chapter 4 combines antenna theory and optimization algorithms to create the two big algorithms that are simulated and tested. How the system is simulated is discussed in chapter 5, while the results of simulation tests are presented in chapter 6. Information gathered from the simulations are used to determine what should be tested in the real life tests, where chapter 7 and chapter 8 show how these tests are done and the results. The last part of the report discusses the results, introduces further work before drawing conclusions. The appendix contains some more detailed optimization theory, as well as pictures from the real life tests and data sheets from some of the hardware.

2. Antenna Theory

2.1. Overview

This chapter will briefly introduce the basic concepts on antenna theory that is relevant for this project, especially when it comes to simulating a directional antenna. The chapter will also introduce basics like coordinate systems and decibel systems commonly used in antenna theory.

2.2. Coordinate Systems

The antennas' position and orientation will be parametrized to the following:

- The antennas' position on Earth in GPS coordinates (latitude, longitude, altitude)
- The antennas' orientation in relation to the Earth. See Fig. 2.1
  - Azimuth is angle between North and the antennas' pointing direction projected on the horizontal plane, and is denoted as ϕ.
  - Elevation is the angle between the projection on the horizontal plane and the antennas' pointing direction projected on the vertical plane, and is denoted as θ.

This section will briefly go over the different coordinate system representations used in this project.

2.2.1. Polar Coordinates

Polar coordinates is a parametrization in r and ψ, where a point is represented as an angle ψ from a fixed axis and a distance r from a fixed point. Polar coordinates and xy cartesian coordinates are related through the following equation:

$$x = r\cos(\psi)$$
$$y = r\sin(\psi)$$

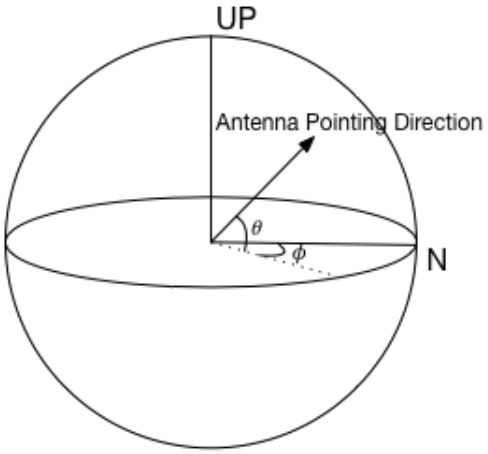


Figure 2.1.: Azimuth (ϕ) and elevation (θ) illustrated

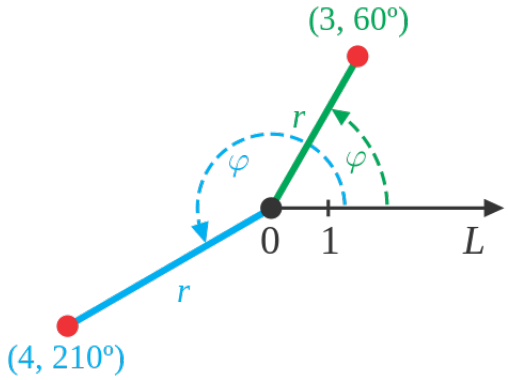


Figure 2.2.: Example of polar coordinates[18]

2.2.2. Spherical Coordinates

Spherical coordinates represent a point on a 3D graph as a distance r from a fixed point, an angle ψ along the azimuth as described in section 2.2, and an angle along the vertical projection called θ'. Note that elevation in section 2.2 equals $90 - \theta'$ if x = N and Z = up. Conversion to 3D xyz cartesian coordinates is given by:

$$\begin{aligned} x &= r\sin(\theta')\cos(\varphi) \\ y &= r\sin(\theta')\sin(\varphi) \\ z &= r\cos(\theta') \end{aligned} \tag{2.1}$$

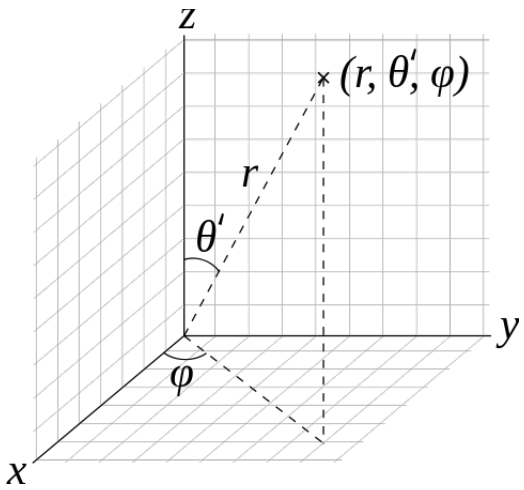


Figure 2.3.: Example of spherical coordinates[19]

2.3. Decibel Scale

The decibel scale is a convenient way to express values that span over several orders of magnitude in a compact way. It is a logarithmic scale that tells the ratio between a value and a reference. It is defined as:

$$L_{dB} = 20 \log_{10}(\frac{A_1}{A_o}) \tag{2.2}$$

Where A_0 is the reference and A_1 is the value. Tab. 2.1 (a) shows a table with common dB values and the corresponding ratio. [20]

(a) Decibel

dB	Ratio
30	1000
20	100
10	10
3	1.995≈ 2
0	1
-3	0.501≈ 0.5
-10	0.1
-20	0.01
-30	0.001

(b) dBm Table

dBm	mW
30	1000 (1 W)
20	100
10	10
3	1.995
0	1
3	0.501
-10	0.1
-20	0.01
-30	0.001

Table 2.1.: Decibel tables

2.3.1. Decibel-Milliwatts

Antennas often operate in milliwatts (mW), so the dBm scale is a dB scale for power ratios where the reference is 1 mW. Because the reference is 1 mW, dBm refers to absolute power measurements. Example values of dBm and the corresponding mW values can be found in Tab. 2.1 (b).[3]

2.4. Antenna Basics For A Single Antenna

This section will introduce relevant antenna theory for a single antenna. Most the theory is taken from the book "Antenna Theory - Analysis and Design" by Constantine A. Balansis and the website Antenna-Theory.com created by Peter Joseph Bevelacqua.[2]

2.4.1. Radio Frequencies

Antennas send and receive Radio Frequencies (RF). These are electromagnetic waves that travel at the speed of light with a certain frequency f.

The frequency of the wave is defined in Hz, and tells how many times the wave oscillates during one second. As electromagnetic waves travel close to the speed of light c, the wavelength λ of the wave is defined as $\lambda = \frac{c}{f}$.

There are many different standards on how the frequency range is divided, but this project will use the broadcast band designation NATO band IV, which is in the frequency range of 4.4-5.0 GHz[1]. This frequency range allows for the signal to be focused to a narrow beam using a parabolic antenna.

2.4.2. Gain

Antenna gain tells how well the antenna converts power to radio waves and vice versa compared to something called a lossless isotropic antenna. It is measured using a decibel scale denoted dBi. A lossless isotropic antenna is a hypothetical antenna that radiates with the same intensity, 0 dBi, in every direction. If an antenna is described as having +27dBi gain in a certain direction, that means that the antenna sends and receives with +27 dB higher power than a a lossless isotropic antenna in that particular direction.

2.4.3. Radiation Patterns

An antenna's radiation pattern describes the direct relation between an antenna's angular direction and the gain transmitted or received. The antenna gain will then

be described as a function $G(\phi,\theta)$, where ϕ is the antenna's azimuth and θ is the antenna's elevation. This function can either be seen as relative to a lossless isotropic antenna and denoted in dBi, or normalized such that the maximum value is 0 and the rest is relative to the maximum and denoted in dB. When discussing the gain of an antenna, the gain is often understood as the maximum value of $G(\phi, \theta)$, as long as $G(\phi, \theta)$ is not normalized.

In Fig. 2.4 we see typical radiation pattern representations. The upper plot shows how the gain can be represented through spherical coordinates where the distance r from the origo to a given point on the plot represents the value of the gain for that angle. The lower plot shows a cross-section of the spherical system where the gain is represented as an amplitude against a single angle. There are usually two plots of this kind, one to represent gain along the azimuth, and another to represent gain along the elevation. These plots assume that the antenna is at the optimal position for the other angle, and are called **principal planes**. The following are usual terms used to characterize a radiation pattern.

- The main lobe - This is typically the lobe with the highest gain value and what the antenna should strive to point with.
- The side lobes - Side lobes are typically lobes lower than the main lobe. In highly directional radiation patterns there are usually two side lobes on each side of the main lobe. The two side lobes have the second tallest peaks.
- The back lobe - This is a lobe that is centered from an 180° angle from the main lobe.
- Null points - Null points are points between the side lobes and main lobes, and are basically points with very low gain.
- The half-power points - These points are the points that are -3 dB$\approx$ 0.5 on the normalized radiation pattern, and are basically the points where the system loses half of the potential power compared to the maximum point.
- The beamwidth - This is the width in angles where the half-power points reside, and is used to characterize how directionally dependent the antenna is.

The beamwidth for a parabolic antenna can be estimated through the following function $BW = 70\lambda/D$, where λ is the wavelength of the emitted radiation and D is the diameter of the parabola. The higher frequency the emitted radiation has, the smaller the beamwidth becomes. [11]

2.5. Antenna Basics For Two Antennas

This section will introduce antenna theory that is relevant to model two antennas pointing at each other.

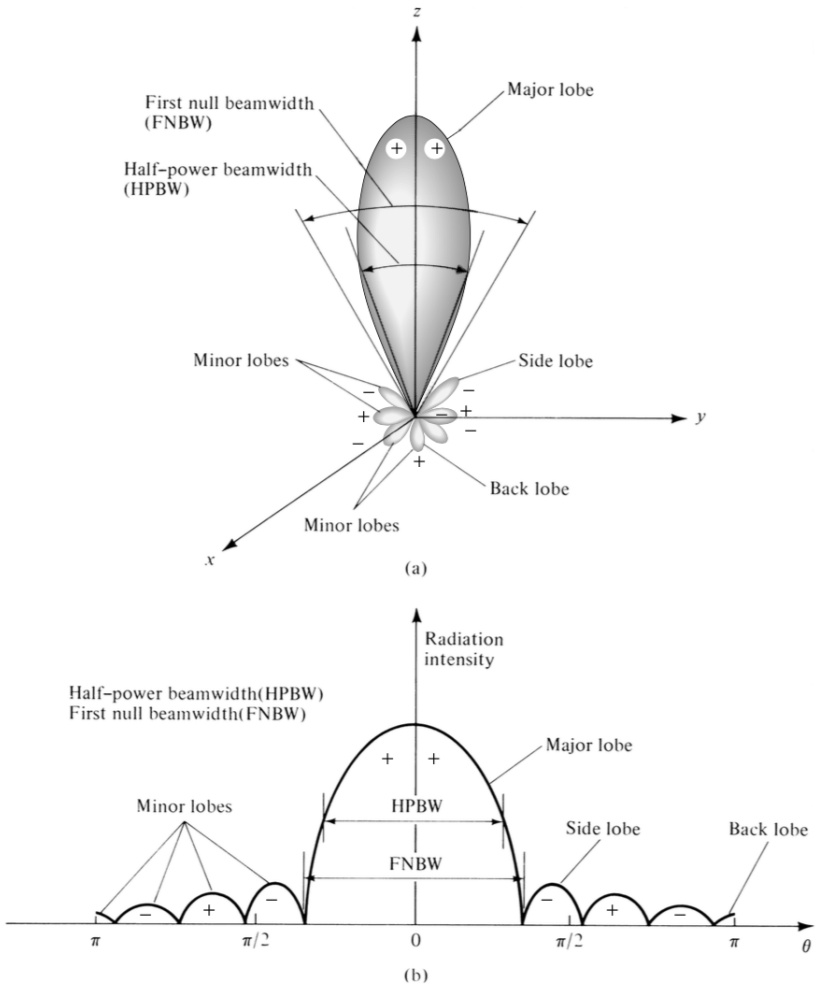


Figure 2.4.: Upper plot: Typical radiation pattern in 3D. Lower plot: Radiation pattern linearized along one axis, called a principal plane [2]

2.5.1. Radiation Patterns With Two Antennas

Each antenna will have a radiation pattern, where the effecting gain is the orientation that points towards the shortest path between the two antennas. This distance is denoted R. Fig. 2.5 shows a 2D example with the radiation pattern of each antenna oriented a small off-set from the optimal direction. Note that the radiation patterns are given in polar coordinates. The thick line points to the gains on the radiation patterns that are relevant for this configuration.

The resulting radiation pattern is dependent on R. Thus R is classified into three regions: The Reactive Near Field, The (Fresnel) Radiating Near Field and The Far (Frauenhofer) Field. These distances are loosely defined through functions that are dependent on the wavelength of the emitted radiation (λ) and the diameter of the emitting dish antenna (D). For the case that $D > \lambda$ the distances are defined as

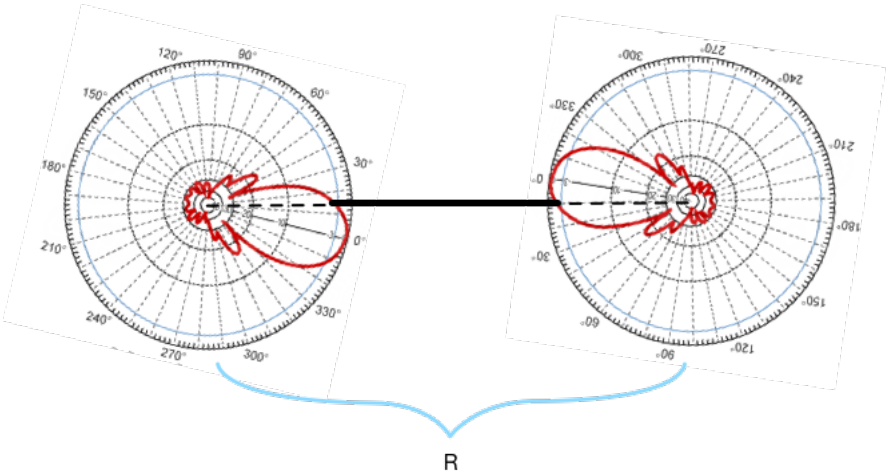


Figure 2.5.: 2D example configuration with two antennas

the following:[2]

- The Reactive Near Field distance is between $0 < R_1 \leq 0.62\sqrt{D^3/\lambda}$
- The Fresnel distance is between $0.62\sqrt{D^3/\lambda} < R_2 \leq 2D^2/\lambda$
- The Frauenhofer distance is $2D^2/\lambda < R_3$

Fig. 2.6 shows how the different radiation patterns can become warped by the distances from the antenna. The Reactive Near-Field shows relatively evenly distributed amplitudes, while the two other fields have distinct lobes. The big difference between Fresnel and Fraunhofer radiation patterns is how the side lobes are are distributed. Far field side lobes are much more distinct, with clear null points between each lobe. Usually radiation patterns operate in the far field, and the lobes represented in Fig. 2.4 are far field.

2.5.2. Friis Transmission Formula

Friis transmission formula is an equation that describes the power an antenna receives when in proximity to a transmitting antenna, and the transmitting power is known. The equation also assumes idealized conditions with a clear line of sight between the two antennas and no other disturbances. The power value is given in dBm and the equation is as follows[4]:

$$P_r = P_t + G_t(\phi_t, \theta_t) + G_r(\phi_r, \theta_r) + 20\log_{10}(\frac{\lambda}{4\pi R}) \tag{2.3}$$

- P_r is power received and is given in dBm.

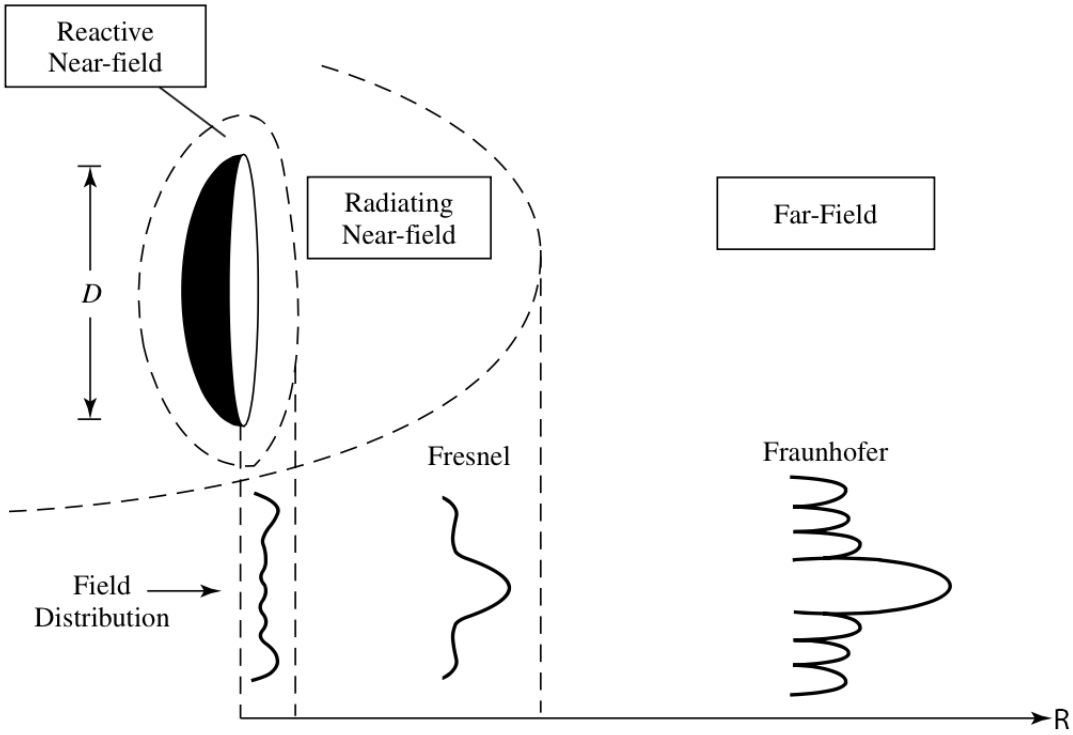


Figure 2.6.: Different field radiation patterns[13]

- P_t is the transmitting power of the remote antenna, given in dBm.
- $G_t(\phi_t, \theta_t)$ is the gain from the transmitting antenna dictated by the radiation pattern, and given in dBi.
- $G_r(\phi_r, \theta_r)$ is the gain from the receiving antenna, dictated by the radiation pattern, and given in dBi.
- $20 \log_{10}(\frac{\lambda}{4\pi R})$is called the free space loss, where R is the distance between the two antennas in meters and λ is the wavelength of the signal transmitted. The free space loss simulates the power the radio frequencies lose by traveling through air, and is given in dB.

2.5.3. Received Signal Strength Indicator (RSSI)

Received signal strength indicator is an indicator of the currently received signal power of an antenna. It is commonly translated from the received power level in dBm, where a higher value indicates a stronger received power level. However, each manufacturer has its own definitions on what values the RSSI should operate in and what values should define a good signal.[8] For this project, only the received power level from Friis Transmission Formula is defined as RSSI.

3. Optimization Theory

3.1. Overview

This chapter will introduce basic optimization problems and the different algorithms tested in this project. The algorithms will be presented as general problems and chapter 5 will present how these algorithms are implemented. The theory in this chapter comes from the book "Numerical Optimization: Second Edition" by Nocedal & Wright. [12]

3.2. The Basic Problem

Optimization theory mostly focuses on the problem of finding minimal or maximal values of functions. The most basic case is written as the following:

$$min_{\mathbf{x}\epsilon\mathbb{R}^n}\ f(\mathbf{x})$$

Where the function $f : \mathbb{R}^n \to \mathbb{R}$ is called the objective function, $\mathbf{x}$ is the state vector and n is the dimension of the problem. An important assumption is that f is a *smooth function.* A smooth function is in this context a function that is continuous and where a second derivative exists.

3.2.1. Local and Global Solutions

When it comes to optimization it is important to identify if an optimal point $\mathbf{x}^*$ is local or global. An optimal minimized solution is the lowest feasible point in a neighborhood of feasible points, and is a local optimal point. For the optimal point to be global the point has to be the lowest feasible point of ALL feasible points for the whole scope of the problem. The algorithms presented in this chapter are designed to find local optimal solutions. Issues around global optimization are minimized through algorithms presented in chapter 5.

3.3. Line Search Methods

Line search methods are common numerical optimization methods, where the solution is found by traversing the solution space where the next states being checked are found through the following equation:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$$

Where k is the current step, $\mathbf{p}_k \epsilon \mathbb{R}^n$ is the current search direction and α is the step length along the direction $\mathbf{p}_k$. The different methods are mostly based on different ways to find good search directions and step lengths.

3.3.1. Step Length - The Wolfe Conditions

Assume a good direction, $\mathbf{p}_k$ is known. How far along this direction should the algorithm jump? An answer to that is to ensure the choice of step length, α, satisfies the Wolfe conditions. The Wolfe conditions are two conditions that ensure that the step length results in a sufficient decrease of the objective function f, as well as ensuring that the step length is not too short. The sufficient decrease condition is given by the following inequality:

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f_k^T \mathbf{p}_k \tag{3.1}$$

Where $c_1 \epsilon (0, 1)$ is a constant. This condition ensures that the α chosen will give lower $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ values than an arbitrary linear function decreasing along the objective value (decided by the constant c_1).

The condition ensuring that the step length is not too short is called the curvature condition is given by the following inequality:

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla f_k^T \mathbf{p}_k \tag{3.2}$$

Where $c_2 \epsilon (c_1, 1)$ is a constant. The left side of the equations is the slope of the current step, and the right side is a desired slope. The condition ensures that the step length is not too short by demanding that the tangent of $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ is larger or equal to a desired slope, decided by c_2.

A line search algorithm that finds an α_k that holds the Wolfe Condition is presented in "Numerical Optimization," and is given by the algorithms Algorithm 3.1 and Algorithm 3.2. In short the algorithm first checks if the sufficient decrease condition

3.1 is held. If it is not held, the algorithm runs the zoom function. The zoom function finds an interval $(\alpha_{lo}, \alpha_{hi})$ where the sufficient decrease function is held. It then tries to find an α_j that holds the curvature condition and returns when found. If Algorithm 3.1 starts with an α that holds the sufficient decrease condition, it checks if the curvature condition is withheld. If this is the case, the α value is returned.

Algorithm 3.1 Wolfe Condition Line Search

Arguments: $(\mathbf{x}_k, \mathbf{p}_k)$

Set $\alpha_0 = 0$, choose $\alpha_{max}>0$ and $\alpha_1 \epsilon (0, \alpha_{max})$

$i \leftarrow 0$;

while

$\mathbf{x}_c \leftarrow \mathbf{x}_k + \alpha_1 \mathbf{p}_k$;

if $f(\mathbf{x}_c) > f(\mathbf{x}_k) + c_1 \alpha_1 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k || ((f(\mathbf{x}_c) \geq f(\mathbf{x}_k) \&\& i>0)$

$\alpha \leftarrow$**zoom(**α_{i-1}, α_i**);**

break;

end

if $|\nabla f(\mathbf{x}_c)^T \mathbf{p}_k| \leq -c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k$

$\alpha \leftarrow \alpha_1$;

break;

end

if $\nabla f(\mathbf{x}_c)^T \mathbf{p}_k \geq 0$

$\alpha \leftarrow$**zoom(**α_i, α_{i-1}**)**

break;

end

$\alpha_{i+1} \leftarrow min(\alpha_{max}, 3\alpha_i)$;

$i \leftarrow i + 1$;

end

3.3.2. The Steepest Descent Method

Gradient methods build upon the assumption that the gradient of f (∇f) is known. The idea is that the gradient holds information to find the best step direction. In

Algorithm 3.2 Wolfe Condition Line Search Zoom

Arguments: $(\alpha_{lo}, \alpha_{hi})$

while

$\alpha_j \leftarrow (\alpha_{lo} + \alpha_{hi})/2;$

$\mathbf{x}_c \leftarrow \mathbf{x}_k + \alpha_j \mathbf{p}_k;$

$\mathbf{x}_{lo} \leftarrow \mathbf{x}_k + \alpha_{lo} \mathbf{p}_k;$

if $f(\mathbf{x}_c) > f(\mathbf{x}_{lo}) + c_1 \alpha_j \nabla f(\mathbf{x}_k)^T \mathbf{p}_k || (f(\mathbf{x}_c) \geq f(\mathbf{x}_k)$

$\alpha_{hi} \leftarrow \alpha$;

else

if $|\nabla f(\mathbf{x}_c)^T \mathbf{p}_k| \leq -c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k$

return α_j;

end

if $\nabla f(\mathbf{x}_c)^T \mathbf{p}_k (\alpha_{hi} - \alpha_{lo}) \geq 0$

$\alpha_{hi} \leftarrow \alpha_{lo};$

end

$\alpha_{lo} \leftarrow \alpha_j;$

end

end

the simplest form there is the steepest descent search, where the step direction is $\mathbf{p}_k = -\nabla f_k$. This ensures that $\mathbf{p}_k$ is always in a direction that minimizes the function f. Combining the algorithm to ensure Wolfe conditions are met for α, one can create a line-search algorithm from the Steepest Descent Method that will ultimately converge on a local optimal point. Pseudocode for the Steepest Descent method is given in Algorithm 3.3.

Steepest Descent has the advantage of not being dependent on second derivatives, but the simplicity is also its biggest drawback, as it can be slow on complex problems. It is also dependent on Wolfe Conditions to find a good step length.

3.3.3. Newton's Method

Newton's method is a gradient search that has the extra assumption that the hessian matrix $\nabla^2 f$ is known for each $\mathbf{x}$. The method sets the step direction as $\mathbf{p}_k = -(\nabla^2 f_k)^{-1} \nabla f_k$, and is built upon the second order Taylor approximation of the

Algorithm 3.3 Steepest Descent Method

Set $\mathbf{x}$ = current x position;

$\varepsilon \leftarrow 0.1$**;**

while($||\nabla f(\mathbf{x})|| > \varepsilon$**)**

$\mathbf{p} \leftarrow \nabla f(\mathbf{x})$**;**

$\alpha \leftarrow$**wolfeLineSearch(**$\mathbf{x}$**,**$\mathbf{p}$**);**

$\mathbf{x} \leftarrow \mathbf{x} - \alpha\mathbf{p}$**;**

end;

objective function $f(\mathbf{x}_k + \mathbf{p}) \approx f_k + \mathbf{p}^T\nabla f_k + \frac{1}{2}\mathbf{p}^T\nabla^2 f_k\mathbf{p}$. Setting the derivative of the Taylor approximation to zero, one gets the step direction for Newton's Method.

From the Newton's Method algorithm Algorithm 3.4, we can see that the step length, alpha, is not dependent on any condition as Newton's Method has natural step-length consideration. However, one of the big problems is that the hessian must be positive definite at all times, but there are methods around this that approximate positive definite hessians. These are called Quasi-Newton methods. Also calculating a good hessian can be computationally demanding, and Newton's Method is dependent on the objective function being close to the second order Taylor approximation.

Algorithm 3.4 Newton's Method

Set $\mathbf{x}$ = current x position;

$\varepsilon \leftarrow 0.1$**;**

$\alpha \leftarrow 1$;

while($||\nabla f(\mathbf{x})|| > \varepsilon$**)**

$\mathbf{x} \leftarrow \mathbf{x} - \alpha(\nabla^2 f(\mathbf{x})^{-1}\nabla f(\mathbf{x}))$**;**

end;

3.3.4. The BFGS Method

The BFGS Method is called a Quasi-Newton method, as it approximates the Hessian $\nabla^2 f_k$ through an iterative matrix called $\mathbf{B}_k$. The step direction is defined as $\mathbf{p}_k = -(\mathbf{B}_k)^{-1}\nabla f_k$, and $\mathbf{B}_k$ is updated each iteration with the following equation called

the BFGS:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \tag{3.3}$$

Where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$.

The derivation of this equation is given in the appendix section A.1.

The big advantage this method has is that it is not dependent on computing the Hessian. In implementation one must choose a good initial approximation for the Hessian, $\mathbf{B}_0$, as well as setting conditions on α, like the Wolfe Conditions. Algorithm 3.5 shows an algorithm using the BFGS method, where $\mathbf{B}_0$ is simply set as an I matrix.

Algorithm 3.5 The BFGS Method

Set $\mathbf{x}_k \leftarrow$ current x position, $\mathbf{B} \leftarrow 0.5\mathbf{I}$;

$\varepsilon \leftarrow 0.1$**;**

$k \leftarrow 0$;

while($||\nabla f(\mathbf{x})|| > \varepsilon$**)**

$\mathbf{p} \leftarrow \mathbf{B}^{-1} \nabla f(\mathbf{x}_k)$

$\alpha \leftarrow$**wolfeLineSearch(**x,p**);**

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \mathbf{p}$**;**

$\mathbf{y} \leftarrow \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$**;**

$\mathbf{s} \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$**;**

$\mathbf{B} \leftarrow \mathbf{B} + \frac{\mathbf{y}\mathbf{y}^T}{\mathbf{y}^T\mathbf{s}} - \frac{(\mathbf{Bs})(\mathbf{Bs})^T}{\mathbf{s}^T\mathbf{Bs}}$**;**

$k \leftarrow k + 1$;

end;

3.4. Non-Derivative Methods

Non-derivative methods are optimization methods that are not dependent on ∇f. For this project only the Nelder Mead Method is implemented.

3.4.1. The Nelder Mead Method

The Nelder Mead Method is a simplex based method where n+1 points form a simplex, (a triangle if n = 2), in the solution space, and the simplex will expand

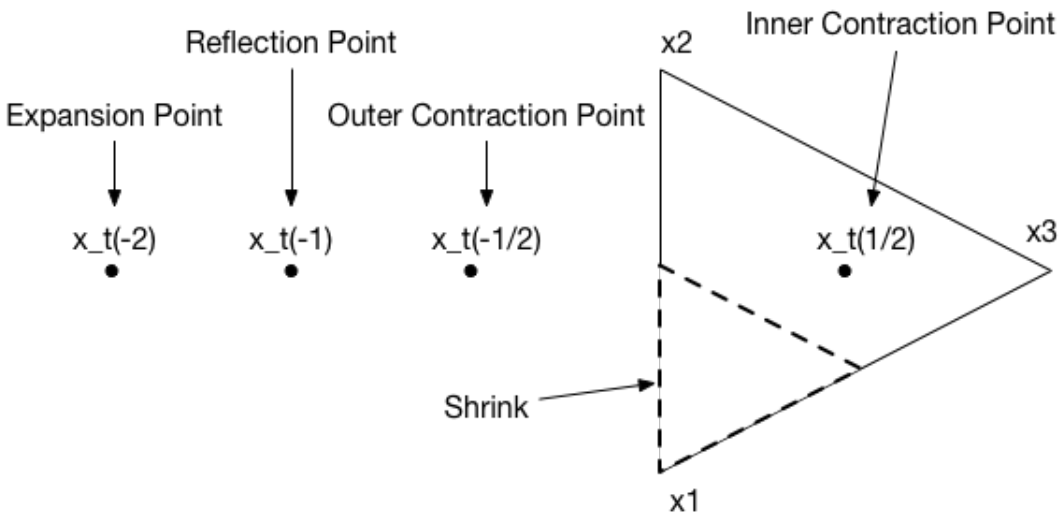


Figure 3.1.: Nelder Mead points illustrated

or shrink towards a local optimum. It decides how the simplex traverses the solution space by sorting the points after objective function values for each point, and changing the worst point for better ones. For each iteration it computes a reflection point, a point that is a reflection of the worst point along the simplex, and checks if this point is better or worse than existing points. If better than the current best, it will try to expand along the line formed between the reflection point and the best point. If the reflection point is worse it will try two contraction points, an inner and outer, that are closer to the simplex. If these are worse than the worst point, the algorithm ultimately shrinks the simplex towards the current best point if it cannot find a better point. Fig. 3.1 shows how these points are represented in a n=2 problem.

The algorithm works with the following:

- Points defined as $x \epsilon \mathbb{R}^n$ where n is the number of states in the system.
- An objective function $f(x), \epsilon \mathbb{R}^n \rightarrow \mathbb{R}$ that returns the value we want to be minimized.
- A sorted vector $\mathbf{x} = [x_1 x_2 .. x_{n+1}]$ of $n+1$ points, where $\mathbf{x}$ is sorted so that $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$.
- The centroid of the vector $\mathbf{x}$ defined as $\mathbf{x}_c = \frac{1}{n} \sum_{i=2}^{n+1} \mathbf{x}_i$.
- A point candidate function defined as $\mathbf{x}_t(t) = \mathbf{x}_c + t(\mathbf{x}_1 - \mathbf{x}_c)$.
  - Fig. 3.1 shows how the function $\mathbf{x}_t(t)$ is related to the discussed candidate points. The different points are different values of t.
- A tolerance calculation that uses the area of the simplex to determine if the algorithm is close enough to a optimal point.

- When $n = 2$, the area is a triangle and Heron's Formula is used. The formula is as follows:
  - Given the three lengths of a triangle,a, b and c, the Area of the triangle can be calculated with

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{3.4}$$

where $s = \frac{a+b+c}{2}$.

Algorithm 3.6 The Nelder Mead Method

```
Start with n+1 points and add them to array x.


while(simpArea >= simpAreaTol)
        Sort x after so that f(x_1) <= f(x_2) <= .. <= f(x_n+1)
        Calculate new simplex area, simpArea.
        Calculate new centroid x_c from x values.
        Calculate reflection point x_r = x_t(-1);
        if(f(x_1) <= f(x_r) && f(x_r) < f(x_n-1))
                Update worst point x_n+1 with reflection point x_r;
        end

        if f(x_r) < f(x_1)
                Calculate expansion point x_g = x_t(-2)
                if (f(x_g) < f(x_r)
                        Update worst point x_n+1 with expansion point x_g
                else
                        Update worst point x_n+1 with reflection point x_r
                end
        end

        didIContract = false
        if f(x_r) >= f(x_n)
                if f(x_r) < f(x_n+1)
                        Calculate outside contraction point x_uc = x_t(-1/2)
                        if f(x_uc) <= f(x_r)
                                Update worst point x_n+1 with outside contraction point x_uc
                                didIContract = true
                        end
                else if f(x_n+1) <= f(x_r)
                        Calculate inside contraction point x_ic = x_t(1/2)
                        if f(x_ic) < f(x_n+1)
                                Update worst point x_n+1 with inside contraction point x_ic
                                didIContract = true
                        end
                end
                if didIContract == false
                        for i = 2,3...n+1
                                replace x_i with (1/2)*(x_1 + x_i)
                        end
                end
        end
end
```

4. Search and Optimization Algorithms

4.1. Overview

This chapter presents two algorithms that both are variants of a search and optimization algorithm designed to find the azimuth and elevation position of the local antenna that gives maximum received signal level, or maximum RSSI, assuming the remote antenna is standing still.

4.2. The Lobe Search

The lobe search is a static search where the antenna will first do a sweep along the azimuth $\pm 40°$ from the starting point while saving the received power measurements. When done sweeping, it will move to the point with the highest measured value. The antenna will repeat the same sweep along the elevation, only that the starting point is the maximum point from the azimuth sweep. The lobe search is done when the antenna has moved to the highest measured value from the elevation sweep. What is important is that the remote antenna does not move when the local antenna does the lobe search, and vice versa.

The data from the lobe search sweeps are used to approximate gradients for the objective function, and is important for gradient optimization algorithms.

The lobe search is designed to at least guarantee that the end point of the search is in the main lobe, and at best find the optimal point. However, there are several downsides with this search that should be compensated for. The downsides are as follows:

- The algorithm is time-consuming. Also, a lot of the data gathered is simply not used.
- The initial azimuth direction search could start in a "valley", where it would be difficult to determine where the lobes are located.
- Noise can introduce single measurement false maximum value.

The first problem can be compensated for using optimization algorithms that do not need gradient information, like the Nelder Mead method, as the method only

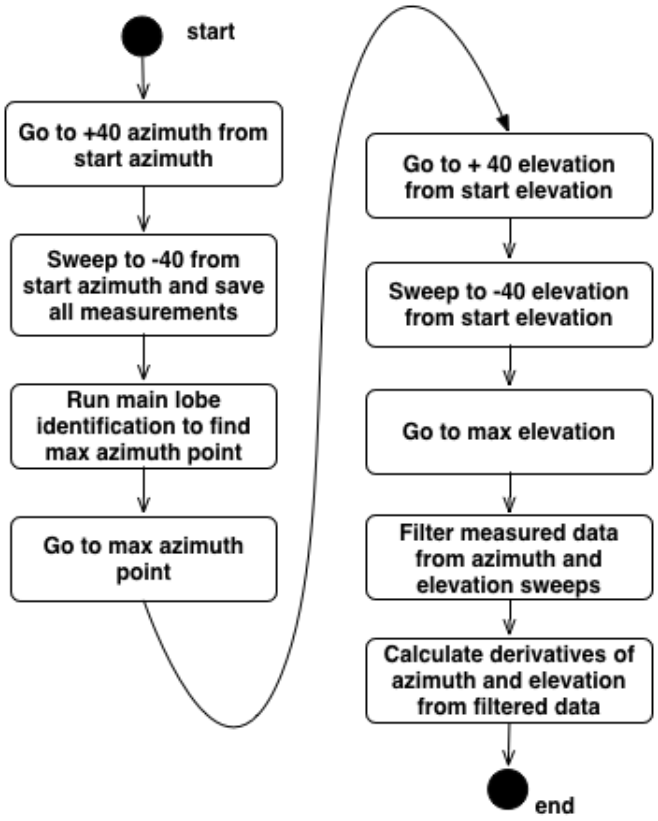


Figure 4.1.: Lobe search flowchart

needs to measure point data. To guarantee convergence towards the main lobe, one of the starting points should be a value along the main lobe that is higher than the maximum side lobe value. These requirements introduce faster sweeps limited only by the sampling of the sensor.

The second problem is solved by using a lobe identification algorithm that uses known properties of the radiation pattern to determine the real main lobe, even if side lobes are measured with higher or similar values. This algorithm is presented in the next section.

The third problem can be compensated for by using a smoothing filter on the measured data before running the optimization algorithm.

4.2.1. Main Lobe Identification Algorithm

The main lobe identification algorithm is an algorithm that runs after the first azimuth sweep is done, see Fig. 4.1, to identify the main lobe in cases where several main lobe candidates are measured. The algorithm uses the symmetry property of the side lobes to identify them as well as potential main lobes. It first identifies the

maximum measured point, and defines a neighborhood of RSSI values around this point, as well as a neighborhood of angle degrees. It then looks for other candidate top points by checking if any other maximum points are within the RSSI neighborhood, but not in an existing angle neighborhood. The angle neighborhood is there to ensure the algorithm does not wrongly identify points on the same lobe as maximum points. Three different cases are identified as sufficient for the application and illustrated in Fig. 4.2. All candidate points must be within the RSSI neighborhood. The red boxes indicate areas around a candidate point where there can be no other candidates, to ensure there is only one candidate point per lobe.

The case in Fig. 4.2 (a) is the simplest one, where only one candidate is found. The main lobe maximum point is the candidate point.

The case in Fig. 4.2 (b) is the most complex one, as the main lobe is actually smaller than the side lobes. Only the side lobes are found, but now a new search must be directed between them, and the single top point found there is the main lobe point.

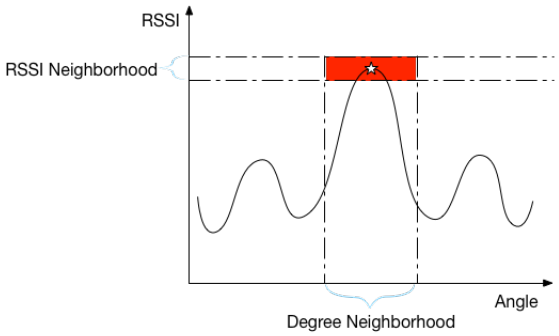
The case in Fig. 4.2 (c) is the rarest case, as it demands the main lobe to be in the same RSSI neighborhood as the side lobes. However, the solution is simple, as the main lobe maximum is the middle candidate.

Fig. 4.3 shows the algorithm in practice, where it correctly identifies the middle lobe as the main lobe from a difficult starting position.

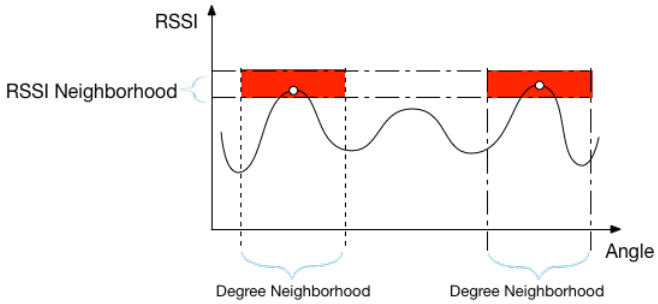
4.3. Algorithm 1: Slow Lobe Search and Gradient Optimization (SLGO)

The algorithms differ in lobe searches by how many measurement samples the lobe search can produce. The more samples, the more time consuming the search is. However, a large amount of samples will build a better model of the radiation pattern, and establish a clearer idea of where the optimal point may reside. This is called a “slow” lobe search compared to a “fast” lobe search where there are much fewer samples but the search is much quicker.

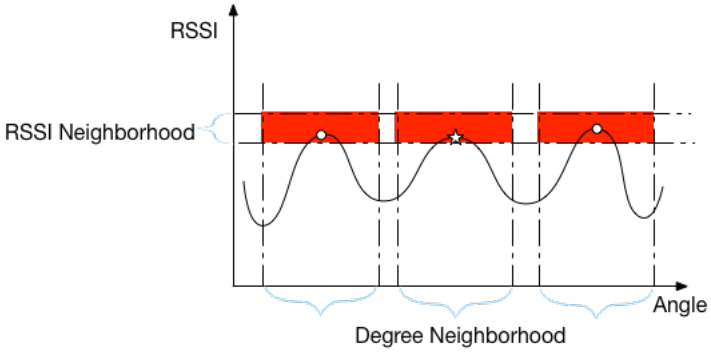
The gradient methods are dependent on approximated derivatives, as well as having a starting position close to the optimal point, or at least on the main lobe to ensure the algorithm does not converge to a side lobe. To build good approximate derivatives, many samples need to be measured. These samples are also filtered using a smoothing filter, as noisy measurements are hard to approximate derivatives from. Another bonus by using a filter on noisy data is that it removes the danger of potential false maximum measurements from the lobe search, and gives a clearer picture of the underlying pattern. Fig. 4.4 shows an example where simulated Gaussian noise causes the lobe-search to return a false maximum point, and where the smoothing filter maximum is closer to the real maximum. When finished approximating the derivates, the algorithm uses these approximations in the gradient algorithm of choice to optimize towards the maximum point.



(a) One point



(b) Two points



(c) Three points

Figure 4.2.: Main lobe identification cases

4.3.1. The Smoothing Filter

The smoothing filter is a zero-phase binomial smoothing filter found using MATLABs "filtfilt" function and tutorial on smoothing filters[9, 10]. The purpose of the smoothing filter is to ensure that the approximation of derivatives calculated has a smooth and continuous data set. A binomial smoothing filter is a repeated application of a weighted moving average filter. A moving average filter is given by the following:

$$y_k = \frac{1}{n}\sum_{i=0}^{n} q_i x_{k-i}$$

Where n is the length of the area one wants to average over around each point, and q is weighting elements for each point. A binomial smoothing filter is repeatedly applying this filter with parameter $n = 2$ and $q = [\frac{1}{2}, -\frac{1}{2}]$ over m times. This gives a new filter with parameters $n_b = nm$ and $q_b = q^m$. This causes the weighting to approximate a Gaussian form, where the weight closest to the real value is the one closest to the current one. The filter presented does not take into consideration

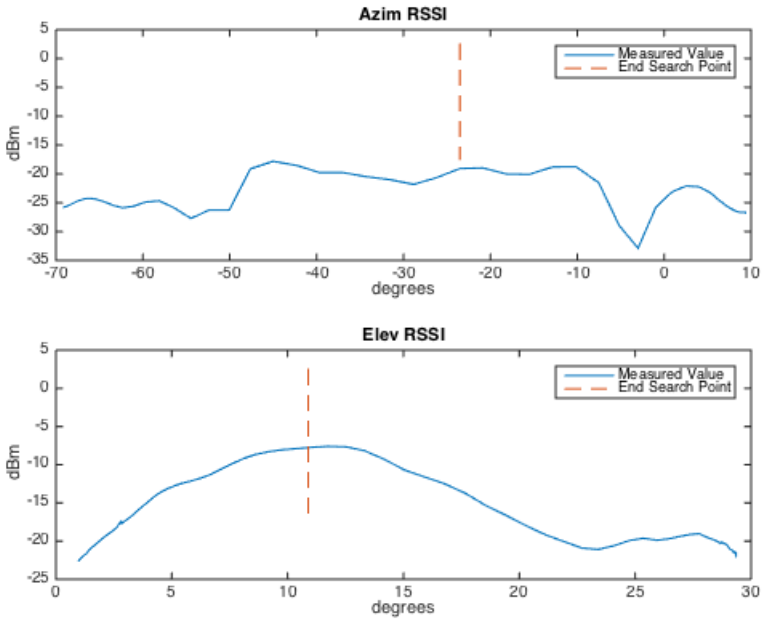


Figure 4.3.: Difficult azimuth search

"future" points, and only weighs along past measurements. This introduces phase shift, so to remove this shift the filtered output is reversed and then filtered again before one last reverse to compensate for the phase shift and to introduce a zero-phase filter. Fig. 4.4 shows the result of a binomial smoothing filter where $m = 4$, used on noisy data from an elevation sweep.

4.3.2. Finite Approximation of Derivatives

The derivatives are approximated through central difference approximation. This means that the algorithm will work with a finite set of derivatives, defined as the following:[12]

$$\frac{\partial f}{\partial \phi}(i) \approx \frac{f(i+dt,c) - f(i-dt,c)}{2dt}$$

$$\frac{\partial f}{\partial \theta}(i) \approx \frac{f(c,i+dt) - f(c,i-dt)}{2dt} \tag{4.1}$$

i is a sample time, dt is the time-step between each sample and c is constant for each i. What this equation means in practice is that the partial difference along the azimuth is approximated from measurements from a sweep along the azimuth,

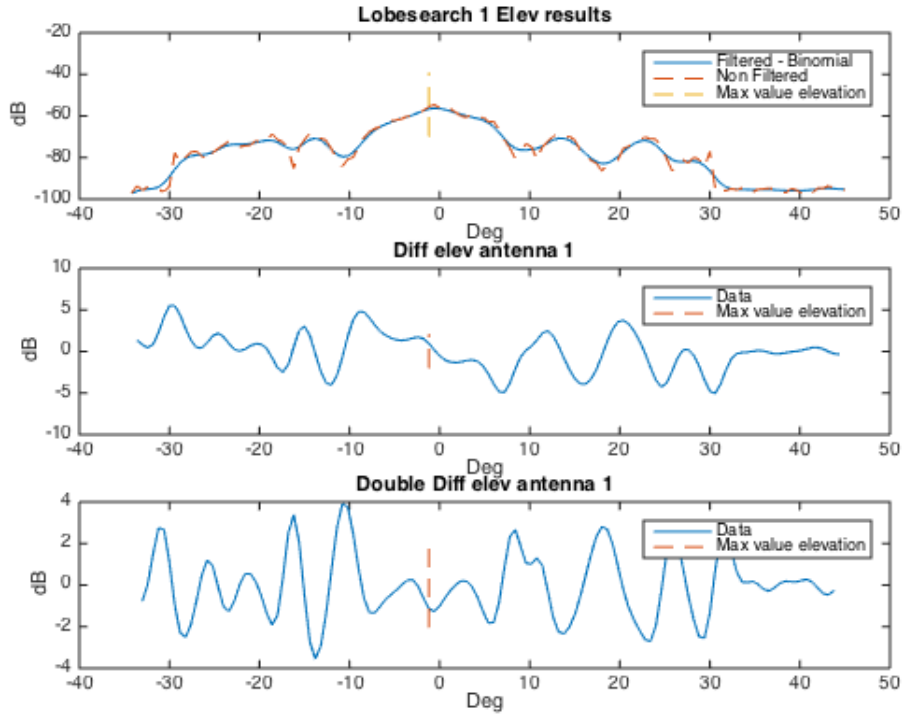


Figure 4.4.: Simulated noisy lobe search with smoothing filter, real maximum is at 0 elevation.

where elevation is constant, and the same is done for a sweep along elevation where azimuth is constant. These derivates are the basis of the gradient:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial \phi} \\ \frac{\partial f}{\partial \theta} \end{bmatrix}$$

4.3.3. Finite Approximation of Double Derivatives

The Hessian $\nabla^2 f$ used by Newton's Method can be approximated to the following:

$$\nabla^2 f \approx \begin{bmatrix} \frac{\partial^2 f}{\partial \phi^2} & 0 \\ 0 & \frac{\partial^2 f}{\partial \theta^2} \end{bmatrix}$$

Where the double derivates are approximated through central difference approxima-

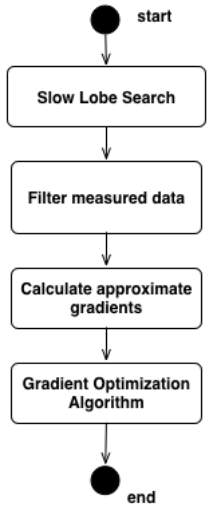
tion of the already approximated derivatives from 4.1:

$$\begin{aligned} \frac{\partial^2 f}{\partial \phi^2} &\approx \frac{\frac{\partial f}{\partial \phi}(i+dt) - \frac{\partial f}{\partial \phi}(i-dt)}{2dt} \\ \frac{\partial^2 f}{\partial \theta^2} &\approx \frac{\frac{\partial f}{\partial \theta}(i+dt) - \frac{\partial f}{\partial \theta}(i-dt)}{2dt} \end{aligned} \tag{4.2}$$

4.3.4. The Algorithm

The algorithm does a slow lobe-search, where slow is defined as slow enough movement to ensure a sample of at least 0.5 resolution in degrees. The measured values are filtered through the smoothing filtered described in section 4.3.1, and then the gradients and double gradients are approximated from the filtered data using 4.1. These approximated gradients are the basis of the gradient optimization algorithm used, depending on whether it is the Steepest Descent, Newton's Method or BFGS that is being implemented.

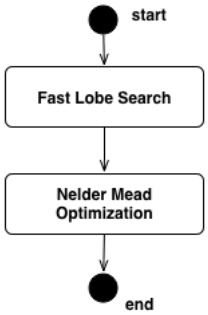
Algorithm 4.1 Slow Lobe Search and Gradient Optimization (SLGO)



4.4. Algorithm 2: Fast Lobe Search and Nelder Mead Optimization (FLND)

This algorithm tries to speed up the lobe-search and reduce the chance of false maxima by simply sampling less through faster movements. The non-gradient optimization algorithm is the Nelder Mead Method, and only requires that one of the points of the initial simplex is on the main lobe. A fast lobe-search should be able to ensure this, and the speed of the search is only limited by the sampling speed, as the resolution can be over 1 degree.

Algorithm 4.2 Fast Lobe and Nelder Mead (FLND)



5. Simulation Environment

5.1. Introduction

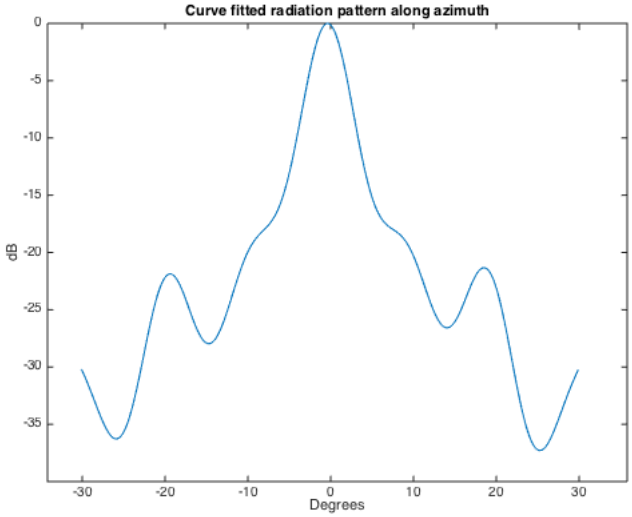
This chapter will introduce how the antenna and simulation is modeled in MATLAB, and how the optimization algorithms are implemented.

5.2. The Antenna Model

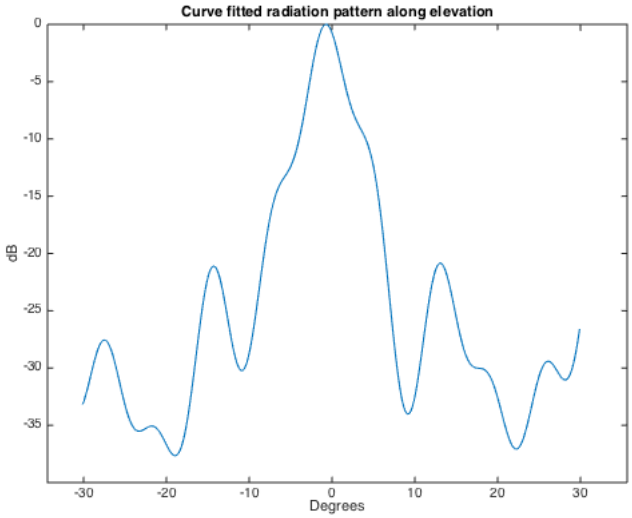
The antenna model is based on principal planes measurements from a high-gain NATO band IV directional antenna. As the exact measurements are considered sensitive, this project will model a sub-set approximation of the actual measurements. Because the project focuses on optimization of an already connected signal, we assume that the work space of this simulation is $\pm 40^{\circ}$ from the optimal position in both elevation and azimuth.

5.2.1. Sub-Section Curve Fitting

To approximate the measurements, as well as to find an equation, MATLABs curve fitting toolbox was used to find a close match. The most important properties of the antenna to simulate is the phases of side lobes and main lobe, especially similarities in maximum and minimum position. The magnitudes are less important. Fig. 5.1 shows the two curve fitted models of the sub sections for azimuth and elevation, and they are modeled as a sum of sines to the 6th degree. The function shown in Fig. 5.1 (a) is noted as $f_{azim}(\phi)$, while the function shown in (b) is noted as $f_{elev}(\theta)$ .The magnitudes are normalized, so the magnitudes show signal loss related to the maximum possible gain in decibels.



(a) Azimuth



(b) Elevation

Figure 5.1.: Curve-fitted radiation patterns

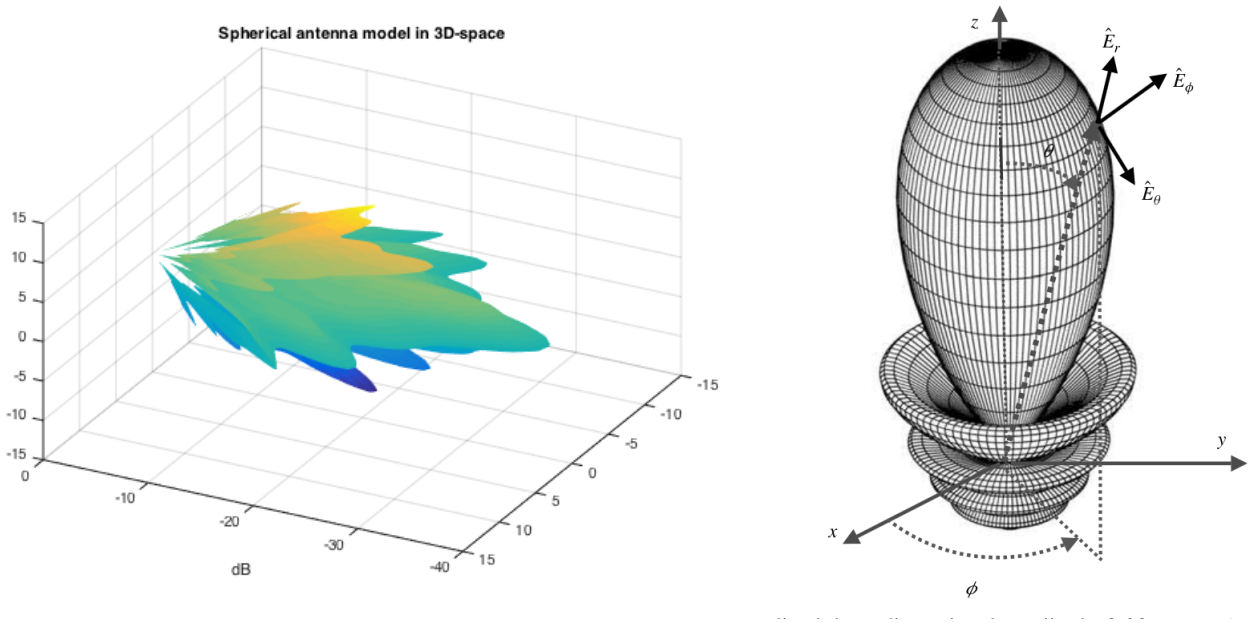
5.2.2. Interpolated 3D Model

To create a $\mathbb{R}^2 \to \mathbb{R}$ function from the principal planes functions $f_{azim}$and f_{elev}, the results from curve-fitting are summed to the following function:

$$G(\phi, \theta) = \frac{f_{azim}(\phi) + f_{elev}(\theta)}{2}$$

Spherical coordinates are used to plot the 3D model of the $G(\phi, \theta)$. These coordinates are found with the following parameterizations $\varphi = \phi$, $\theta' = -90 + \theta$ and $r = G(\phi, \theta) + G_{max}$. G_{max} is the maximum gain of the antenna, and for the simulations is set to 30 dB. Before plotting, these coordinates are turned into cartesian coordinates using the equations given in 2.1. Fig. 5.2 (a) shows a spherical 3D model of the interpolated model in the angular ranges of ±30°from the top point in azimuth and elevation. The G_{max} for this plot was set to the maximum value that ensured that no measurement was negative. The limitations on azimuth and elevation are set to reduce the computational demand of plotting and the simulations will only operate in this range.

Comparing the simulated model with a theoretical model shown in Fig. 5.2 (b), there is a big difference in how the side lobes are modeled. This is the simplest way to interpolate principal planes, but there are more complex ways to do it, like weighting the gains with the angle difference from the principle planes.[16]



(a) Simulated model **(b)** Theoretical model[2]

Figure 5.2.: 3D Radiation patterns

5.3. Optimization Model

The optimization model is a simplification of the total antenna gain model. Because the remote antenna is standing still, the contributions of the remote antenna's position to the received gain can be seen as constants. For the local antenna, the

received power can be modeled through Friis Transmission Equation, and applied to the simulation model will get the following form:

$$P_l = 30dBm + 10\log_{10}(P_r) + G_l(\phi_l, \theta_l) + G_r(\phi_r, \theta_r) + 20\log_{10}(\frac{\lambda}{4\pi d}) + \omega(t) = G_l(\phi_l, \theta_l) + \omega(t) + C$$

Where the index l describes parameters for the local antenna, the index r describes the remote antennas parameters, C is a constant value, where all values from the remote antenna are seen as constant and $\omega(t)$ is white Gaussian noise on the signal in dBm.

The task is to maximize the power received, so the optimization problem will then be defined as:

$$min \ -f(\phi_l, \theta_l) = P_l = G_l(\phi_l, \theta_l) + \omega(t) + C$$

Where the state vector is $x = [\phi \, \theta]^T$ and $\phi = \phi_l$, $\theta = \theta_l$, and gradients are defined using finite approximation discussed in section 4.3.2.

5.4. The Simulation Environment

The simulation environment defines and keeps track of the following parameters:

- The positions of the two antennas (latitude, longitude, altitude)
- The orientation of the two antennas, and start values of these (azimuth, elevation)
- The optimal direction calculated from geographic libraries (optimal_azimuth, optimal_elevation)
- Angular velocity of azimuth and elevation of the two antennas.
- Frequency of wave signal of each antenna.
- Power of transmitted signal in Watt.
- Simulation time-step for each movement of antennas
- A table of received power over time for each antenna

These parameters are defined during initiation, before different functions and scripts are called to ultimately maximize the received power.

5.4.1. Constant Parameter Values

The following parameters are constant for all simulation tests:

- Antenna positions are set in latitude and longitude, and then converted to cartesian coordinates through a mapping toolkit. The coordinates come from Google Earth.
  - Antenna 1: Latitude 59.829687, Longitude 10.411962, Altitude 191 m.
    - These are the coordinates of Kongsberg office with estimated altitude that is height over sea.
  - Antenna 2: Latitude 59.958962, Longitude 59.958962, Altitude 374 m.
    - These are the coordinates of Grefsenkollen Restaurant. This is a typical line of sight Kongsberg has done before with parabolic antennas.
- Frequency of the signal is set to 4.7 GHz.
- Power transmitted for each antenna is 5 W≈+37dBm.
- Time-step for simulation is 0.1 s or 10 Hz.

5.4.2. Simulated Antenna Movement

The antennas are simulated using loops and a constant time step ∇t. For each loop, the antenna orientation is updated with the following formula

$$\mathbf{x}_{k+1} = \mathbf{x_k} + \mathbf{v_k}\nabla t$$

Where $\mathbf{x} = [\phi\ \theta]^T$,$\mathbf{v} = [v_\phi\ v_\theta]^T$, where v_θ is the tilt speed of the antenna, v_ϕ is the pan speed of the antenna and k is time step indexes. Movement is run in loops which exit when desired orientation is measured within tolerance values of $v_\phi \nabla t$, as the pan speed is more likely to be faster than the tilt speed. Note that the velocity is instantaneous, as acceleration is not modeled.

5.5. Received Power Optimization for Two Antennas

The optimization algorithm starts immediately when the two antennas have found each other and finished a handshake protocol to ensure that they can start sharing data. The algorithm is a simplification of the optimization flow-chart designed for the prototype project, see Fig. 5.3, where the antennas decide on either being a Master or Slave antenna, and then taking turns doing the lobe searches described in section 4.2, before taking turns running an optimization algorithm. The combination and parameters for the tests are based on the two algorithms SLGO Algorithm 4.2 and FLND Algorithm 4.1.

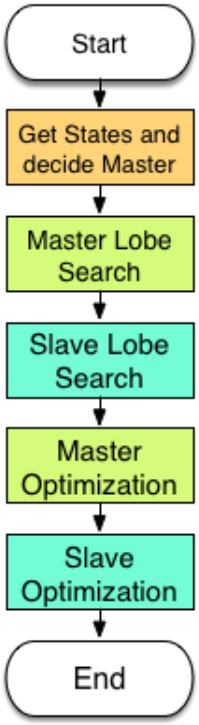


Figure 5.3.: RSSI optimization algorithm

5.5.1. Optimization Algorithms

The different optimization algorithms will all start from the best point found from the lobe search. The lobe search should ensure that the starting point is in close proximity of the optimal point, so that we can assume the local optimal point found from the optimization algorithms will also be the global optimal point.

The following algorithms have been implemented in the simulation environment:

- Nelder Mead Method
- Steepest Descent Method
- Newtons method
- BFGS method

5.5.1.1. Wolfe Conditions Implementation

The Wolfe line-search does many measurements along the given direction to find a step length value that fits the Wolfe Conditions. However, this could be time-consuming when having to move the antenna in small increments, so the implementation instead assumes the measurements from the lobe searches are valid approximations. This speeds up the optimization algorithm as it now no longer has to move

the antenna to check several step-lengths, on the cost of less precision if the step length withholds the Wolfe conditions.

5.5.1.2. Gradient Method Implementation.

The Gradient Method algorithms are dependent on a smooth objective value function, so the noisy data has to run through a zero-phase smoothing filter to ensure no sudden jumps in derivatives so the top point can be more easily identified. The filter used is a binomial smoothing filter with $m = 4$. (See section 4.3.1.)

5.5.1.3. Nelder Mead Algorithm

Nelder Mead is implemented as shown in section 3.4.1 where n = 2, point x_i is defined as x_i = (azimuth_i,elevation_i) and the objective function value is the RSSI values gathered on the measured points. The objective function is then a function that moves the antenna to the given point and returns the RSSI value measured on that point. The tolerance function is Hedron's Triangle Area formula (3.4), and the tolerance is set to 0.05. The initial points are determined by creating a triangle with given distances from the initial point. The distances are called Azimuth Span (AS) and Elevation Span (ES). Fig. 5.4 shows how the two points P1 and P2 are created, where P1 is the point (-AS,ES) from the lobe search point, and P2 is the point (AS,ES) from the lobe search point. This initial simplex is based upon the assumption that the antenna system is much more sensitive to movement along the elevation compared to movement along the azimuth.

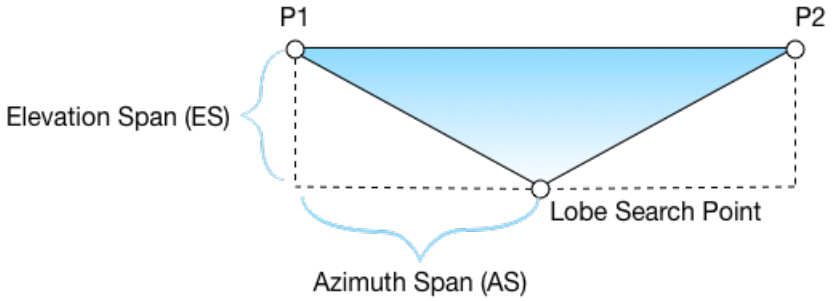


Figure 5.4.: Nelder Mead initial simplex

6. Simulation Results

6.1. Overview

This chapter presents results from the simulations done in MATLAB of two antennas taking turns doing lobe-searches and optimizing towards a position. The results will be presented sequentially from lobe search to optimization algorithms, and results from the lobe search will be considered in the results from the optimization algorithm. Noise is simulated as Gaussian noise with a standard variance given i dBm.

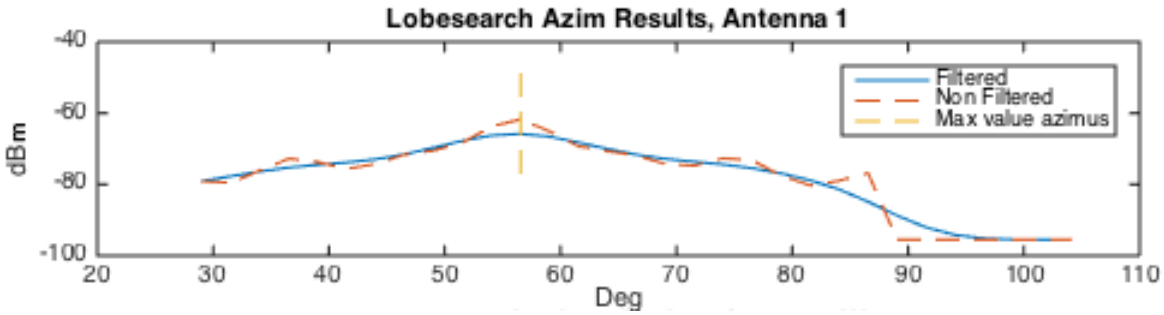
6.2. Lobe Search Results

Lobe searches are dependent on initial orientation and noise. A non-noisy lobe search will by definition find the real top measured value, and will only be restricted by sampling time from the rotation speed. The results are given as graphs with measured value over azimuth or elevation sweep, as well as filtered values and the approximated finite derivative and double derivative. The times used is the time the whole algorithm takes from start to finish including both antennas and both azimuth and elevation. The max speed for azimuth and elevation are taken from the limits for the rotator used, and is 25 deg/s for azimuth and 8 deg/s for elevation.

6.2.1. Speed

Fig. 6.1 shows the difference in how speed affects lobe search results. The trick is to find the best compromise between duration and resolution of data gained to find the optimal position. The speedy results (b) shows that the main lobe can be determined for the Nelder Mead algorithm, while the slow results in (a) manages to build good approximations of the differential and contain many details of the lobes. The time used for the slow lobe search (a) is 62.9 seconds, while the fast lobe search (b) finished the algorithm in 52.7 seconds. The value of 15 deg/s for the SLGO algorithm was fast enough and could compete with time used for the FLND algorithm, while still yielding results that the gradient optimization could use effectively.

(a) Speed 15 deg/s



(b) Speed 25 deg/s

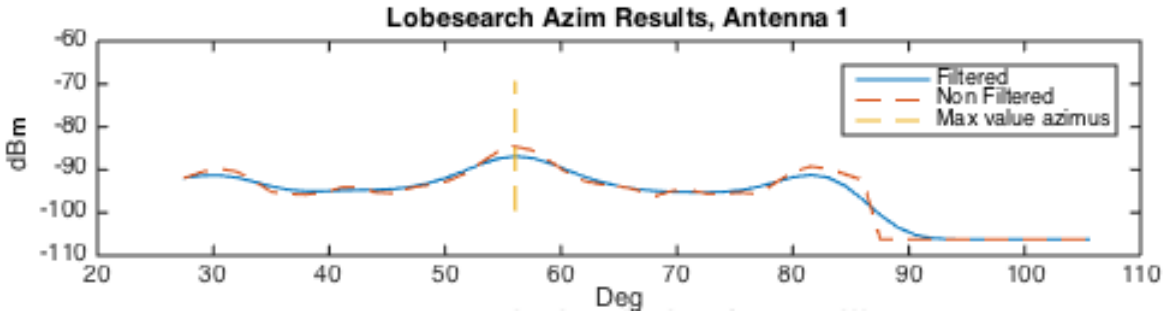
Figure 6.1.: Non-noisy lobe search from position close to optimal

6.2.2. Start Position and Noise

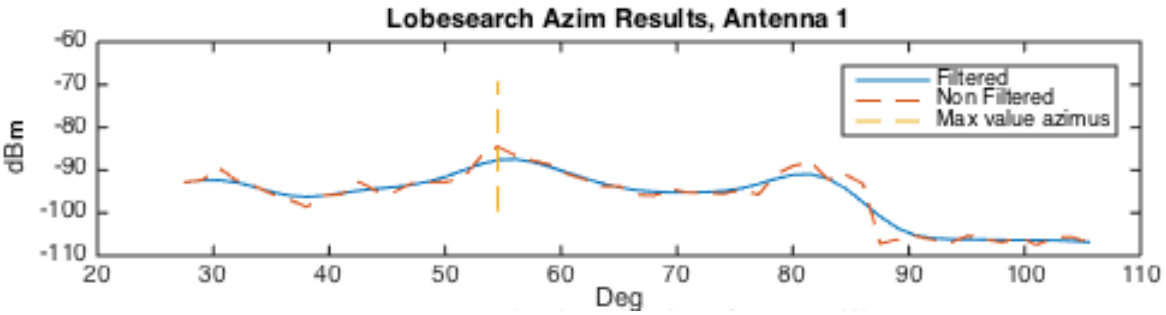
The start position determines the signal strength of the main lobe along the first azimuth sweep. The offset in degrees for the far away position is 10 degrees in azimuth and -20 degrees in elevation. As azimuth is the first angle to be measured in the algorithm, the initial elevation is the parameter that affects the results the most. The further away the start position for the lobe search is from the optimal point, the flatter the measured pattern. This can be seen by comparing the graphs of Fig. 6.1 (a) with Fig. 6.2 (a). The flattening of the pattern causes noise to affect the lobe search algorithm much more when the initial position is further away, as the noise amplitude is constant. The smoothing filter is at work in Fig. 6.2 (b), and one can clearly see a case of a false maximum point arising from the simulated noise. The filtered maximum point is closer to the real maximum value.

6.3. Gradient Method Results for SLGO Algorithm

This section will compare the different gradient methods, Steepest Descent, Newton's Method and the BFGS Method, where they will work from a lobe search with the following parameters:



(a) Non-Noisy



(b) Noisy

Figure 6.2.: Lobe search from position far from optimal and 15 deg/s

- Added zero mean Gaussian noise with 1 dBm standard deviation
- Pan speed of 15 deg/s
- Tilt speed of 8 deg/s
- Starting point that is 10 degree in azimuth direction and -20 degree in elevation from optimal point.

The results are based on the same noisy data, so the start position and filtered data that the gradient searches are based on are the same for each run. The results for this test is presented in Fig. 6.3 and Tab. 6.1. The figures show contour plots of the path the gradient optimization algorithm took. This is zoomed in close to the main lobe. The only difference between these results is the optimization method used, and the parameters that define their individual behavior, like tolerance values and initial B matrix. Note that the end RSSI value is also affected by noise, and it should be considered that the RSSI value could vary with $\pm$1dBm on the same points. There does not seem to be a clear benefit of using the more complex gradient methods BFGS and Newton over the simple Steepest Descent, as the starting point is so close to the optimal position.

	Steepest Descent	Newton's Method	BFGS
Number of Iterations	4	2	2
Diff from optimal azimuth [°]	-0.69119	0.33390	0.96495
Diff from real elevation [°]	-0.33295	0.514578	0.078095
Best RSSI [dBm]	-38.6731	-38.6704	-38.5183
Time Taken [s]	1.9	2.6	2.4

Table 6.1.: Comparison of gradient algorithms from same lobe search

6.4. Nelder Mead Algorithm

The most important parameter for the Nelder Mead algorithm is the points of the initial simplex. The lobe search's goal is to ensure that one of the initial points is on the main lobe, and also is on a measured value higher than the highest side lobe. The other two points must be chosen so the algorithm can converge towards the optimal point and not terminate early. However, they should not be too large that it takes too long for the algorithm to converge towards the optimal point.

	AS=2, ES=1	AS=5, ES=3	AS=10, ES=5
Number of iterations	4	11	10
Diff from optimal azimuth [°]	4	-0.8	-0.8
Diff from real elevation [°]	-0.1	0.6991	-0.9
Best RSSI [dBm]	-42.2661	-39.077	-38.141
Total Time Taken [s]	13.5	23.6	50.4

Table 6.2.: Different initial simplex results for Nelder Mead optimization

6.4.1. Initial Simplex

The initial simplex is defined by the sizes Elevation Span (ES) and Azimuth Span (AS), discussed in section 5.5.1.3. Three different initial simplex sizes are tested, and results presented in Fig. 6.4 and Tab. 6.2. Not surprisingly, a larger initial simplex is much more time consuming. However this does reward slightly better results, as the algorithm tests more points. When the initial simplex is too small, the algorithm does not get to explore many sample points before terminating. This causes worse results compared to larger initial simplexes.

6.5. Comparison Between SLGO and FLND

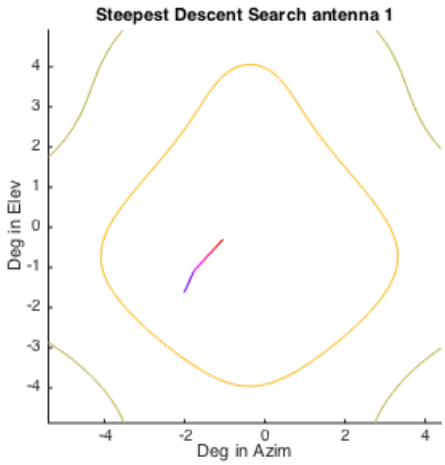
The comparison will be based on the following parameters:

- Common parameters:
  - Starting point is 10 degree in azimuth direction and -20 degree in elevation direction from optimal point.
  - 1 dB standard deviation zero mean added Gaussian noise.
- Slow Lobe Search - Gradient Optimization (SLGO)
  - Gradient method: BFGS
  - Lobe Search Speeds:
    - Pan Speed = 15 deg/s
    - Tilt Speed = 8 deg/s
- Fast Lobe Search - Nelder Mead (FLND)
  - Nelder Mead Initial Simplex:
    - Azimuth Span = 5
    - Elevation Span = 3
  - Lobe Search Speeds:
    - Pan Speed = 25 deg/s
    - Tilt Speed = 8 deg/s

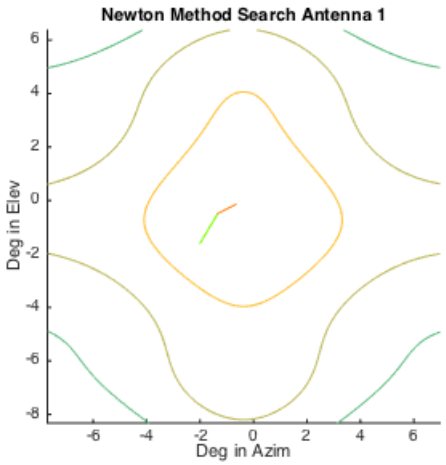
	Antenna 1		Antenna 2	
	FLND	SLGO	FLND	SLGO
Difference from optimal azimuth [°]	0.000	-0.785	0.8	-0.900
Difference from optimal elevation [°]	0.7	-0.124	0.69911	0.301
Best RSSI [dBm]	-39.0971	-37.9277	-38.9877	-38.2337
Time Taken [s]	75.9	64.6	75.9	64.6

Table 6.3.: Comparison between FLND and SLGO results

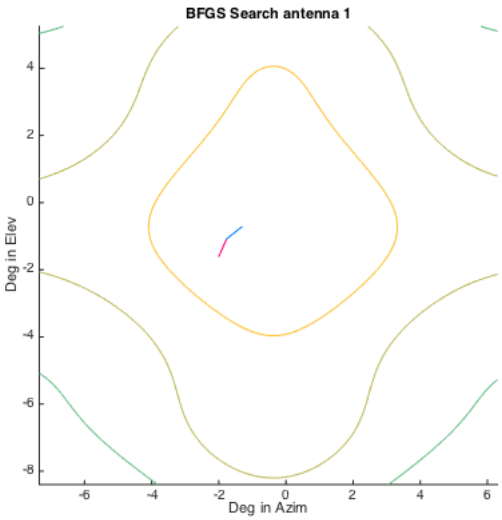
Tab. 6.3 shows that both algorithms perform very close when looking at the end RSSI values. The big difference is the total time taken of around 11 seconds, where SLGO is the fastest. What should be analyzed here are statistical properties introduced by the noise. How well each algorithm performs over several runs, and how sensitive each algorithm is to changes in speed, start position and noise level.



(a) Steepest Descent

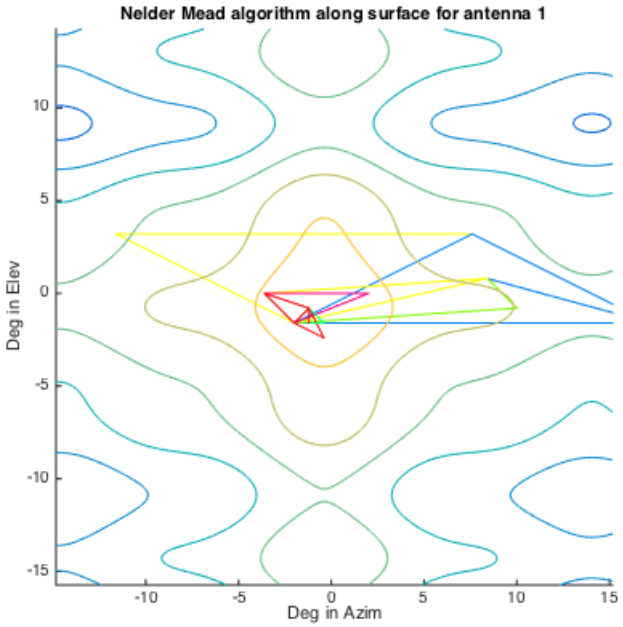


(b) Newton's Method

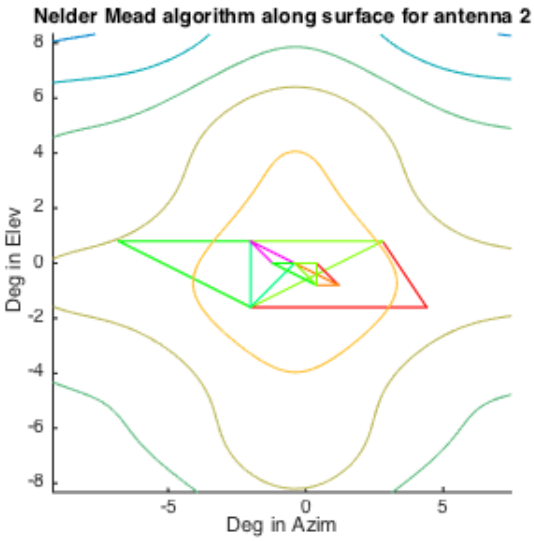


(c) BFGS Method

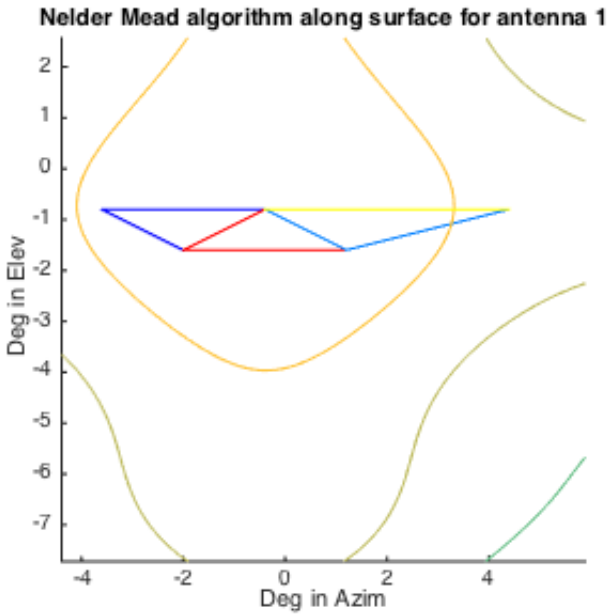
Figure 6.3.: Optimization path for gradient algorithms



(a) AS=10, ES=5



(b) AS=5, ES=3



(c) AS=2, ES=1

Figure 6.4.: Different initial simplex Nelder Mead optimization

7. Real Life Test Environment

7.1. Overview

This chapter will introduce the different components of the real life test, and what algorithms were tested. The real-life test was done in an Radio Frequency (RF) anechoic chamber in which two high-gain parabolic antennas were placed, one statically mounted high on a high point and one mounted to a pan-tilt rotator. The static antenna was connected to a high frequency signal generator, while the antenna on the rotator was connected to a power sensor. The goal of the rotator antenna was to be able to find the best orientation to maximize the signal gain sent from the static antenna.

7.2. Hardware

The test used the following equipment:

- 2x Comrod High Gain Band 4 Parabolic Antennas
- 1x Rohde & Schwarz SMB 100 A Signal Generator
- 1x Rohde & Schwarz NRP-Z21 Power Sensor
- 1x Moog QPT-90 Pan Tilt Rotator

7.2.1. Comrod High Gain Band 4 Parabolic Antenna - SHF4450P08

The Comrod high gain antenna operates in a frequency range of 4.4-5.0 GHz, with a beam width of $5°@\pm 3$dB. This is a typical highly directional parabolic antenna, and will lose over half the potential power if the antenna is oriented over 5° from the optimal point. Fig. 7.1 shows the documented radiation pattern from the antenna data-sheet along the azimuth principle plane. Fig. 7.2 shows the gain curve over the frequency range. The parabola diameter is 0.77 m. The data sheet for the antenna is embedded in section A.3 in the appendix.

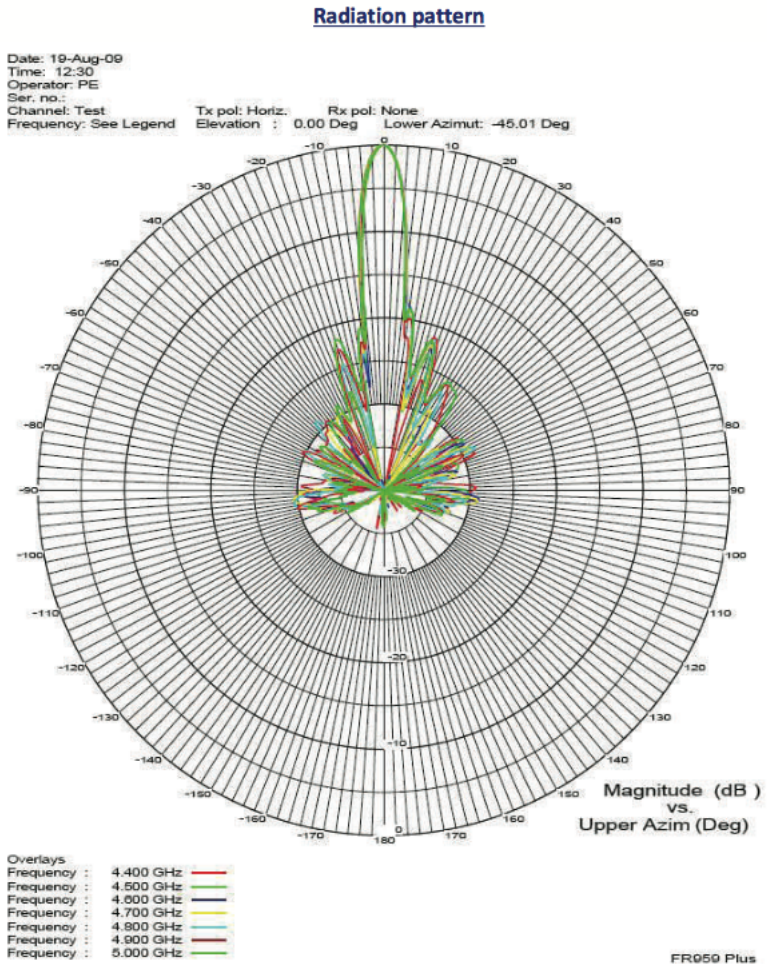


Figure 7.1.: SHF4450P08 Radiation Pattern[6]

7.2.2. Rohde & Schwarz Signal Generator 100 SMB A

The signal generator delivers signals in frequencies from 9 kHz to 6 GHz, and levels -30 dBm to 30 dBm. The RF signals are sent from a standard N-connector through a coaxial cable.[14]

7.2.3. Rohde & Schwarz Power Sensor NRP-721

The power sensor is a continuous averaging power sensor. It supports a level measurement range of -67 dBm to 23 dBm, as well as a frequency range of 10 MHz to 18 GHz. It is connected to a computer with a USB. [14]

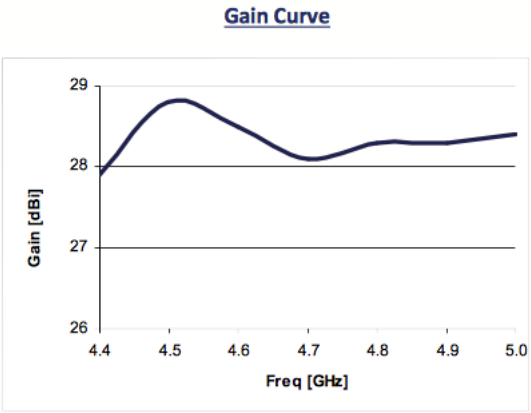


Figure 7.2.: SHF4450P08 Gain Curve[6]

Figure 7.3.: Rohde & Schawrz Signal Generator 100 SMB A

7.2.4. Moog QPT-90 Pan Tilt Rotator

The QPT-90 is a pan-tilt rotator capable of precise movement with 0.01° resolution with the help of optical encoders in both pan and tilt. It operates on pan speeds between $0.005° - 25°/sec$ and tilt speeds between $0.005° - 8°/sec$. It is controlled through an RS-232 serial interface using the Moog Quickset protocol. It weighs 34 kg, and has a load area of 39,9x20,9 cm. The data sheet for the rotator is embedded in section A.3 in the appendix.

7.3. Software

7.3.1. Overview

This section will detail the software implemented for this test. The software is a heavily modified and stripped version of the software used on the summer prototype

Figure 7.4.: Rohde & Schwarz Power Sensor NRP-721

Figure 7.5.: QPT-90 Moog Sentry

(see section 1.2.2). The following modules are completely new for this project:

- A driver for the Moog rotator, which includes an implementation of the Moog Quickset Protocol. This driver also receives orientation data from the Moog rotator as a basis for the measured orientation data.
- An interface for the NRP-721 driver, so the program can read sensor measurements.
- Implementation of the Nelder Mead and Steepest Descent optimization algorithms using the new drivers and sensors.

The following modules are used and improved from the summer prototype:

- An improved lobe-search state machine to better utilize how the new servo driver works.
- Created shortcuts through the existing state machines to reflect the fact that only one antenna is connected to a rotator.

- Updated the thread handler and shared databases to utilize the new NRP and Moog modules.

The following modules from the summer prototype are NOT used:

- Anything that has to do with the IMU. For this project, the orientation data from the Moog rotator is used instead. This is due to simplicity as there is only one rotating antenna, and no algorithms that are dependent on orientation relative to Earth.
- The networking thread. There is only one instance of the system running, so there is no need to communicate.

7.3.2. Module Overview

The following sections will go into detail about the code that is used in the project. Fig. 7.6 and Fig. 7.7 show Unified Modeling Language (UML) class diagrams that show how the different classes are related. The classes presented are stripped down to the essential variables and methods.

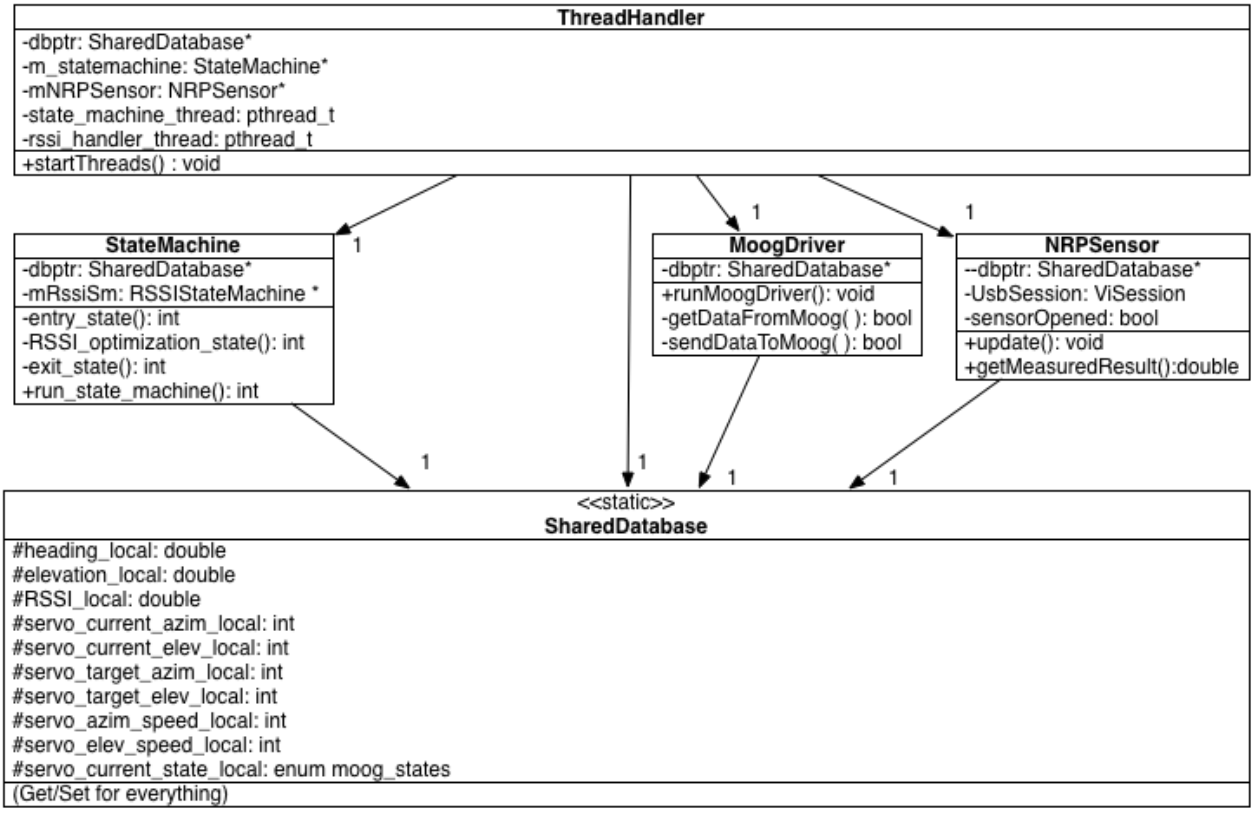


Figure 7.6.: ThreadHandler class diagram

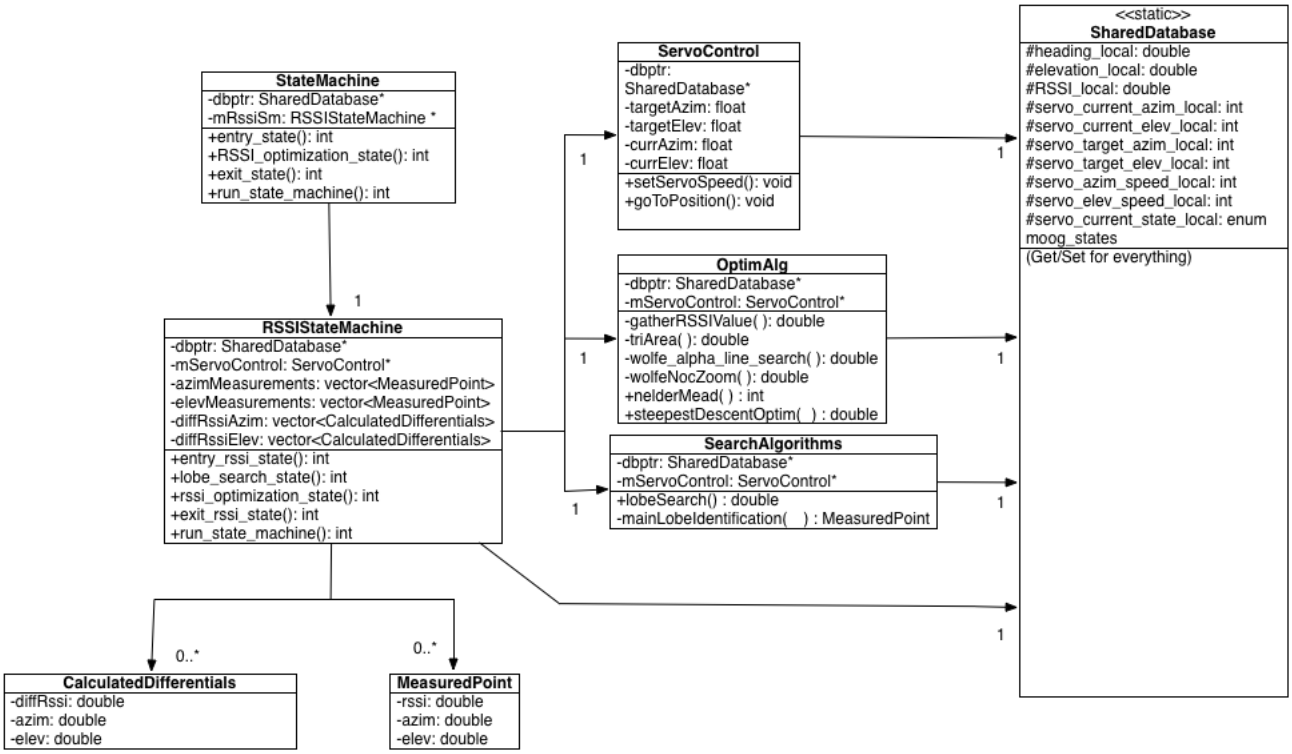


Figure 7.7.: RSSIStateMachine class diagram

7.3.2.1. ThreadHandler

This software is multithreaded, where the ThreadHandler class starts the three main threads, a sensor thread, a state machine thread and a thread that interacts with the Moog QPT-90 called the MoogDriver. In the summer prototype there is also a thread for the IMU sensors and a networking thread for communications between the two systems, but they are excluded. The main thread is the state machine thread, while the two other threads are mostly asynchronous communication with the sensors and rotator hardware. The ThreadHandler starts all three threads from the runThreads() function, and ensures that the threads are kept alive. There is no system for graceful degradation or restart of crashed or failed threads. ThreadHandler also implements the most important class for threaded communication, the SharedDatabase, and sends a pointer for this database to every thread it creates.

7.3.2.2. SharedDatabase

The SharedDatabase is a class that holds all the information the different classes want to share with each other, protected by mutexes. The database is much larger than Fig. 7.6 presents, but the values presented are the only values used for the tests. ThreadHandler creates a static implementation of this database, and passes a pointer

to the database to all threads and classes that need access. Other classes that already own a copy of the pointer pass their pointer if they create a class that needs access. The SharedDatabase is designed to only hold the most recent readings of the various variables, so if one wants historical data one must implement a function that saves recent data from the SharedDatabase. One example on how the SharedDatabase is used is how ServoControl communicates with the MoogDriver. All the functions of the ServoControl that are designated to the Moog QPT-90 are ultimately written to the SharedDatabase variables servo_target_azim and servo_target_elev. The MoogDriver will continuously read these values and check if they have been updated since last check. If a new value is found, it will send the new values to the QPT-90.

7.3.2.3. NRPSensor

The NRPSensor class is a simple thread that continuously reads from the NRPSensor using an installed NRP-721 driver and library. Most of the code is taken from a developer example code, and has the unfortunate property of blocking the reading until measurements are ready from the sensor. If the received levels are poor, (under -30 dBm), the driver can block the thread for several seconds. This causes problems as the QPT-90 has no synchronization with the NRPSensor, and will keep on moving even if measurements are old. However, on levels higher than -30 dBm the frequency of readings are around 10 Hz, the same as the frequency of Moog readings.

7.3.2.4. MoogDriver

The MoogDriver is implemented as a thread of its own because the Quickset Protocol demands periodic "Get Status" messages from the computer for the QPT-90 to stay alive. The QPT-90 also returns the current pan/tilt position as a response for each "Get Status" message, and this position is used as reference for the antennas azimuth and elevation orientation. A system with two antennas should of course use an IMU with GPS to relate its position relative to Earth, but for this test the relative position to the QPT-90 is enough. The driver runs the state machine shown in Fig. 7.8 in a loop of 10 Hz (the maximum frequency recommended by the Quickset Protocol), where the driver either sends a "Get Status" message or a "Move To" message. The DRIVER_INIT state is an initiation state where the driver will send repeatedly "Get Status" messages until the QPT-90 responds. The QPT-90 has an auto-baud feature where it will not respond until it gets a set amount of data so it can determine the baud rate from the sender. When the Driver receives a valid response from the QPT-90, it moves to SEND_STATUS where it will repeatedly send and receive status messages from the QPT-90 and update the SharedDatabase with current orientation of the QPT-90. It also checks the SharedDatabase for changes in target azimuth/elevation, and changes state to SEND_MOVE if it detects a change. From the SEND_MOVE state, there Driver will repeatedly send "Move To" messages with the new target until an acknowledgment for the MOVE_TO message has been

received, or a timeout has been calculated. It then returns to SEND_STATUS. Outside of the state machine it also checks for new speed settings it can send over. No error handling for error messages from the QPT-90 has been implemented.

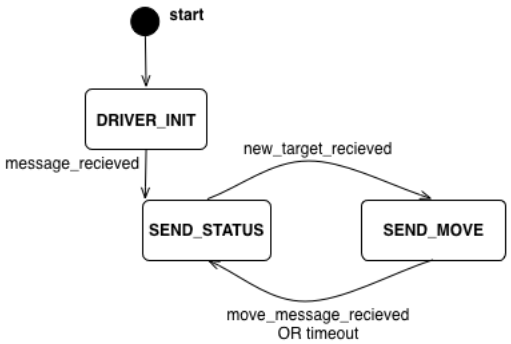


Figure 7.8.: MoogDriver state machine

7.3.2.5. StateMachine

For this project the StateMachine class is pretty stripped. For the summer project it handled the searching algorithms and states that are relevant before the antennas find each other and start communicating. As this project only focuses on the optimization aspect, the StateMachine now only creates the ServoDriver class, sets the starting point for the algorithm and proceeds directly to the RSSIStateMachine

7.3.2.6. ServoControl

The ServoControl is the interface between the MoogDriver and the rest of the program. It contains functions that convert a standard float notation of degrees to how the Moog QPT-90 represents degrees. For example the degrees 45.25° will be converted to the integer 4525 before written to the MoogDriver. ServoControl has no direct connection to the MoogDriver thread, but communicates indirectly through the SharedDatabase.

7.3.2.7. RSSIStateMachine

The RSSIStateMachine is a state machine that handles the RSSI optimization phase. The implemented version of the state machine is very simple and sequential, as seen in Fig. 7.9. The version from the summer prototype had much more states that considered that there where two antennas that could rotate, and the state machine ensured that each antenna stood still while the other either did a lobe search or RSSI optimization. From Fig. 7.7 we can see that the RSSIStateMachine class calls implementations of the lobe search and RSSI optimization algorithms

Nelder Mead and Steepest Descent. The class also saves the measurements from the lobe search, and handles the filtering and derivative calculations before passing the saved values as arguments to the Steepest Descent search. Values are saved in vectors of the class MeasuredPoint, and calculated derivatives are saved in vectors of the class CalculatedDifferentials. These classes are simply storage containers. The filter implemented is a translation of MATLABs filtfilt function by combining Chen Yangquan's implementation of filtfilt in C with Jan Simon's filtM faster filter for MATLAB, written in C as well[15, 21]. The calculations of derivatives are done using the approximate finite derivative equations given in 4.1.

The two algorithms FLND and SLGO are defined by compilation flags, where the parameters for each algorithm is set using these flags.

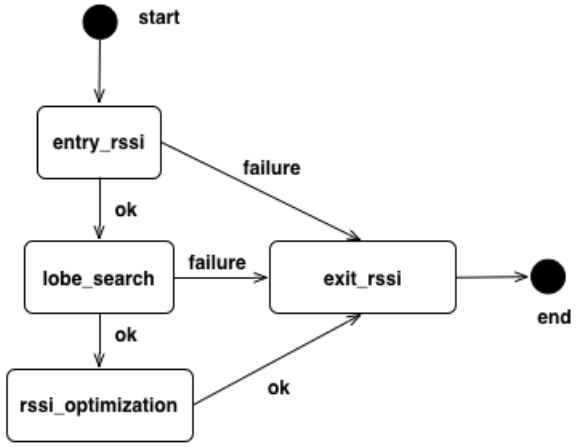


Figure 7.9.: RSSI state machine

7.3.2.8. SearchAlgorithms

The SearchAlgorithms class is a collection of different search algorithms such as random search and pattern search. For this project only lobe search is relevant. Fig. 7.10 shows the simple sequential search, where the algorithm saves measured points during the AZIM_SWEEP and ELEV_SWEEP states. These are sweeps from top limit to bottom limit to ensure continuous measurements. One measurement is an instance of the MeasuredPoint class, where the azim and elevation are read from the tilt and pan position from the MoogDriver and RSSI is the power level in dBm read from the NRPSensor. Measurements are saved in two vectors of MeasuredPoints, azimMeasurements and elevMeasurements, one for each sweep. The LobeSearch algorithm also implements the main lobe identification algorithm from Fig. 4.2 after the AZIM_SWEEP to determine the maximum point for GO_TO_maximum_AZIM. For the main lobe identification algorithms the parameters sideLobeNeighbourhood is set to 10°, while the levelThreshold is set to 1 dBm. This lobe search algorithm differs slightly from the lobe search presented in Fig. 4.1 as the filtering and

derivative calculations are done in the RSSIStateMachine state lobe_search after the lobeSearch function from SearchAlgorithms is done. Due to physical limitations of the system, the lobe search along elevation is always between $0 - 30°$, and not relative to the starting elevation. These restrictions are discussed in section 7.4.3.

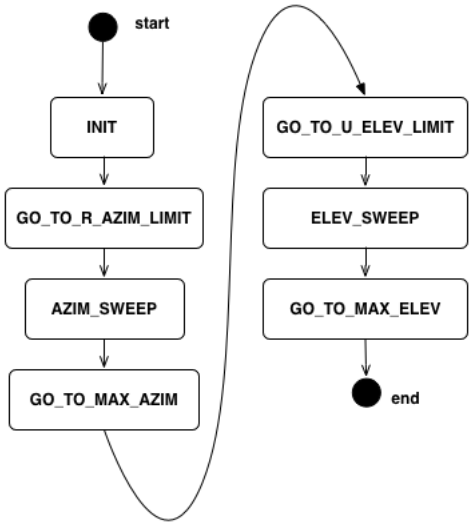


Figure 7.10.: Lobe search state machine

7.3.2.9. OptimAlg

OptimAlg is similar to SearchAlgorithms, only for optimization algorithms. These are the algorithms called during the RSSI_optimization state of RSSIStatemachine, and contains implementations of the Nelder Mead algorithm and steepest descent method. These algorithms use the Eigen C++ library to help with the linear algebra[1]. Nelder Mead is a pretty straight-forward implementation of the algorithm Algorithm 3.6, where the objective function $f(x)$ is implemented as gatherRSSIValue, which takes a position $x = \begin{bmatrix} azim \\ elev \end{bmatrix}$ and moves the QPT-90 to that position. When done moving, the function returns the RSSI power level measured from NRPSensor at that point. The initial simplex is created from theFig. 5.4, where AS is $\pm 5°$ and ES is $+3°$. This ensures a triangle that at least covers a decent area of the main lobe, as the beam-width is $5°$, see section 7.2.1. The triArea function is an implementation of Heron's Formula, see 3.4, and the algorithm terminates when this area is smaller than 0.05.

The SteepestDescent is more complex as the wolfe_alpha_line_search is implemented as a search through measured vectors instead of new measurements. This ensures that the algorithm does not need to wait for the antenna to gather a new

[1] http://eigen.tuxfamily.org/index.php?title=Main_Page

measurement for each time the objective function needs to be compared. This algorithm terminates when the norm of the differential vector for the current point is smaller then 0.5. $|| \begin{bmatrix} \frac{df(x_k)}{x_{1k}} \\ \frac{d(fx_k)}{x_{2k}} \end{bmatrix} || < 0.5$, where $x_{1k} = azim_k$, $x_{2k} = elev_k$ and k is current discrete measurement.

Only the Steepest Descent of the gradient optimization algorithms is implemented. This is due to the simplicity of the algorithm, as the more advanced algorithms didn't show a clear enough results improvement to justify the work of implementing them properly.

7.4. Test Configuration

This section will go into detail about configurations that were universal for every test done on the real life system. Pictures from the test room and antenna configurations can be found section A.2.

7.4.1. Test Room

The test is done in an RF anechoic chamber. This is a room designed to have no reflective surfaces to avoid measurement errors and ambiguous data, as well as external noise. In practice this is done by donning the walls with triangle shaped cardboard that both absorbs and ensures that reflection only hits other cardboard triangles.

7.4.2. Physical Placements

The transmitting antenna is positioned 2.13 m over the floor on a stationary antenna mount. It is rotating 9.5° as an optimal point towards the receiving antenna. This angle was found by manually adjusting the sender antenna while the receiver antenna was connected to the NRPSensor and a computer was showing current continuous reading on the power level. The distance between the antennas is 3.7 m, the maximum the chamber allowed. Note that this is in the Fresnel Field for this parabola (see Fig. 2.6), so the radiation patterns will not present deep null points, but there will be clear side lobes and main lobes. The sender antenna is mounted onto the Moog QPT-90 with the help of a custom made aluminum board connected to a 40 mm rod that fits the antenna mount. The QPT-90 is fastened to a wheeled table with screws. This mostly for security reasons as the QPT-90 is heavy and the mounted antenna may topple over if not properly fixed to something solid and heavy. Fig. 7.12 illustrates the placements.

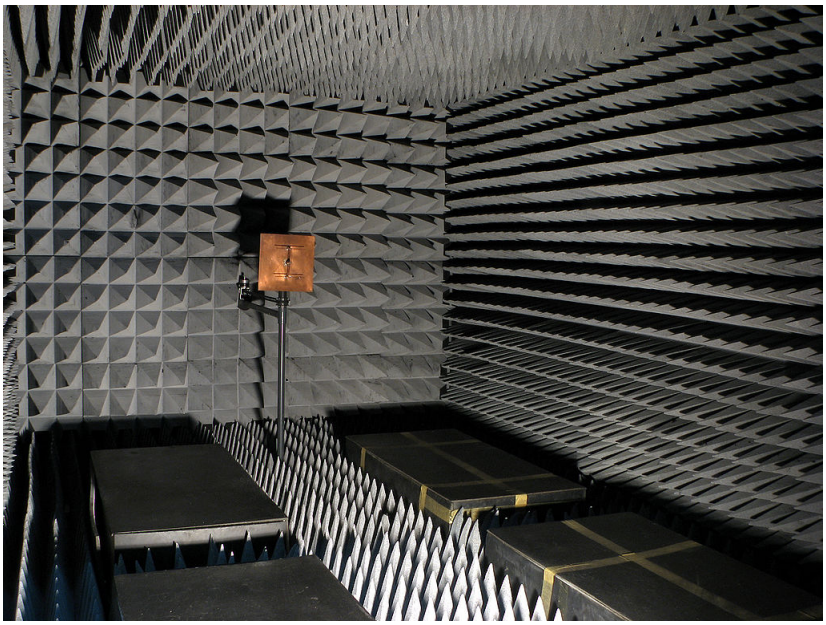


Figure 7.11.: Example of RF anechoic chamber[17]

7.4.3. Physical Limitations

The QPT-90 cannot tilt further down then ±90° from upright position. As the antenna is mounted perpendicularly to the QPT-90 load-bearer without any angle off-set, the limitations on elevation have been set to $0 - 70°$, where 0° elevation is when the QPT-90 has tilted +90° from upright position, and 90° elevation is when the QPT-90 is upright. Fig. 7.12 shows the antenna with elevation=0°. The lower limit is due to mentioned hard constraints from the QPT-90, while the upper limit is a programmed soft constraint to hinder the QPT-90 from tilting the antenna too far back, so it crashes into the ground and potentially breaks.

7.4.4. Power and Frequency of Signal

The power level of the signal sent is important in context of the problems discussed about the NRPSensor, see section 7.3.2.3. Because the sensor can end up blocking the measurement thread if the measured levels are not high enough, the sender level must be high enough so that the sensor reads levels over −30 dBm along the whole lobe search. 20 dBm was chosen as the signal generator started throwing warnings around 25 dBm. Fig. 7.13 shows two radiation patterns where one is a 4.7 GHz signal with 20 dBm while the other is the same frequency signal with only 1 dBm. What happens on the 1 dBm graph is that the NRPSensor is blocking the measurement thread, so only an old value is saved in the SharedDatabase while the QPT-90 keeps on turning.

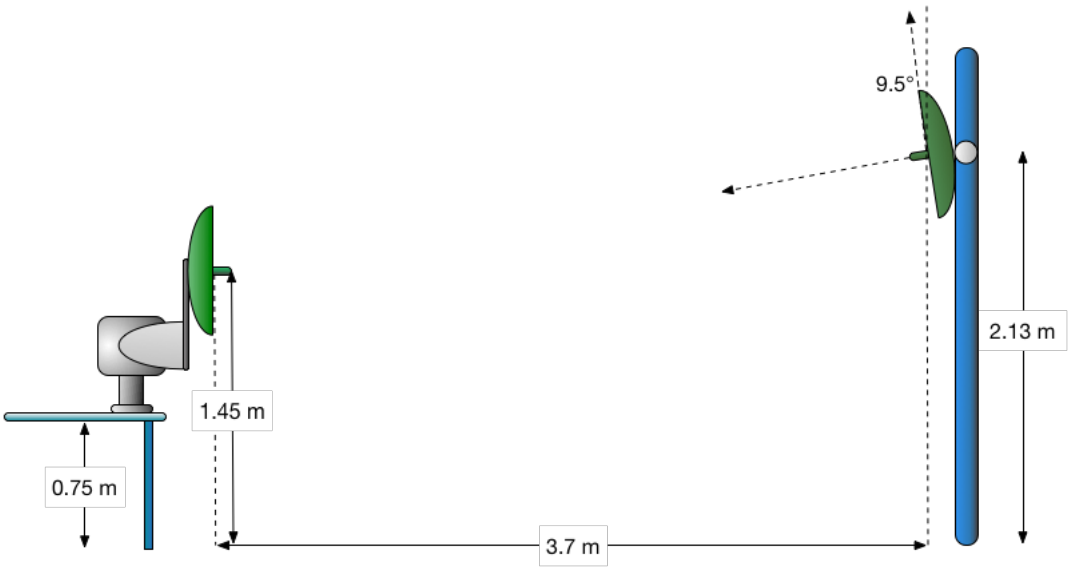


Figure 7.12.: Physical measurements of test configuration

7.4.5. Theoretical Maximum Received Power Level

To ensure that everything is correctly set up, the maximum theoretical receiving power level was calculated on the assumption that the antennas were perfectly aligned towards each other. The cable losses were measured and the theoretical free space loss was calculated. The losses are summed up as following:

- Measured loss along cables and equipment:
  - Short cable between receiving antenna and power sensor = -1.7 dB
  - Long cable between sending antenna and signal generator = -9.43 dB
  - Level offset from generator = -0.29 dB
- Free space loss:
  - Wavelength $= \lambda = \frac{c}{f}$, where $c = 3 \cdot 10^8$ and is light speed and f is frequency of the signal.
  - Distance $= R$
  - $20\log(\frac{\lambda}{4\pi R}) = 20\log(\frac{\frac{3e8}{4.7e9}}{4\pi 3.7}) \approx -57.25$ dB = Free Space Loss

The gain of each antenna is maximum+27 dB and the transmitting power is 20 dBm, so using Friis Transmission Equation, 2.3, as well as the losses in the cables and equipment, we got the following theoretical maximum received power:

$$P_r = 20dBm + 27dB + 27dB - 57.25dB - 0.29dB - 9.43dB - 1.7dB = \mathbf{5.33}dBm$$

The best value found manually was 1.03dBm with an azimuth value of $-30°$ and an elevation value of 10.5°. The reason for the difference between actual and theoretical values may be due to higher free space loss in practice.

Parameter	**Value**
Length between antennas	3.7 m
Lobe search azimuth range	$\pm 40°$
Lobe search elevation range	$0 - 30°$
Power Level	20 dBm
Frequency	4.7 GHz
Manually best found azimuth	$-30°$
Manually best found elevation	10.5°
Manually best found RSSI	1.03 dBm

Table 7.1.: Universal Test Parameters

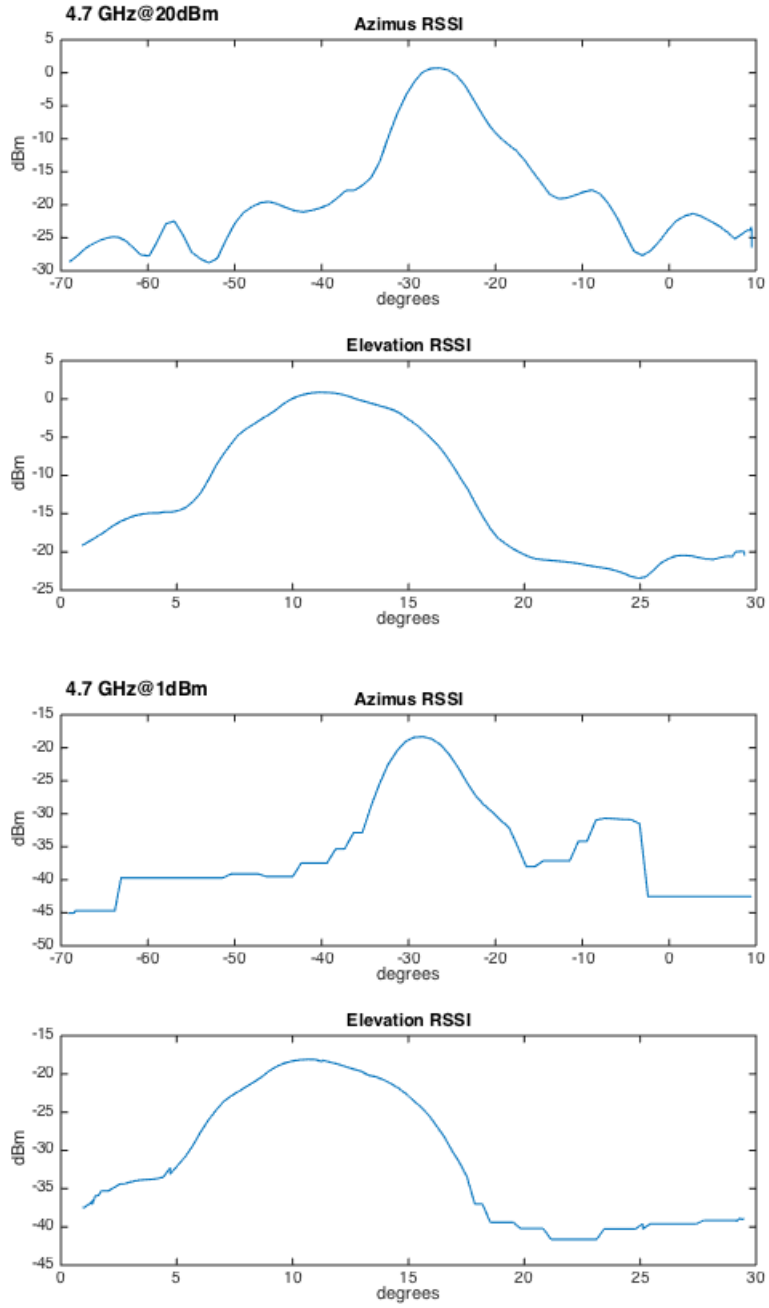


Figure 7.13.: Measured radiation patterns with different levels

8. Results from Real Life Tests

8.1. Overview

The following section will first show the results from a movement speed parameter test to determine the best movement speed for the lobe searches. Then the two algorithms will be run against each other with the best results from the speed parameter tests. The orientation data points will be given in the form of (azimuth/elevation) in degrees.

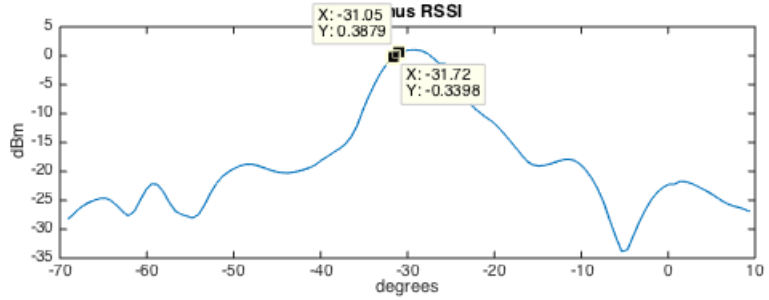
8.2. Movement Speed Tests

These tests were designed to determine the best movement speed for the two algorithms SLGO and FLND. More precisely, these tests were designed to determine the definition of "fast" and "slow" lobe searches. The QPT-90 Quickset protocol does not directly set the movement speed in deg/s, but rather defines a speed variable between 0-255, were 255 is maximum speed of 25 *deg/s* for azimuth and 8 *deg/s* for elevation. The tests are simple lobe sweeps, were the resolution of the measured data is important, especially in context of the gradient optimization.

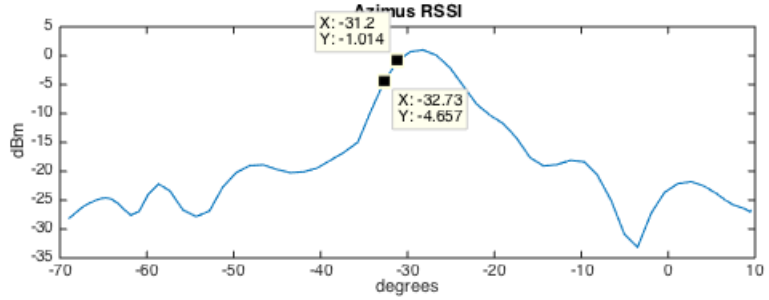
The start position for all these tests is azimuth $= -30°$ and elevation $= 10°$ and is close to the optimal position to ensure good readings on the lobe search. The resolution for the Steepest Descent method should be under $0.05°$ as this gives an appropriate number of measurements for the algorithm to calculate derivatives with high precision. The table Tab. 8.1 shows calculated resolution of the different lobe searches with different speed settings. 70 is found as an appropriate speed setting for the slow lobe search, as the resolution is just under $0.05°$. For the fast lobe search, one only needs a resolution that can identify the top point. Fig. 8.1 (c) shows that maximum speed (255) can identify the maximum lobe.

Speed	Resolution in degrees	Time taken in seconds
255	2.08	37.19
150	1.53	41.55
100	1.03	48.98
70	0.048	62.10
50	0.013	85.11

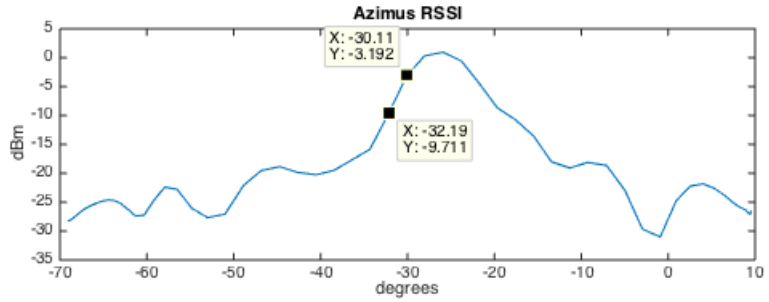
Table 8.1.: Resolution of different speeds



(a) Speed of 70



(b) Speed of 150

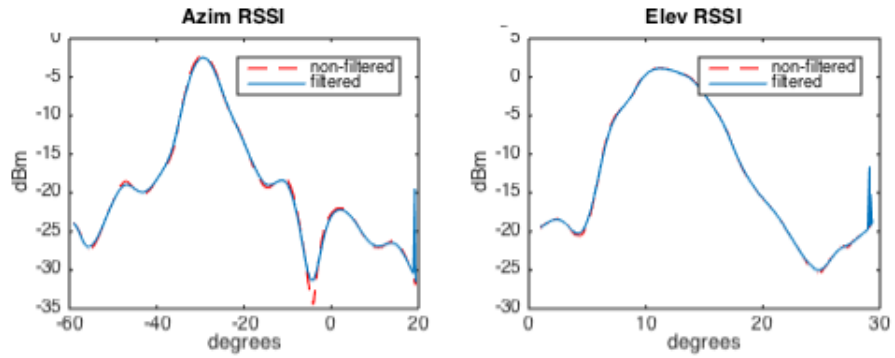


(c) Speed of 255

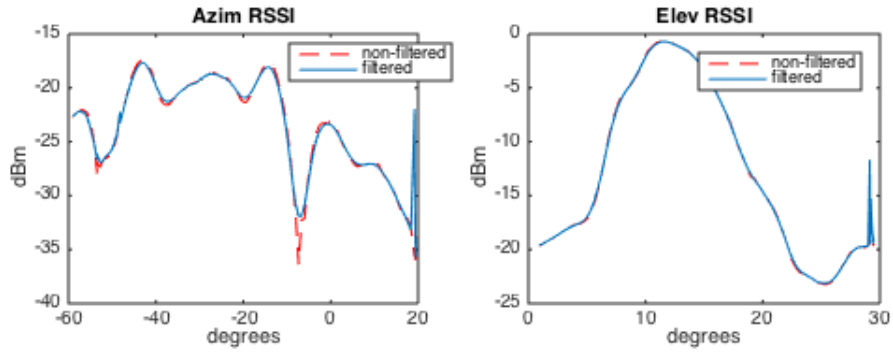
Figure 8.1.: Speed comparison of lobe search

8.3. SLGO Results

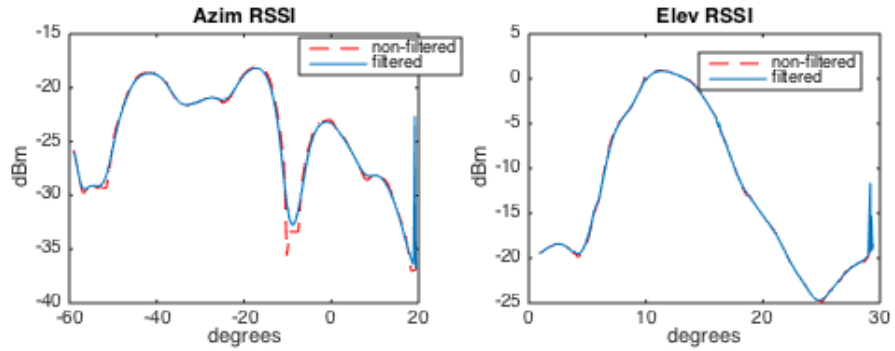
The following results show how well the SLGO performs from different initial positions, with the assumption that the initial position is sufficiently close to the main or side lobes to assume a connection has been made. The optimal point is around (-30,10.5), so a "close" elevation start point is around 15°, while a "far away" point is 2°.



(a) (-20,15)



(b) (-20,2)



(c) (-20,0)

Figure 8.2.: Difference in initial values for SLGO

Start Point [°]	(-20, 15)	(-20, 2)	(-20, 0)
End Point [°]	(-28.98, 10.17)	(-27.03, 10.69)	(-28.52, 10.43)
Best RSSI [dBm]	**0.826841**	-0.6165	0.756629
Time taken [dBm]	**58.1907**	65.9858	59.0644

Table 8.2.: SLGO - Different initial values results

8.3.1. Initial Values Results

Simulations show that initial elevation is most important for how well the lobe search performs, and the real life tests are no different. Therefore the tests are done with the same initial azimuth point, but different initial elevation points. The results are taken from an initial point close to the optimal called point 1: (-20, 15), a point further away called point 2: (-20, 2), and a point far away called point 3: (-20,0). Fig. 8.2 shows the different lobe search measurements, as well as the corresponding filtered values and calculated differentials. Optimal point results are shown in Tab. 8.2. Note that there are no gradient optimization results. This is because the lobe search was so effective on finding the optimal point that the Steepest Descent optimization terminated immediately.

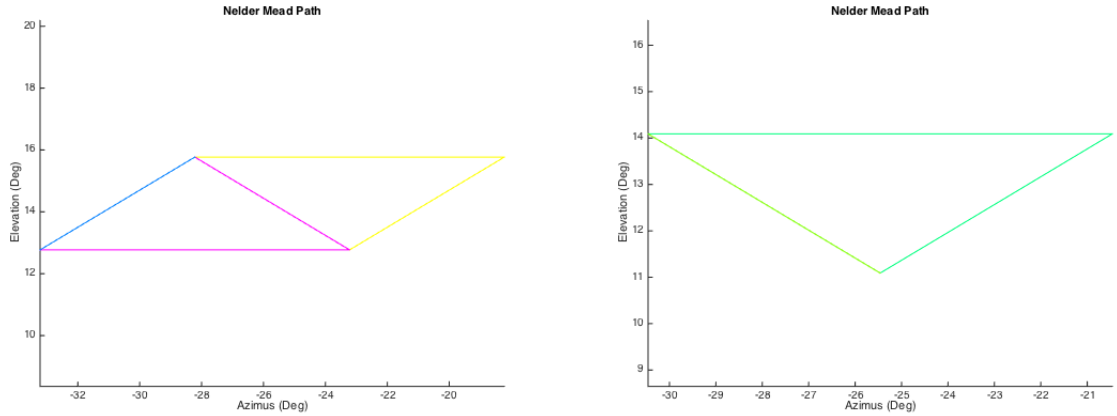
Not surprisingly, the best RSSI value was found from initial positions closest to the optimal point. However, the initial point 3 found a better end point then the initial point 2, even if point 2 is closer to the optimal point. The most likely reason can be traced to the measurement in Fig. 8.2, were the top point (c) for the main lobe is much smaller then the top point for the main lobe in (b). This could actually help the lobe search more easily identify the maximum point of the main lobe, as the measured lobe (c) is sharper than the lobe in (b). This could also just be a statistical issue.

8.4. FLND Results

Before gathering results to compare with the SLGO, a suitable speed for the lobe search must first be found. The first section will show results comparing two different speeds, while the next section will present the results with the chosen speed.

8.4.1. Movement Speed for Fast Lobe Search

From the movement speed tests in section 8.2 it was determined that the movement speed of 255 was enough for the lobe search to find the main lobe. However, tests show that the algorithm performs better when the lobe search speed is reduced. Fig. 8.3 (a) shows the Nelder Mead algorithm when the lobe search runs on maximum



(a) Speed of 255 **(b)** Speed of 150

Figure 8.3.: FLND speed comparison measurements, start position (-20, 2)

		Speed of 255	Speed of 150
Best Point Lobe Search	RSSI [dBm]	-7.85373	-3.23146
	Azimuth [°]	-23.22	-25.46
	Elevation[°]	12.69	11.00
Best Point Nelder Mead	RSSI [dBm]	-2.95639	**-0.417485**
	Azimuth[°]	-33.22	-30.46
	Elevation[°]	12.77	14.09
	Time Taken [s]	69.9368	**64.8849**

Table 8.3.: FLND speed comparison results, start position(-20,2)

speed, while (b) shows the Nelder Mead algorithm when the lobe search runs on speed 150. The lobe search in (a) resulted in an initial point further from the optimal point, and the Nelder Mead algorithm corrects for this difference. The Nelder Mead run in (b) terminates immediately, as the simplex collapses onto itself. From Tab. 8.3, we see that the maximum point found in (b) is not from the Nelder Mead, but one of the starting points of the initial simplex, found through the method discussed in section 5.5.1.3. We also see that the maximum point found with lobe search speed of 150 has both better RSSI and is less time consuming to find compared to the algorithm run on maximum speed. The conclusion is that the Nelder Mead algorithm is inefficient compared to lobe search, and a slower lobe search yields better results in both time taken and achieved RSSI.

8.4.2. Results with different initial values

These tests were done with a lobe search speed of 150. The initial points tested are the same initial points as from the SLGO initial points test in section 8.3.1. Fig. 8.4

	Initial Values	(-20, 15)	(-20, 20)	(-20, 0)
Best Point Lobe Search	RSSI	0.880053	-3.93452	-21.5654
	Azimuth	-28.33	-25.19	0.25
	Elevation	10.99	10.97	10.58
Best Point Nelder Mead	RSSI	**0.88053**	-0.356347	-21.5654
	Azimuth	-28.33	-30.19	0.25
	Elevation	10.99	14.07	10.58
	Time Taken	61.4871	**60.6735**	67.5737

Table 8.4.: FLND - Different initial values results

shows the measured lobe values and Nelder Mead path for each run. The run shown in (b) shows a typical main identification lobe algorithm case, where the middle lobe was correctly identified as the main lobe when the side lobes have similar amplitudes. The run in (c) shows a failure case, were the orientation off-set from the optimal point was too large, so the NRP Sensor did not measure the power levels needed for continuous updates. This caused a gap in the measurements that the main lobe identification algorithm is not designed to handle, and identifies a side lobe as the main lobe. All the runs have a similar Nelder Mead optimization path, were the algorithm terminates immediately after only discovering the initial simplex.

The optimal point results are given in Tab. 8.4. Only the (-20,20) run finds a new optimal point from the Nelder Mead algorithm, while the other two runs share optimal points with the lobe search results. The best results are from the initial values closest to the optimal point, because this gives clearer lobe search measurements to work with.

8.5. Comparison between SLGO and FLND

Tab. 8.5 shows tables that compare the end results of the SLGO and FLND algorithms over different initial values, as well as the manually found best point. SLGO is faster over all the runs, but both algorithms yield very close RSSI results except for the point far away from the initial value, (-20,0). In this case the FLND simply failed because the lobe search experienced measurement errors due to the sensor and search speed.

8.6. Creating Disturbances

A simple test was conducted to try to emulate some real life disturbances. A metal board was placed under the receiving antenna to see if reflecting radio waves could

(a) Middle - Start Point = -20/20

Algorithm	SLGO	FLND	Manually Found
End Azimuth [°]	-27.04	-30.91	-30
End Elevation [°]	10.45	13.68	10.5
End RSSI [dBm]	-0.755376	**-0.263392**	1.03
Time Taken [s]	**56.233**	60.1512	

(b) Close - Start Point = -20/15

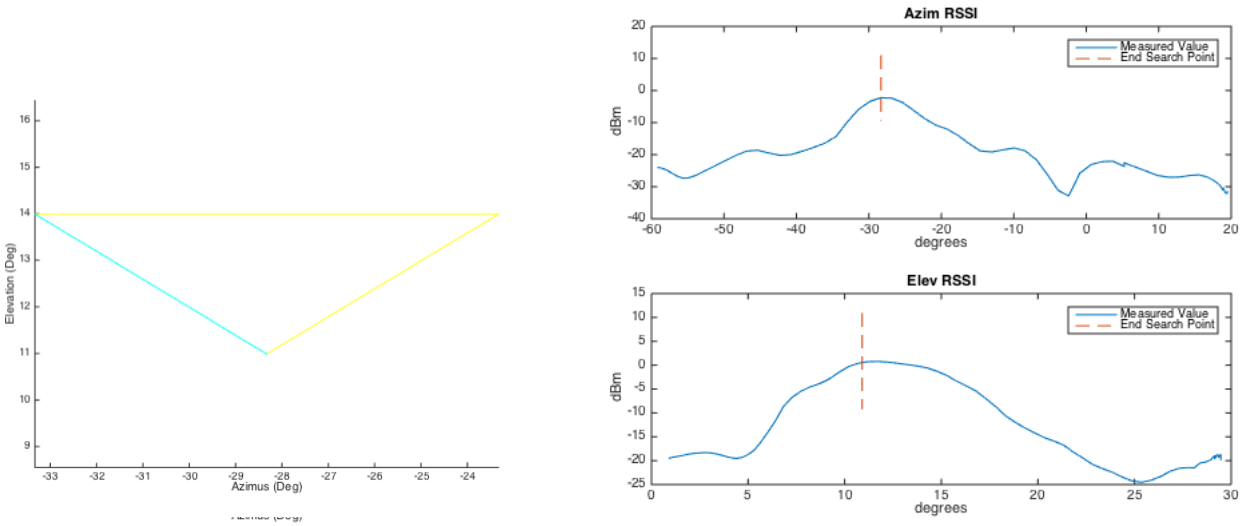
Algorithm	SLGO	FLND	Manually Found
End Azimuth [°]	-28.9	-30.19	-30
End Elevation [°]	10.17	14.07	10.5
End RSSI [dBm]	**0.826841**	-0.356347	1.03
Time Taken [s]	**58.1907**	60.6735	

(c) Far - Start Point = -20/2

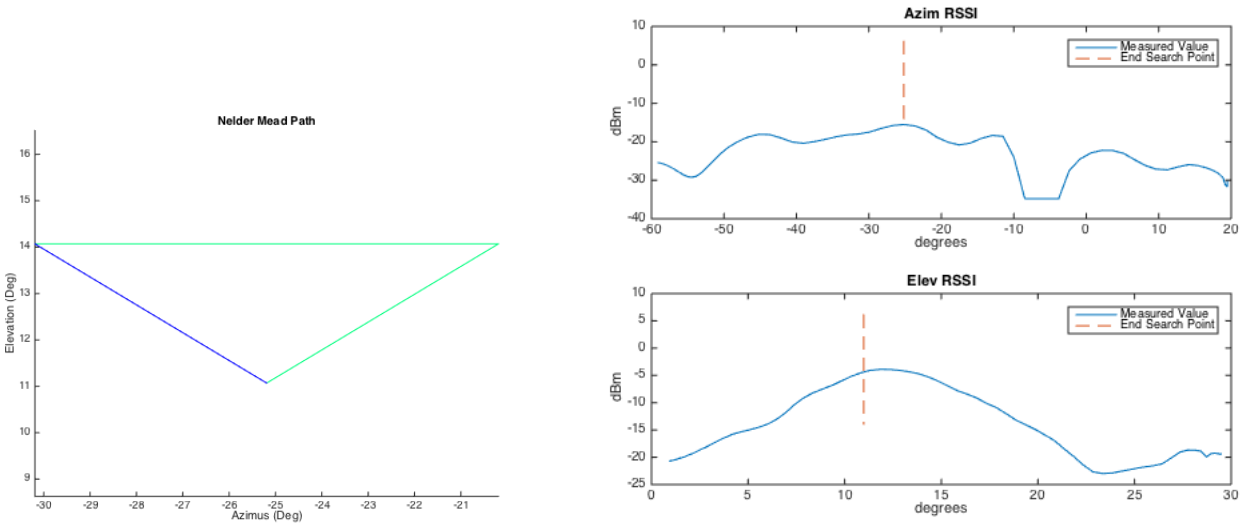
Algorithm	SLGO	FLND	Manually Found
End Azimuth [°]	-27.03	-33.22	-30
End Elevation [°]	10.69	12.77	-10.5
End RSSI [dBm]	**-0.6165**	-2.95639	1.03
Time Taken [s]	**65.9858**	69.9368	

Table 8.5.: SLGO and FLND results comparisons

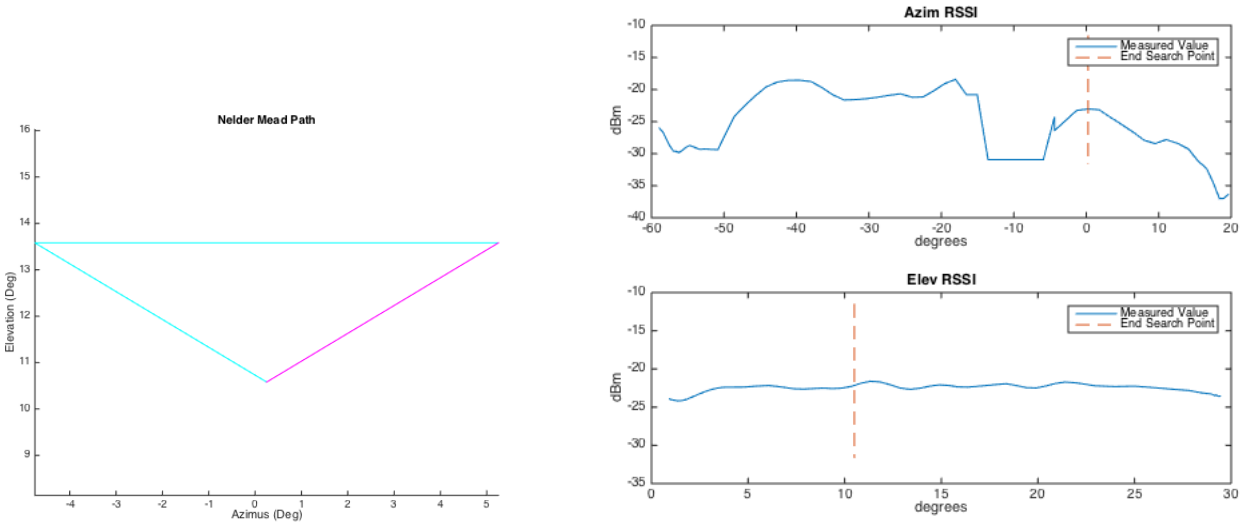
cause interference that could break the search algorithms. A picture of this test can be found in the appendix section A.2. Fig. 8.5 shows two lobe searches, with and without disturbances. As the metal board was placed under the antenna, it is not surprising that the elevation measurements were mildly affected. What happened is that the reflected radio waves "filled in" a zero point on the lower end of the elevation range. This can be seen by comparing the elevation measurements from degrees around 3-5. In this case the reflections only affected non crucial patterns of the lobe search, as the azimuth patterns are similar.



(a) Start Point = (-20, 15)

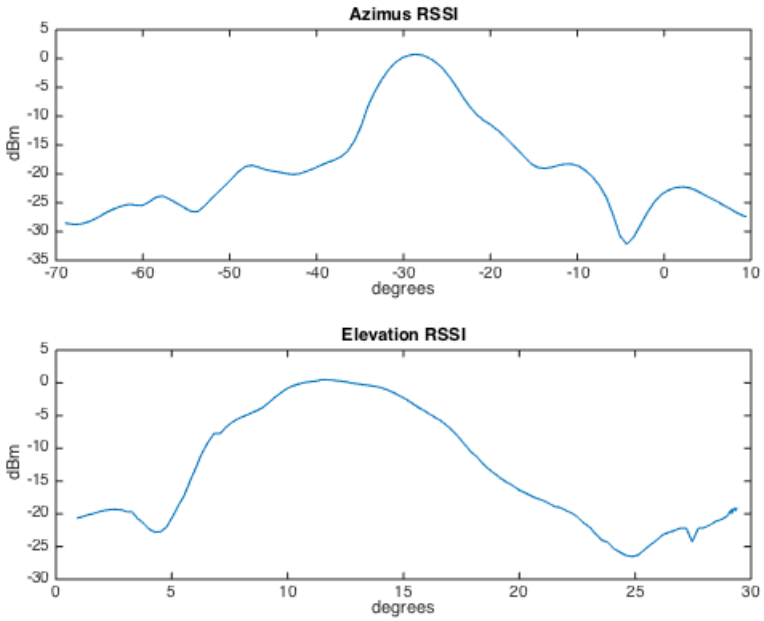


(b) Start Point = (-20, 20)

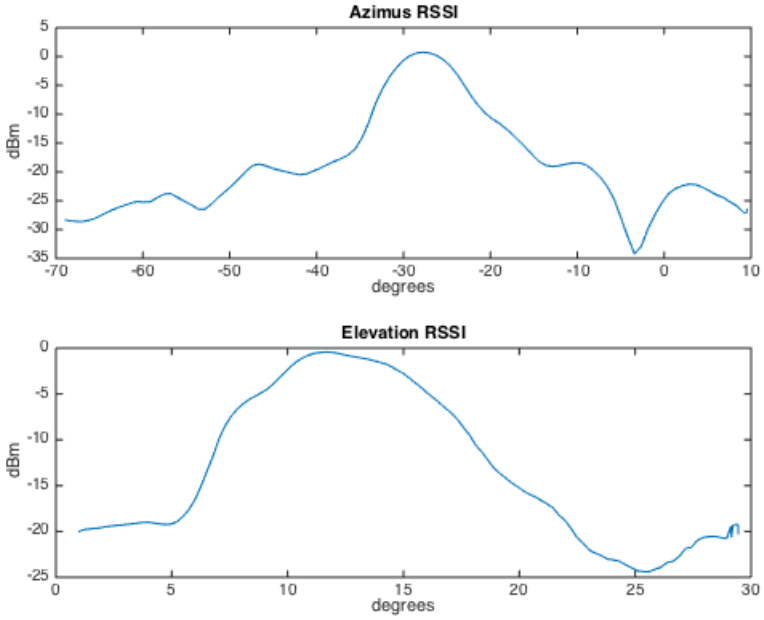


(c) Start Point = (-20, 0)

Figure 8.4.: Different initial values measurements for FLND



(a) Without Disturbance



(b) With Disturbance

Figure 8.5.: Lobe search results with disturbance

9. Discussion

9.1. Overview

This chapter will discuss the simulated and real life results, and will form the basis for this project's conclusion.

9.2. Time Used

Time used is mostly compared between the two algorithms tested, but what are acceptable absolute values? Manually directing a parabolic antenna can be a very time-consuming task, especially when the antenna is placed far away and must be controlled by ropes or something similar. This project is the second part of a three-phase search, optimize and lock system, were the search can vary very much depending on the information the system has. The results show an optimization time with the lobe search of around one minute for a single antenna. This is a task that could end up being performed several times during an operation cycle due to antennas moving out of position so that a new optimal position needs to be calculated. However, one minute of down time is acceptable.

9.3. Difference Between Simulated Results and Real Life Results

9.3.1. Noise and Filtering

The largest difference was most notably how noise was simulated compared to how real noise acted. Real noise came in the form of measurement lag because of how the sensor was implemented into the program. Due to this, the gradient optimization was useless for most SLGO runs, as the lobe search managed to find the optimal point without the need for adjustment. There are some example runs (not shown in the results chapter), where the filtered top point differed from the lobe search maximum value, but that happened in cases where the measured main lobe was so sharp that the filter filtered out the top point. Fig. 9.1 shows an example run where this happened. The lobe search RSSI results was 0.0385 dBm, while the Steepest

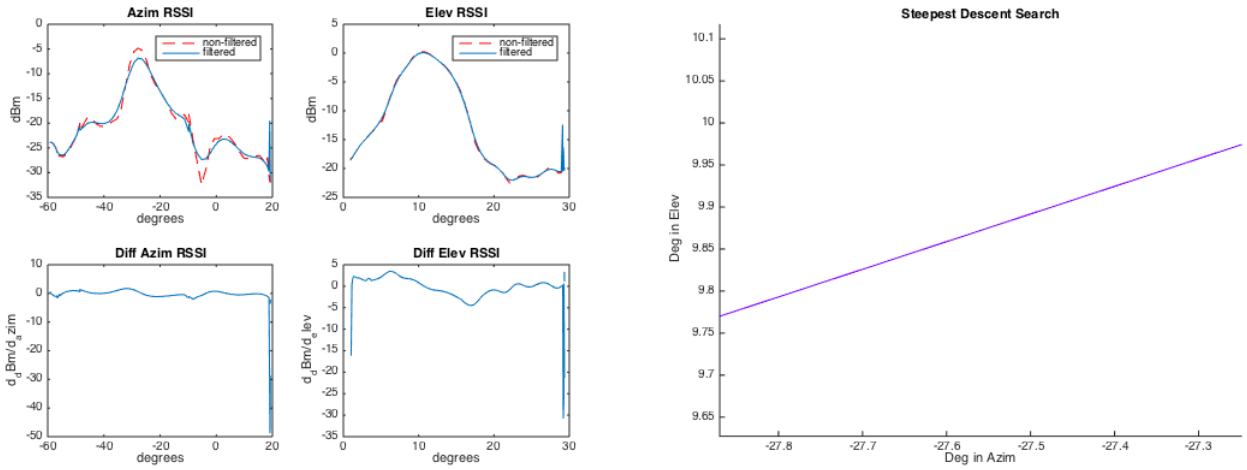


Figure 9.1.: Example of SLGO were Steepest Descent was run

Descent algorithm finished with a RSSI result of -0.4446 dBm. In other words, the filter did not filter out false maxima but instead blurs the real top value.

Noise in measurements is dependent on what type of sensor is used and the environment in which the measurements are taken. With this in mind, the real life tests are set in ideal conditions with a high performance sensor. It could be that cheaper sensors, or radio frequencies in higher or lower power levels will produce more random and discontinuous measurements so that the filter could still be of use, but nothing from the tests point to these assumptions.

When run under outdoor real conditions noise will most likely produce warped lobes and smooth unexpected differences. section 8.6 discusses a mild example of where reflected radio waves distorted the lobe search measurements with a small amount. However, this was not enough to affect the main lobe identification algorithm. There could be environments that reflect signals along the azimuth range, for example mountain walls, that could flatten out lobes or create new side lobes with higher power. Fig. 9.2 shows a thought out worst case example that would confuse the main lobe identification algorithm. The algorithm would find two candidate lobes, and try to find the main lobe between them. However, no lobe would be found. This could be solved by introducing more cases for the algorithm to identify, but how many cases should the algorithm account for? Could a completely new, different search algorithm be more effective?

There is also the case of signal-to-noise ratio, where reflected radio waves introduce a time delay. In this project only the measured power is taken into account through RSSI. Reflections introduce deviations as reflections can amplify the received power, but introduce a time delay that decreases the overall quality of the data the radio waves received. A solution to this problem could be to use a signal-to-noise indicator for measurements instead of RSSI. This would naturally filter out reflected waves, and still give similar radiation patterns as the RSSI. However, the best solution is

likely a weighted sum of both RSSI and signal to noise ratio, as signal-to-noise ratio could have a much smaller operational area or resolution around the optimal point.

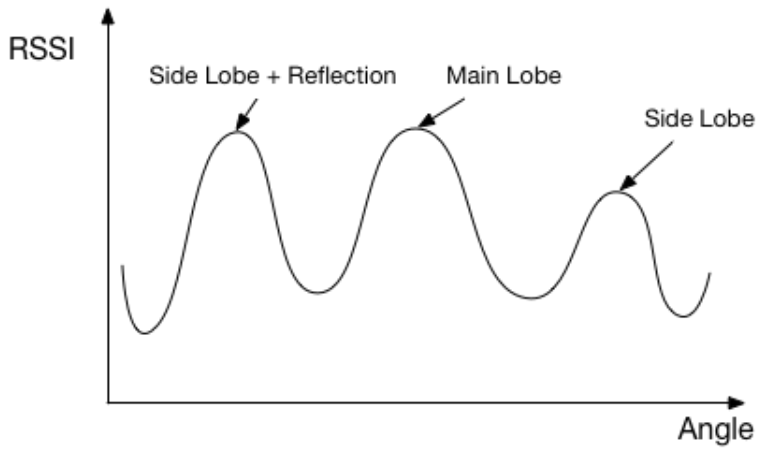


Figure 9.2.: Lobe search with worst case reflections

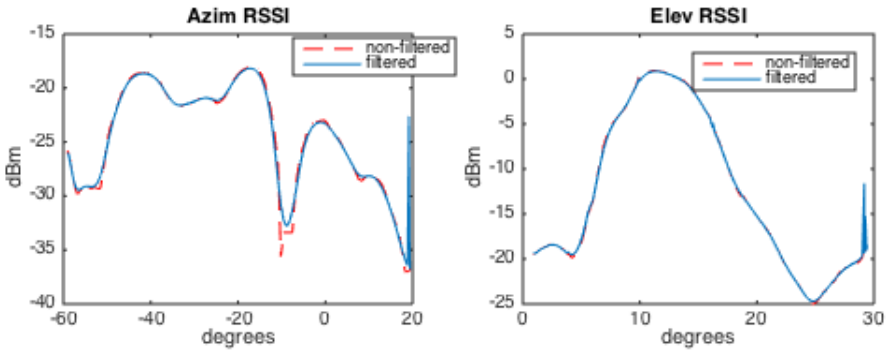
9.3.2. Radiation Pattern Model

The simulated 3D-model is approximated by simply summing the principle planes gains. This approximation loses important radiation pattern characteristics, especially along the first azimuth sweep in the lobe search algorithm. Compare the two azimuth sweeps in Fig. 9.3. Both of the sweeps start in elevations far from the optimal elevation. In the simulated case it is 20 degrees, while the real-life case is around 15-14 degrees from optimal elevation. The simulated azimuth simply flattens out the pattern the further away from the optimal elevation the sweep starts, but the main-lobe is still clearly identifiable as the side-lobes have lower power levels. The real-life model shows an example were the side lobes have much higher levels than the main lobe, when the azimuth sweep starts in an elevation far from the optimal elevation. The main lobe identification algorithm is not used on the simulated model, but the real-life tests are dependent on the algorithm on initial value elevation ranging over 12 degrees from the optimal point.

Clearly a better approximation for the 3D model is needed for simulations to stay relevant. One could try to create a 3D-model by doing exhaustive measurements on a real life antenna, as only $\pm 30 - 40°$ from the optimal point along azimuth and elevation is relevant for the current scope. Another alternative is to approximate from principle planes using more complex interpolation methods that add weights depending on angular distances from the planes [16], or geometric approaches were the elevation is approximated by rotations that take into consideration the azimuth radiation pattern[7]. There is also the possibility of creating 3D-models with antenna design software, and even using the software to simulate complex environments.

This is especially useful when wanting to simulate how radio wave propagations may alter the radiation pattern. Examples of such software is the FEKO Suite (https://www.feko.info) and CST Microwave Studio (http://www.cst.com).

(a) Simulated pattern



(b) Real life pattern

Figure 9.3.: Radiation pattern differences

9.3.3. Sensor and Orientation Synchronization

In the real life system there is no synchronization between the measuring thread and the rest of the system. This means that there is no real guarantee that the current measurement actually corresponds to the paired orientation point measured from the rotator. This problem is also escalated by how the NRP-Z21 sensor driver works, as it blocks the measurement update thread until a new measurement is complete. When measured power levels are over around -30 dBm, the sensor updates in a timely manner of around 10 Hz. However, anything lower, and the measurement can take over a second. The rest of the system has no idea that this is occurring, and pairs old measurements to current orientation data. Even if the environment is configured to never measure anything below -30 dBm, the lack of synchronization causes doubts if the orientation data paired with the measurement is actually the correct orientation. It could be that the measurements lag constantly behind the orientation data.

The best solution would be to both implement a time-out for the NRP sensor so that it does not block the thread for a long time, as well as a synchronization between the measurement thread and the lobe search sweep to ensure that the orientation measurement and RSSI measurement are paired correctly. This most likely means that the lobe search must change the lobe sweep parameter from rotation speed to number of points to measure along a single sweep.

9.4. Fast Lobe Nelder Mead VS Slow Lobe Gradient Optimization

Simply put, the slow lobe gradient optimization (SLGO) algorithm is the fastest and produces the best results. The slow lobe search is so efficient that the gradient optimization seems to be superfluous, as the few times the optimization algorithm ran during the real life tests, it only did so because the filter filtered out the sharp main lobe. The Nelder Mead algorithm did produce competent results, with a few degrees offset from the best point, but it used more time. The real life results show that is was better to reduce the lobe search speed so that the Nelder Mead would start closer to the main lobe, as moving to all the points that need to be measured is inefficient.

9.4.1. The Point of Gradient Optimization?

The initial aim of the gradient optimization step was that the lobe search could be done quickly with few sampling points, as long as the initial point was close to the main lobe optimum. This was dismissed when it was discovered that the gradient optimization is dependent on the sampling of the lobe search to build approximate derivates.

Now that the lobe search was slowed down, it would find the optimal point on its own as there is less room for error. Here the idea of simulated noise was introduced. The engineers at Kongsberg were not sure about the effect of noise in RSSI measurements, but did not find it surprising if the values would fluctuate between $\pm$1dBm. The simulated noise could cause "false maxima" points were the lobe search would end up with an offset from the real optimal value. A solution was to use a smoothing filter on the measurements, and approximate derivatives from the filtered values to use for gradient search. This would both ensure smooth approximated gradients and a better optimal point approximation.

However, why not just traverse directly to the maximum point of the filtered value, as these are values already measured and known?

To test this out, we create a simple simulated test run with a slow lobe search and append on a simple "move to" command that moves directly to the maximum point

of the filtered measurements. Call this algorithm SLMT. Tab. 9.1 shows the end results compared with the same slow lobe search results with the BFGS method run afterwords. We see that the results are very similar. The difference from real optimal is determined by offset introduced from filtering, the tolerance value of the "move to" command and the speed of this particular movement. Remember that the best RSSI values are also affected by noise, and could vary up to ±1 dBm on the same point. The point is that the SLMT is a much less demanding algorithm, and does not depend on approximated differentials.

	Antenna 1		Antenna 2	
	SLMT	SLGO	SLMT	SLGO
Difference from optimal azimuth [°]	-0.400	-1.0949	-0.400	-0.747
Difference from optimal elevation [°]	-0.500	0.3645	-0.501	-0.211
Best RSSI [dBm]	-38.519	**-37.898**	**-38.2882**	-38.818
Time Taken [s]	**63.5**	64.5	**63.5**	64.4

Table 9.1.: Slow Lobe Move To compared to best SLGO Results

The conclusion is that this system and problem does not need to use gradient optimization, and can jump over the approximate derivative steps. If filtering is required, extracting azimuth and elevation maximum values should be trivial. If filtering is not needed, a well calibrated lobe search should manage to find optimal point.

9.4.2. How Can Nelder Mead Still Be Relevant?

The Nelder Mead algorithm has the clear advantage of being point-based. If a cheaper sensor simply needs more time on each measurement, even a slow movement lobe search could end up taking several minutes to complete. The alternative could be a fast lobe search were the parameter for the lobe search is the number of points to be measured, instead of the speed of the rotator movement. For this case however, the measurement speeds and precision favor a simple slow lobe search.

9.5. Lobe Identification

The lobe identification algorithm correctly identified the main lobe for all the results presented except for Fig. 8.4 (c) where the sensor was out of sync. This can be proven by looking at the results from the SLGO runs, and compare the radiation patterns in Fig. 8.2 (b) and (c) to the optimal points presented in Tab. 8.2. For all the cases the main lobe identification algorithm is designed to identify, the lobe search found points a few degrees from the optimal position.

However, there are several factor about the main lobe identification that should be scrutinized. The algorithm was reactively implemented, as the real life systems

showed patterns the simulated system simply could not duplicate, as the simulated model was not a good approximation of a real 3D radiation pattern. The algorithm was designed to handle specific cases that showed up during test setup described in chapter 7, where the sender antenna was pointing directly at the rotating antenna. There is no guarantee that the side lobes will display the same symmetry when the sender antenna is at an angle, but taking into consideration Friis and the fact that one of the antennas will always stand still, the lobe search will display variance decided only by the moving antennas gain curve, as the static antenna gain is constant during the search. This should theoretically result in symmetrical side lobes pattern when measuring across the azimuth.

10. Further Work

This chapter will present a summary of different ideas for further work discussed in different parts of the project.

10.1. Both Simulation and Real Life

The following bullet points show improvements that affect both the simulations and the real life software:

- Implement a more advanced received signal indicator than RSSI, that takes into consideration signal-to-noise ratio so that propagated radio waves are filtered from the lobe search measurements.
- Test the system with reflected radio waves and other sources of disturbances and noise.
  - Modify the main lobe identification algorithm if needed to handle new possible cases.

10.2. Simulation

The simulation needs some certain improvement in order to better represent the real life situation:

- Better 3D-models of radiation pattern. There are several ways this could be improved upon:
  - Approximate 3D-models from the principal plane measurements using more complex interpolation methods.
  - Do an exhaustive measurement on a real life directive antenna configuration and gather enough data to complete a detailed 3D-model of the antenna within $\pm 30 - 40°$.
  - Use advanced EMC/Antenna simulator software to simulate antenna configurations with radio wave propagation. An example of such software is the FEKO suite. (www.feko.info)

10.3. Real Life

For real life testing, the following work is suggested:

- Synchronize measurements with orientation measurements. This is a software implementation issue, and the lack of synchronization may cause a constant off-set between measured value and the paired orientation.
- Use an Inertial Measurement Unit (IMU) for orientation data. This is important when running the optimization phase in windy conditions, as orientation data from the servo is useless if the servo changes position relative to earth.
- Test two rotating systems that communicate.
  - Implement real time state machines that wait for remote systems, while also using timeout for failure states that should reset protocols for both systems.
  - Check how well the system perform when both start in positions where the side lobes point towards each other.
- Complete a prototype of the hardware solution using Kongsberg equipment as well as a smaller pan-tilt rotator that can fit on top of a mast. Fig. 10.1 shows a hardware solution with a Kongsberg 542a Radio Link[5] mounted under a QPT-50[1] (a smaller version of a QPT-90). The sketch is created by Kongsberg Defense Systems.
  - The software must interface with the 542a Radio Link and extract RSSI data from it.

[1] http://www.moogs3.com/literature/Space_Defense/QuickSet/QPT-50.pdf

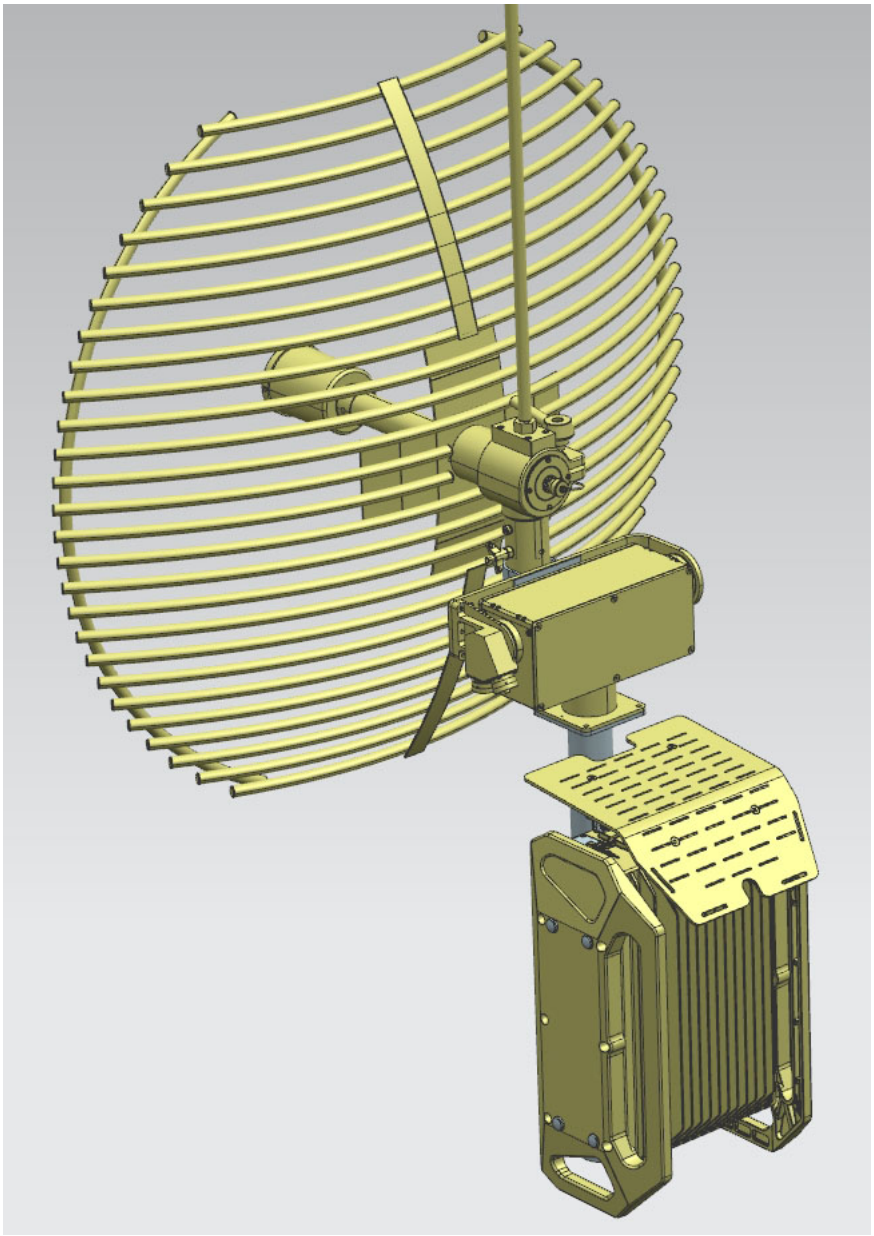


Figure 10.1.: Concept sketch of hardware solution for antenna pointing system

11. Conclusion

The goal of this project was to find an optimization algorithm for an automatic directional antenna control system. The project built upon the assumption that an automatic search between two directional antennas would not initially be in an optimal position when finding each other, and that the antennas could communicate with each other. Using this communication channel, a lobe search algorithm was designed to identify the desired main lobe and avoid that the antennas point at each other with side lobes. The lobe search algorithm was based upon antenna theory presented in this project. Building upon data from the lobe search, two optimization algorithms where proposed:

- Slow Lobe Search - Gradient Optimization (SLGO)
- Fast Lobe Search - Nelder Mead Optimization (FLND)

SLGO used filtered data to approximate a gradient for the antenna patterns in both azimuth and elevation, that was then used in a gradient optimization algorithm to find the optimal point. FLND simply did a quick lobe search to ensure the Nelder Mead optimization algorithm had an initial value close to the main lobe. The algorithms were first tested through a simulation environment built in MATLAB before being tested in a real life test environment.

SLGO ended up being built upon flawed logic, especially the gradient optimization part. The measurements did not need filtering in real life, and the simulated filtered values did not need gradient optimization as the maximum value was already known from the measured data from the lobe search. FLND produced mostly worse results then SLGO, as well as taking several seconds longer for each run. This applied to both real-life tests and simulations, and was due to the Nelder Mead algorithm simply being too time consuming.

This Master Thesis work has through simulations and real life tests indicated that the most promising algorithm is simple lobe search that is slow enough to build a radiation pattern were the main lobe is identifiable. The lobe search managed to effectively find close to optimal values from difficult initial measurements. Further work is needed to see how the system works with reflected radio waves in a real life environment.

Bibliography

[1] *EUROCOM Enhanced System Specifications ESS, part D: System Specification, rev. 1*, 1999.

[2] Constantine A. Balanis. *Antenna Theory: Analysis & Design*. John Wiley & Sons, INC, 2005.

[3] Peter Joseph Bevelacqua. Decibels (db). `http://www.antenna-theory.com/definitions/decibels.php`, 2009. [Online; accessed 1-January-2015].

[4] Peter Joseph Bevelacqua. Friis equation - (aka friis transmission equation. `http://www.antenna-theory.com/basics/friis.php`, 2009. [Online; accessed 1-January-2015].

[5] Defence Communications. *RL542A - Tactical Radio Link*. Kongsberg Defence & Aerospace AS, February 2013.

[6] Comrod. *SHF4450P08 Band 4 Radio Relay (LOS) Dish Antenna*, 2013.

[7] Waslon Terllizzie Araujo Lopes, Giuseppe Glionna, and Marcelo Sampaio de Alencar. Generation of 3d radiation patterns: A geometrical approach. In *Vehicular Technology Conference*, volume 2, pages 741–744. IEEE, 2002.

[8] Gough Lui, Thomas Gallagher, Binghao Li, Andrew G. Dempster, and Chris Rizos. Differences in rssi readings made by different wi- fi chipsets: A limitation of wlan localization. In *2011 International Conference on Localization and GNSS*, pages 53–57, 2011.

[9] MathWorks. filtfilt - zero-phase digital filtering. `http://se.mathworks.com/help/signal/ref/filtfilt.html`. [Online; accessed 9-January-2015].

[10] MathWorks. Signal smoothing. `http://se.mathworks.com/help/signal/examples/signal-smoothing.html`. [Online; accessed 9-January-2015].

[11] Daniel Minoli. *Satellite Systems Engineering in an IPv6 Enviroment*. Auerbach Publications, 2009.

[12] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization: Second Edition*. Springer, 2006.

[13] Y. Rahmat-Samii, L. I. Williams, and R. G. Yoccarino. The ucla bi-polar planar-near-field antenna measurement and diagnostics range. *IEEE Antennas & Propagation Magazine*, 37(6), December 1995.

[14] Rohde & Schwarz GmbH & Co. KG. *R&S NRP Power Meter Family*, 2014. Product Brochure.

[15] Jan Simon. Filterm. `http://www.mathworks.com/matlabcentral/fileexchange/32261-filterm`, 2011. [Online; accessed 15-January-2015].

[16] Theodore G. Vasiliadis, Antonis G. Dimitriou, and George D. Sergiadis. A novel technique for the approximation of 3-d antenna radiation patterns. *IEEE Transactions On Antennas And Propagation*, 53, July 2005.

[17] Wikipedia. Anechoic chamber — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Anechoic_chamber&oldid=637900968`, 2014. [Online; accessed 1-January-2015].

[18] Wikipedia. Polar coordinate system — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Polar_coordinate_system&oldid=639288983`, 2014. [Online; accessed 1-January-2015].

[19] Wikipedia. Spherical coordinate system — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Spherical_coordinate_system&oldid=640251893`, 2014. [Online; accessed 1-January-2015].

[20] Wikipedia. Decibel — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Decibel&oldid=641904346`, 2015. [Online; accessed 12-January-2015].

[21] Chen Yangquan. Ansi c implementation of matlab functions filter and filtfilt. `http://mechatronics.ece.usu.edu/yqchen/filter.c/`, 2003. [Online; accessed 15-January-2015].

A. Appendix

A.1. Derivation of the BFGS method

The following will show the derivation of $\mathbf{B}_k$ used in the BFGS method. The theory in this section is taken from "Numerical Optimization: Second Edition".[12]

The derivation starts with the Taylor second order approximation:

$$f(\mathbf{x} + \alpha\mathbf{p}_k) \approx f_k + \nabla f_k^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \mathbf{B}_k \mathbf{p} = m_k(\mathbf{p})$$

Setting the derivative of the approximation to zero, we get the step direction:

$$\mathbf{p}_k = -(\mathbf{B}_k)^{-1} \nabla f_k$$

To find the next step we define a new second order approximation for the next step:

$$m_{k+1}(\mathbf{p}) = f_{k+1} + \nabla f_{k+1}^T \mathbf{p} + \frac{1}{2}\mathbf{p}^T \mathbf{B}_{k+1} \mathbf{p}$$

To connect the two steps, we want the gradient of the objective function to equal the gradient of the second order approximation $m_k(\mathbf{p})$ for $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. In other words, $\nabla f_k = \nabla m_k$ and $\nabla m_{k+1} = \nabla f_{k+1}$. The second condition works if we set$\nabla m_{k+1}(0) = \nabla f_{k+1}$.

For the first condition we set:

$$\nabla m_{k+1}(-\alpha_k \mathbf{p}_k) = \nabla f_{k+1} - \mathbf{B}_{k+1}\alpha_k \mathbf{p}_k = \nabla f_k$$

We rearrange and get

$$\mathbf{B}_{k+1}\alpha_k \mathbf{p}_k = \nabla f_{k+1} - \nabla f_k$$

We simplify $\alpha_k \mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \mathbf{s}_k$ and $\nabla f_{k+1} - \nabla f_k$ and get the following called the secant equation:

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k$$

Using the secant equation and the condition that $\mathbf{B}_k$ must be symmetric and positive definite, one determines $\mathbf{B}_{k+1}$ by solving the following optimization problem, where $\mathbf{H}_k = \mathbf{B}_k^{-1}$

$$\begin{gathered} min_{\mathbf{B}_{k+1}} ||\mathbf{H}_{k+1} - \mathbf{H}_k|| \\ \mathbf{s}.t \\ \mathbf{H}_{k+1} = \mathbf{H}_{k+1}^T \\ \mathbf{H}_{k+1}\mathbf{s}_k = \mathbf{y}_k \end{gathered}$$

The solution to this problem is:

$$\mathbf{H}_{k+1} = (I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k})\mathbf{H}_k(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{A.1}$$

The Sherman-Morrison-Woodbury formula is given by:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}}$$

Using the Sherman-Morrison-Woodbury formula while substituting in $\mathbf{B}_k = \mathbf{H}_k^{-1}$ in A.1 gives A.2:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \tag{A.2}$$

A.2. Pictures of the real life tests

This section shows pictures of the real life tests done in an RF anechoic chamber.

(a) QPT-90 antenna mount **(b)** Static antenna mount

Figure A.1.: Antenna mounts

Figure A.2.: Disturbance test

A.3. Data sheets for antenna and pan tilt rotator

SHF4450P08

Band 4 Radio Relay (LOS) Dish Antenna

Application:

- Band 4 frequency, 4.4 –5.0 GHz
- Designed for line-of-sight (LOS) and high capacity line-of-sight (HCLOS) radio relay communications.
- High gain
- Rotating feed to set vertical or horizontal polarisation
- Rugged high quality antenna with a durable construction
- Suitable for harsh environments.
- Standard mast mounting interface
- Removable feed for protection and easy transportation

Electrical specifications:

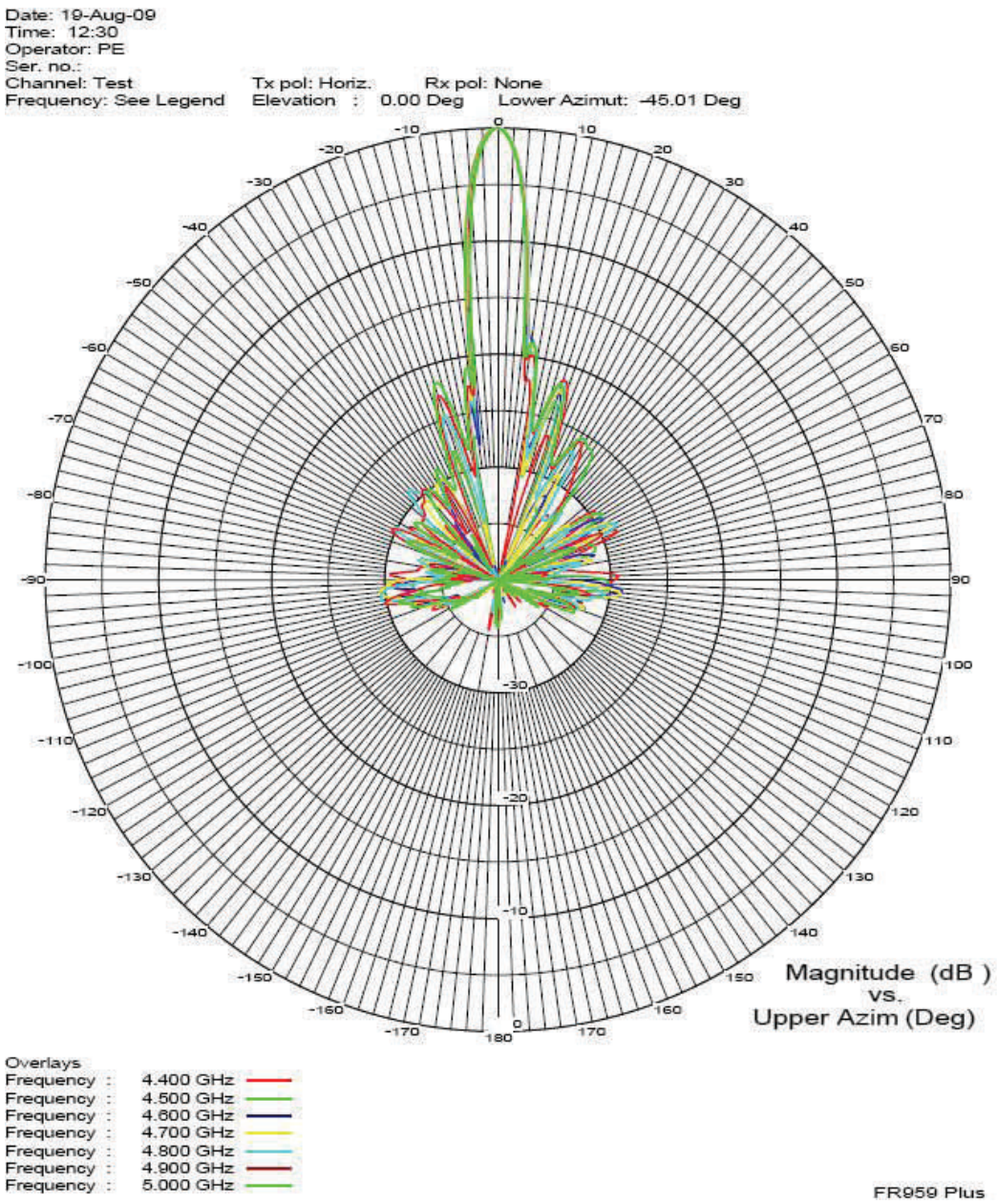
Frequency range	4.4-5.0 GHz
VSWR	≤ 1.5
Connector	N - female 50Ω or customer specified
Power rating	20 W
Gain	> 27dBi
Beam width	5° @ ± 3dB
Polarisation	Vertical or Horizontal
Front to back ratio	> 30dB
Cross Pol. rejection	> 20dB

Mechanical specifications:

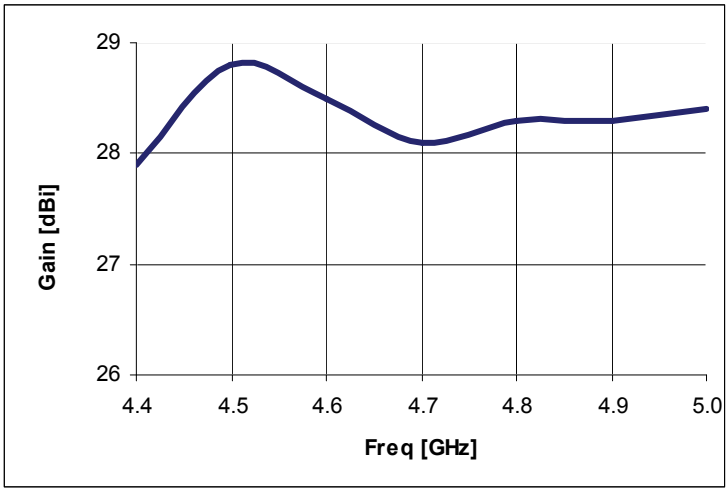
Design	Primary feed parabolic dish antenna. Dish and fixing made in aluminium and feed in fibreglass laminate.
Diameter	Ø 0.77m
Weight	5.2kg
Wind load	1000N @ 45m/s
Finish	Polyurethane lacquer
Temperature range	-55 °C, +71°C; -67 °F, +160 °F
Mast interface	Mast with Ø 39.5mm male spigot

Removable feed for easy transportation

Radiation pattern



Gain Curve



PAN & TILTS_

MOOG QUICKSET

QPT-90 Series

Pan & Tilt Positioners

The QPT-90 Series of Pan & Tilt Positioners is a high performance and versatile platform designed for a wide variety of positioning and sensor support applications. The QPT-90 supports payloads requiring 90 foot pounds of torque.

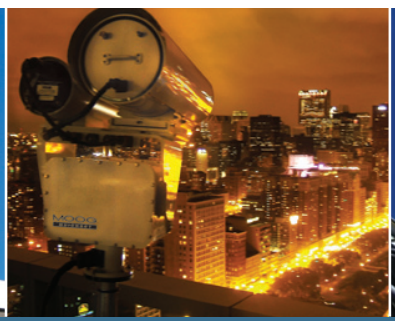
Multiple models are available to fit your specific requirements. Integrated control processor (IC) units feature dual sensor serial control, lens drive and power supply interfaces making sensor integrations quick and easy. Serial IP units communicate via networked communications or dedicated joystick controller (Unicom™ controller). Analog units are effective solutions where simple command and control are required without a PC. The Sentry line takes our products to the next level by integrating stepper motors into the design for higher precision movements and broader speed control. Universal configurations provide internal and external payload interfacing. Our RF units provide one to three pass-through channels for up to 18GHz bandwidth performance.

Key Features

- Payloads up to 90 lb-ft (122 Nm)
- Analog driven or Digital Serial Integrated Controller (IC) models
- Mounting platforms include plain formed table top, table top with single tilt-axis connector and 4 connector Universal models
- Internal wire table top for IC or pass-through sensor wiring
- Fixed, Inverted or Mobile Installations
- Mil-Spec Connectors
- Tough metal housing and gearing for durability in harsh environments
- Marine configuration that meets IP-67 standards
- RF pass-through connectivity (RF rotary joint, 1-3 channels)
- Thermostatically controlled heaters standard

Sensor Integration

- Multi-Spectrum Cameras (Visible/NIR/SWIR)
- Thermal Imagers
- IR and Visible Illuminators
- Laser Range Finders
- Communication Antennas
- Acoustic Devices



QPT-90 Series

Serial IP Features
Available with DC brush or stepper motors
Microprocessor control
Software controlled with status feedback
Serial Communication: RS232/422/485 and IP
Control Protocols: Moog QuickSet and Pelco D
2 programmable tours and 32 presets

Universal Features
Pass-through wiring
Full feature serial control of sensors
Motor drivers for camera lens zoom and focus control
2 Auxiliary relay controls for wipers, illuminators, laser range finders, etc.

Analog Features
Simple command and control with one controller for one positioner
Azimuth/Elevation position feedback output
Power supply integrated into controller

Standard Performance	
Load Capacity	90 lb-ft (122 Nm) maximum
Operating Voltage Range	24VDC (±4VDC)
Total Power	Pan & Tilt Axes: 7.5A pk, 2.0A continuous at 24VDC Heater: 4.4A at 24VDC Standby: <0.8A at 24VDC (no heater current)
Pan-Axis Range	360° continuous rotation (slip ring) 435° (±217.5°) (non-slip ring)
Pan-Axis Speed	0.005° – 25°/sec
Tilt-Axis Range	180° (±90°)
Tilt-Axis Speed	0.005° – 8°/sec at 90 lb-ft
Internal Heater	Thermostatically controlled 0°C on/1.7°C off (32°F on/35°F off)
Operating Temperature	Without Heater: -15°C to 55°C (5°F to 131°F) With Heater: -30°C to 55°C (-22°F to 131°F)
Rotational Limits	Fixed tilt hard limit, adjustable soft limits on both axes
Feedback	Optical Encoders (0.01° readout)
Repeatability	0.25° (0.05° on Sentry models)
Duty Cycle	20%
Motor Type/Drive	Stepper (Sentry) and DC Brush
Communication to Pan & Tilt	RS232/422/485, IP Ethernet: 10/100 Base-T
Communication to Sensors	RS232/422, Ethernet Pass-Through
Control Protocol	Moog QuickSet or Pelco D
Connector Specifications	Mil-Spec grade used on all configurations
Load Connector Interfaces	1 Mil-Spec connector at tilt axis (certain models) 4 Mil-Spec connectors on Universal tilt table top
Materials	Housing 6061-T6 Aluminum, stainless steel hardware, permanently sealed radial ball bearings.
Finish/Color	White powder coat paint over alodined chromate for corrosion resistance standard. Other colors and CARC available upon request
Weight	37 lbs (16.8 kg) to 75 lbs (34 kg) depending on model
Dimensions	See page 4
Test Cable and Software	6 ft test cable and software included with all IC and Sentry configurations

Note: Test software compatible with Windows-95 SP2, 98, ME, 2000 and XP version. Not compatible with NT. Moog QuickSet control protocol documentation supplied. Different models may vary.

QPT-90 Sentry Universal 4-Port Payload Connectivity*

QPT-90 Marine Tilt A/B Payload Connectivity**

Standard Housing (FT)

Serial/IP Configuration

	DC Brush-Type Motor Configurations		Stepper Motor Configurations (Sentry)	
	12 VDC	24 VDC	24 VDC	48 VDC
Pan Speed Range (deg/sec)	1° – 10°	.25° – 8°	0.005° – 30°	0.005° – 45°
Tilt Speed Range (deg/sec)	1° – 3°	.1° – 3°	0.005° – 8°	0.005° – 20°
Weight	37 lbs (16.8 kg) standard configuration, 75 lbs (34 kg) marine configuration		75 lbs (34 kg)	75 lbs (34 kg)
Number of Connectors	1 or 4 - depending on model		4	4

Note: Speed ranges dependent on model, weight and payload configuration - contact factory for details

Analog Configuration

	12 VDC	24 VDC	115 VDC	24 VAC	115 VAC
Pan Speed Range (deg/sec)	0.5° – 10°	0.3° – 8°	0.3° – 8°	8°	8°
Tilt Speed Range (deg/sec)	0.1° – 3°	0.1° – 3.5°	0.1° – 3°	3°	3°
Motor Type	DC Brush	DC Brush	DC Brush	AC Brush	AC Brush
Weight	37 lbs (16.8 kg)				

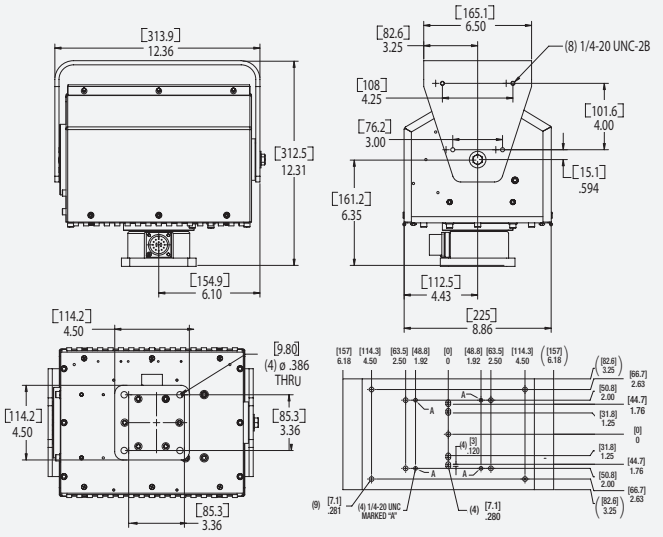
Note: Speed ranges dependent on model, weight and payload configuration - contact factory for details

* Note:
4-Port Payload Connectivity
2-Channel: Internal processor payload serial control, camera lens drivers / feedback input, Ethernet, payload power supply, video coax to base connector wiring.
2-Channel: Payload pass-through wiring for customer supplied payload interfacing including Ethernet, power, serial control, video coax to base connector wiring, and more.
(See details in Moog Quickset Universal Pan/Tilt data sheet)

** Note:
Tilt A, Single Channel Payload Connectivity:
Internal processor payload serial control, camera lens drivers / feedback input, Ethernet, payload power supply.
Tilt B, Single Channel Payload Connectivity:
Payload pass-through wiring for customer supplied payload interfacing. Includes base to tilt connector wiring for Ethernet, power, serial control, video coax to base connector wiring, and more.

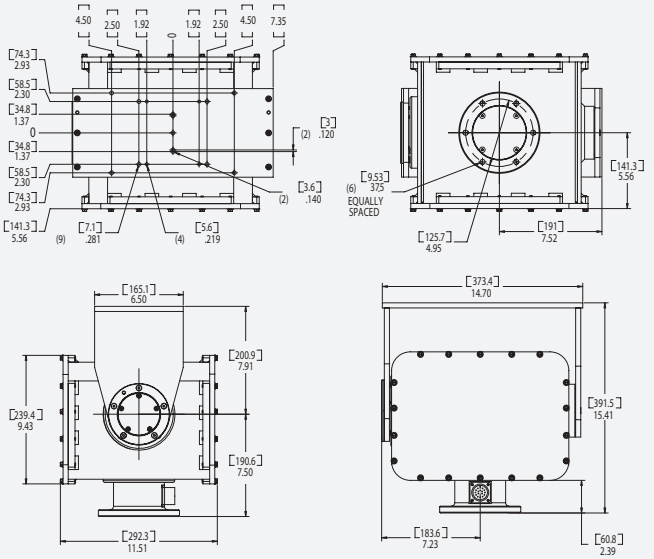
QPT-90 Dimension Drawings

Standard Housing



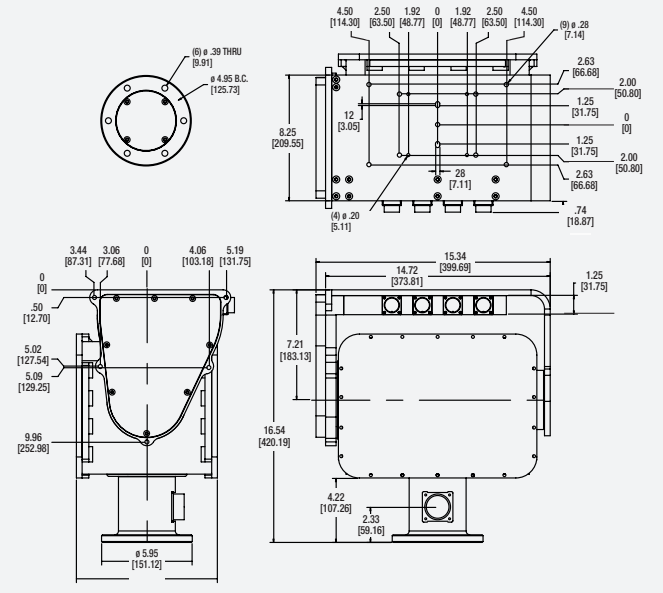
Dimensions are in Inches [mm]

Marine Housing



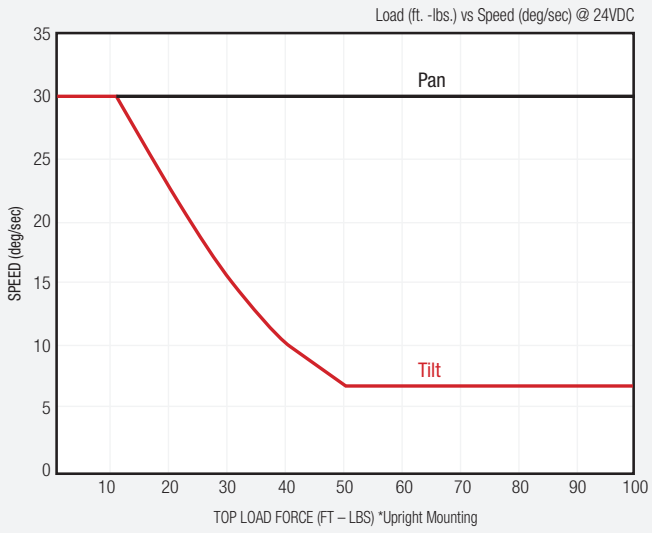
Dimensions are in Inches [mm]

Sentry Universal



Dimensions are in Inches [mm]

Sentry 90 Torque Curve



MOOG QUICKSET

3650 Woodhead Drive,Northbrook, IL 60062-1895 USA
+1.847.498.0700 Fax: +1.847.498.1258
www.quickset.com

Form 500-622 0213