Master Thesis

Universiteit Utrecht

Indoor Localization by UHF RFID Technology

An approach from probability and statistics

Jan Boogert

supervised by
Dr. Gerard Sleijpen (UU)
Ir. Matthijs van der Weg (Intellifi)

April 11, 2016

intellifi
localization & identification | easy & affordable

Abstract

Radio Frequency Identification (RFID) is a rapidly growing market for many applications. Like bar codes, it enables automatic identification of physical objects, but it does not require a line of sight, and it has a much bigger range. This bigger read range makes it suitable for localization purposes. In our research we present an efficient localization procedure for indoor environments, based on signal strength measurements of the radio communication.

The received signal strength is very noisy, an exponential probability distribution appears to model it well. Hence, the localization can be realized by applying the method of maximum likelihood to signal strength measurements at multiple RFID readers. Maximizing the likelihood is an optimization problem which we solve numerically by a quasi-Newton method. Furthermore, we show by simulated signal strength measurements the dependence of the localization accuracy on the physical parameters. Finally, we check the results by solving the localization problem by an alternative method, neural networks, which is a black box solver.

The research was done at Intellifi company, which provided a modern RFID system.

Contents

1 Introduction

The main challenge of this study is to employ RFID technology for indoor localization purposes. However, RFID technology was originally designed for the purpose of automated identification, which is slightly different. In this chapter, we will first describe how RFID suits this purpose, Section 1.1. Thereafter, we give a brief summary of the historical development of RFID technology, Section 1.2. In Section 1.3, we will explain briefly the technical design of RFID equipment. The information in these more general sections is taken from [6].

This study was done at the Intellifi company, which provided RFID equipment. The particularities of this equipment are explained in Section 1.4. Then we are ready to post our problem in more detail, Section 1.5. In the concluding Section 1.6, we describe shortly the proposed solutions in literature, and argue which solution we studied in our research.

1.1 Automated identification and RFID

In present-day life, computers create plenty of possibilities to simplify everyday problems. A big power of computers is their ability to process quickly huge amounts of data. However, in many cases it is still a challenge to feed the right data automatically to a computer system. One such case is automated identification (auto-ID). The aim of auto-ID is to identify physical objects and gather data about them without human involvement. Examples are product identification at a store checkout, face recognition, voice recognition, localization of objects in a warehouse, tracking properties in an hospital, and so on.

A straight-forward solution of these challenges is to capture image or sound data in a digital file, and recognize the object using computer algorithms. This is the usual way of operation for recognition of hand-written or printed text, finger print pattern recognition, face recognition, recognizing spoken language, and other problems with a clear visible or audible component. However, recognition by visible ways is not always easy. For example, a store can sell multiple types of the same product. Moreover, the computational costs of processing images are typically big.

To overcome these problems, bar codes have been invented. A bar code is a representation of data, designed to be easily readable by machines. Bar codes operate basically as follows: one prints or attaches a bar code on every object of interest, and by imaging or scanning the code an identifying number is easily extracted. So bar codes help visible recognition for computers.

Bar codes have become the by far most common way of identifying a physical object. In the form of the of the Universal Product Code (UPC) and its modern descendents (like the EAN-13 code), they are everywhere around us. Originally bar codes consisted of parallel lines of varying widths at varying spacings. The data representation was clearly one-dimensional. Later on, two dimensional bar codes were developed, which can capture more information on the same area. A typical exam-

ple is the QR-code. Although two dimensional bar codes do not necessarily feature bars, these are still called 'bar codes', because they do not change the basic principle of the original bar code.

An important deficiency of bar codes and optical methods in general is that they require a clear line of sight. Not only objects, but also dirt, paint, ink, and other ordinary substances, can distort or deface bar codes. To remedy some of the disadvantages of optical identification, radio-frequency identification (RFID) has been developed, which employs electromagnetic waves to communicate with a reading device. The benefits are clear: the read range is typically much bigger than for bar codes, a clear line of sight is not needed any more, and there is less sensitivity for dirt. In this way, RFID technology is very well suited for the purpose of automated identification. Moreover, the cost of RFID equipment have declined over the years, such that this technology becomes economically feasible for more and more applications.

1.2 Evolution of RFID technology

Historically, RFID and similar technologies have existed already since the 1940s. During the Second World War, radar technology was developing rapidly for the purpose of identifying and localizing enemy's aircrafts and seacrafts, and to distinguish them from friendly ones. In this way, much research in the field of radio frequency electronics was done, which continued through the 1970s. This research included the development of devices powered by radio waves and the technique to modulate and backscatter received radio waves.

In the late 1960s the first RFID applications became commercial available. For instance, anti-theft and security equipment were developed, which included anti-theft gates placed at the doors of stores. These systems were simple 1-bit systems, which means that detection of the presence of RFID tags was possible, rather than identify them.

Next, by the emergence of information technology since the 1970s, efficient communication networks became available. In this way, the 1980s brought the first widespread commercial RFID systems. These were rather simple systems like livestock management, keyless entry and personnel access systems. In the late 1980s the first toll applications were implemented for road traffic.

By breakthroughs in materials science technology during the 1990s, RFID tags became much cheaper to manufacture. Hence, RFID became economically feasible for many new applications. The introduction of standards enabled interoperability between systems of different manufactures. In the early 2000s it had become clear that very cheap ($0.05) tags would be possible, and that RFID technology could some day replace bar code systems. Both Wal-Mart, the world's largest retailer, and the US Department of Defense, the world's largest supply chain, required suppliers to start employing RFID technology by 2005. This constituted an enormous market for RFID, and many other manufacturers and retailers followed.

1.3 The RFID System

An RFID system consists basically three components. The first component is a *tag*, which is sometimes called a transponder. It is composed of a chip, an antenna, and for some systems a battery. The second component is a *reader*, also called an interrogator or a read/write device. It is composed from an antenna, an RF electronics module, and a control electronics module. The third component is the *controller* or *host*, which most often takes the form of a PC running database and control software.

These components interact normally as follows. The tag and the reader communicate by using radio waves. The tag transmits its identification number, and probably some more information. This data is picked up by the reader which sends it to the host by some network connection. Then the host visits a database which contains all known tag identification numbers and the associated physical objects. In this way, the host knows what the meaning is of the data from the reader, and can take action.

The communication between the tag and the reader can take some different forms, depending on the kind of tag. We can distinguish three types of tags, *passive*, *semi-passive* and *active*. Passive tags do not have a battery, but can transmit data by using energy from an interrogating signal which is sent by the reader. This interrogating signal is modulated and is broadcast by the tag, which is commonly called *backscattering*. Because the activity of these tags depend on the interrogating signal transmitted by the reader, these tags are called *passive tags*.

Unfortunately, the effective range of passive tags is rather short, and often too short. Therefore, semi-passive tags are invented. These tags use the same communication technique as passive tags, but are equipped with a battery to achieve a more powerful signal and hence a larger effective range. So these tags need still to be activated by an interrogating signal, but are battery assisted. Therefore, they are called *semi-passive tags*.

There are also tags that periodically broadcast, which is not controlled by the reader. These are called *active tags*, and need of course a battery. Passive tags are typically much cheaper than semi-passive or active tags.

There are four typical frequency bands where RFID systems operate, 125-134 kHz in the low frequency (LF) band, 13.56 MHz in the high frequency (HF) band, 860-960 MHz in the ultra high frequency (UHF) band, and 2.4 GHz and above in the super high frequency (SHF) or microwave band. For example, the chipcards in public transport in the Netherlands contain chips which operate at 13.56 MHz.

In general, higher frequencies allow faster data transmission and smaller tags, but worse performance near metal or liquids. Furthermore, passive RFID tags which operate at high frequencies

(UHF and microwave) are potential to be very cheap, whereas tags at low frequencies (LF en HF) are relatively expensive. For reference, common analog radio AM (or medium wave) broadcasts use radio frequencies roughly between 500 kHz and 1700 kHz. The more enhanced FM radio broadcasts use radio frequencies between 87 and 105 MHz. Television and mobile phones have use many separated radio bands, mostly situated between 400 MHz and 3000 MHz.

1.4 The Intellifi system

In the Intellifi system, the reader or interrogator is called *spot*, and can take on many tasks from the host. Furthermore, usually a internet service looks after the controller tasks, such that an average user does not need a PC to employ an Intellifi system.

The Intellifi system supports passive UHF RFID tags, and active Bluetooth tags, which operate at 2400-2480 MHz, which is the microwave band. This corresponds wavelengths of 0.33 meter and 0.12 meter respectively.

Besides the regular communication between spots and tags, the spots are also able to measure the strength of the signals they receive from the tags. Under idealized conditions, this signal strength is related to the distance between spot and tag by the Friis equation (to be discussed in next chapter, Section 2.3). In this way, the distance can be estimated. However, in our indoor application of RFID, the measured signal strength is heavily disturbed.

There exist also other quantities in addition to the signal strength, which can be used to get more information than the presence of a specific tag. Examples are the phase (see Section 2.5) and angle of arrival. However, the Intellifi spots are currently not able to determine these quantities, so we have to focus on signal strength measurements only.

1.5 Problem statement and research outline

The main objective of this research is to obtain the most accurate localization routines for RFID technology in common industrial indoor circumstances, using signal strength measurements only. Because the signal strength measurements are often heavily disturbed, we have to remove that noise. We will evaluate both the statistical method of maximum likelihood estimation and neural networks for that purpose, and compare their results.

The research outline is as follows. As explained in previous section, we can use signal strength measurements to compute the distance between tag and spot. The first aim of this research is to determine how the signal strength is related to distance. For this purpose, we will consider the physical properties of the radio signal in Chapter 2. Here, we will also see why signal strength is a noisy measure to estimate distance. In order to remove that noise, we will try to model the received signal strength by random variables, and apply statistical tools, which are discussed in

Chapter 3. Here we will encounter the method of maximum likelihood, which will take a central position in the sequel.

Because maximizing likelihood is an optimization problem, we discuss the most important optimization techniques in Chapter 4. After these general chapters we are ready to model for the received signal strength by some random variable, and determine its relation to the distance between transmitter and receiver. This is the subject of Chapter 5.

In Chapter 6 we will come to the localization problem, which is the main objective of this research. Using the method of maximum likelihood, we will derive a general localization algorithm. Using simulated data, we estimate the behavior of the localization problem with respect to its physical parameters. We will also apply the localization algorithm to real measurements to check the reliability of the results.

In order to know how to appreciate the results of our localization algorithm, we will also apply neural network simulators to the localization problem. Because neural network simulators are black-box solvers, they provide optimal localization results, but at the cost of the need of training data. This is discussed in detail in Chapter 7.

1.6 Localization by RFID in literature

In literature, there is already spent much effort concerning localization by RFID. We can distinguish basically the following solutions (as described for example in [1], [2], and [3]).

1. *Zone detection.* Compare signal strength measurement at some spots. The tag is likely closest to the spot which holds the strongest signal, and so the tag is positioned in the 'zone' surrounding that spot. This method is mainly used for passive tags.
2. *LANDMARC* (LocAtioN iDentification based on dynaMic Active Rfid Calibration). For each spot, compare the signal strength received from the tag to be located, with a number of reference tags with known locations. The tag is located near the position of the reference tag which resembles the measured signal strengths at most.
3. *Received Signal Strength.* This method uses the signal strength measurements to estimate the distance between the tag and the spots. Using measurements of multiple spots, the location of the tag can be found. In 2D-applications, at least three spots are needed, in 3D-applications, at least four spots are needed. This localization process is called *trilateration.*
4. *Angle of Arrival Triangulation.* When a reader can measure somehow from which direction the signal from the tag arrives, angle of arrival measurements of at least two readers can be triangulated in order to find the location of the tag.
5. *Time of Flight.* This method uses time measurements to estimate the distance between tag and reader. Because a radio signal needs some time to cover a distance (it travels by the speed of light), the propagation time of the signal is proportional to the distance. So if the

tag communicates the time of transmitting and the spot determines the time at which the signal is received, the distance can be computed. Subsequently, the location of the tag can be derived by trilateration.

6. *Differential Time of Arrival.* In this variant of the previous method, it is not needed that the tag tells at which time it broadcasts the signal. In order to localize the tag, multiple spots measure the time at which they receive a signal from the tag. Now by comparing those times, the location of the tag can be derived. This method needs an extra receiver.
7. *Phase Difference of Arrival.* The distance between spots and tags can also be estimated by phase measurements. By comparing the phase of arrival at different frequencies, the distance modulo some wavelength factor can be computed.

Every method has its own characteristics when applied in real environments. In practice, accuracy, costs and efficiency considerations will determine which method should be chosen. Of course, by combining two or more localization techniques, better results can be expected. In literature, following statements are supposed.

1. Time-based methods are most accurate [7], because they can filter out multipath effects. The main draw back of this method is that they need very precise time delay measurements (we need nanoseconds). This can only be realized by costly equipment. For comparison, time of arrival is the method which is used by the well-known Global Position System (GPS). [9]
2. Zone detection is the easiest method, but is only adequate when we need only zone detection, and the zones are large and far enough apart from each other. [4]
3. Techniques based on signal strength, suffer heavily from multipath environment. [3]
4. Angle of Arrival, the time-based methods, and Phase of Arrival perform poorly if there is no line of sight and so no direct path of the signal. [3]

From these statements it is clear that we should not expect too much from localization by RFID technology. All localization methods have their draw backs, and exact localization seems to be impossible in general. Now we might think that doing some scene analysis would solve these problems. But a scene analysis is not very desirable, because only by strict conditions a scene analysis makes sense, and during operation the scene can change by moving objects. LANDMARC solve these problems partially, but it is a problem to position its reference tags appropriate. [15]

All considered, we chose in our study to use simply signal strength measurements in a statistical approach, see Chapter 5. This corresponds essentially to method 3, such as described above. Here we try to improve the localization accuracy by repeated measurements in a mathematically responsible way, the maximum likelihood estimate (see Section 3.5). In this decision played a role the fact that the Intellifi equipment we used was only able to perform measurements on signal strength. However, the maximum likelihood estimate can easily combine quantities of distinct nature, and so the localization engine we constructed in this research can easily be improved once the equipment offers also the opportunity to measure quantities like angle of arrival or phase of arrival.

2 Physical considerations

Radio communication is a very commonly used technology, which enables wireless communication. Even a line of sight is not essential, which makes radio communication very attractive. However, there are some limits. Therefore, in this chapter we will explore the characteristics of radio communication. Section 2.1 discusses the characteristics of radio waves, which carry information from sender to receiver. Radio antennas, the instrument to generate the radio waves, are discussed in Section 2.2. Section 2.3 and Section 2.4 explain how radio waves lose energy over distance. In Section 2.5 it is how the phase information of radio waves is related to distance, and how it can be used to estimate the distance between sender and receiver. This chapter concludes with a description of the Intellifi equipment in light of the preceding sections.

2.1 Radio waves

To exchange data, radio communication uses *radio waves*, electromagnetic waves, which travel by the speed of light. These transverse waves have some properties: *wavelength*, *frequency*, *amplitude*, *phase*, and *polarization*. These properties are are explained below.

Frequency and wavelength

The frequency f of a radio signal is the number of waves which pass by per unit time. The wavelength λ of a radio signal is the distance between the waves. These are connected by the speed of the wave. In vacuum, the wave travels by the speed of light $c \equiv 299\,792\,458$ meters per second. Then we have the relation

$$\lambda = \frac{c}{f}, \tag{1}$$

with λ in meters, and f in hertz (Hz).

Amplitude and amplitude modulation

The amplitude A of a radio wave is the 'strength' of the signal. It is a measure of the change of the radio signal in a single period. The most common measure is the peak amplitude, the maximum displacement in the wave. However, in electrical engineering the root mean square (RMS) amplitude is often used. It is defined as the square root of the mean of the squared displacement with respect to the rest state.

So assume that the wave oscillation can be described by $t \to u(t)$, where the oscillation has frequency f, and the rest state is u_0. Then the peak amplitude is

$$A_{\text{peak}}(u) \equiv \max_{t\in[0,1/f)} \big|u(t) - u_0\big|, \tag{2}$$

and the RMS amplitude is

$$A_{\text{RMS}}(u) \equiv \sqrt{f\int_0^{1/f} \big(u(t) - u_0\big)^2 \,\mathrm{d}t}. \tag{3}$$

When measuring the signal strength of some RFID signal, one has to take into account that RFID communicates data by amplitude modulation. Here, zeros and ones of the binary data are encoded by specific modulation patterns in a carrier wave. So the amplitude of the radio wave is not constant. Therefore, the measurement should be taken so long that it is clear where there are damplings and reinforcements in the signal.

Phase

The phase offset with respect to some reference wave, is some angle

$$\varphi \equiv 2\pi\Big(\frac{r}{\lambda} - \Big\lfloor \frac{r}{\lambda} \Big\rfloor\Big) = 2\pi\Big(\frac{r}{\lambda} - k\Big) = 2\pi\Big(\frac{rf}{c} - k\Big). \tag{4}$$

Here, $\varphi \in [0, 2\pi)$, r is the distance to the transmitter, and $k \equiv \lfloor \frac{r}{\lambda} \rfloor \geq 0$ is an integer. Here, phase offsets created by wires of transmitting and receiving devices are neglected. The phase is measured in radians.

Polarization

Because radio waves are transverse waves, the oscillations occurring in the wave are perpendicular to the direction in which the wave travels. So in a (x, y, z)-space where the wave propagates in z-direction, the wave oscillations are directed in (x, y)-plane. The way in which these oscillations behave, is called the *polarization* of the wave. When all oscillations in the wave are on the same line in (x, y), the wave is called to be *linear polarized*. It is also possible to change the direction of the oscillations together with the oscillations themselves. In this way we get circular or elliptical polarization. In case that the direction of the oscillations changes at random, the wave is called to be unpolarized.

It is important to think about the polarization of the radio waves, because the antenna performance of a radio receiver can be much less when it is not correctly aligned with the polarization of the radio wave.

2.2 Radio antennas

An antenna is an device which is used to transmit or receive radio waves. When used for transmitting, it is fed by some alternating current at some frequency. This current causes electrons in the antenna to oscillate, which generates electromagnetic radiation. This radiation is transmitted into the space surrounding the antenna.

When an antenna is used for receiving, the system acts backwards: electromagnetic waves from the surrounding area cause electrons in the antenna to oscillate, which generates an alternating current in the antenna, which is fed to the processing unit in the radio receiver.

Currents from receiving antennas are commonly much smaller than in transmitting antennas. In order to receive enough, antennas together with the electrical network attached to it are designed to be resonant at the frequency of the signals it is looking for.

How much radiation is received from every direction, or transmitted to every direction depends on the shape of the antenna. Therefore, antennas exist in many different models, which have different characteristics. For example, directional antennas transmit most power in a specific direction, and others are designed to be omnidirectional. Moreover, it can be shown that it is impossible to construct an isotropic antenna, an antenna which transmits power equally in all directions, and which is equally sensitive to signals from all directions.

The antenna *gain* is the technical term which describes the directional characteristics of an antenna. It also incorporates antenna's efficiency. When we think of the situation where a far-field source transmits radio waves (which generates electric power in the antenna), the antenna gain is defined to be the ratio of the power produced by the antenna to the power produced by a hypothetical lossless isotropic antenna. So the antenna gain depends on the direction from which the radio wave arrives.

The antenna gain is related to the so-called *effective aperture* of the antenna. The effective aperture is a measure of how effective an antenna is at receiving power of radio waves. This depends on the wavelength of the radio wave: shorter wavelength leads to less effective aperture. An hypothetical lossless isotropic antenna has effective aperture $\lambda^2/(4\pi)$, where λ is the wavelength. For real antennas, the effective aperture is $G\lambda^2/(4\pi)$, where G is the antenna gain. [6].

In a situation where an antenna is not correctly aligned with the polarization of a wave, the generated current becomes smaller as well. This problem can be avoided by using multiple antennas at different orientations, as we will show later on.

2.3 Path loss

There are a few reasons why a receiver usually receives much less energy from a radio wave than the transmitter originally transmitted. The most important one is the natural expansion of the wave: when a radio wave propagates through free space, the wave spreads in the shape of an ever-increasing sphere. In this way, the energy of the radio wave in a unit part of sphere surface decreases along the total surface area of the sphere.

We can also distinguish absorption losses (for example by air), refraction or defraction losses by obstacles, diffraction losses when the wave passes through a slit, and multipath fading. *Multipath* is the coincidence that one signal travels to the receiver along more than one way. It can travel to the receiver along the line of sight, but for example also via some wall. This route is longer, and therefore arrives at the receiver at an other phase. Hence it may damp (or strengthen) the signal

from the direct path. This effect is called multipath fading.

Usually, losses by absorption, refraction, defraction, or multipath, are usually less relevant, but we will come back to them later on.

We can compute the path loss by the natural expansion of the wave as follows. The surface $A(r)$ of a sphere with radius r is given by

$$A(r) = 4\pi r^2.$$

Now let us assume that the energy of the radio wave is equally divided over the sphere. Then a receiver with an antenna of unit gain situated at distance r from the source of the radio wave, can expect to receive only the fraction $1/(4\pi r^2)$ of the original total transmitted energy: the rest is simply transmitted in other directions.

However, the fraction $1/(4\pi r^2)$ is no perfect rule for path loss. As pointed out in the previous section, antenna gain and antenna aperture play a role. We plug these factors in, and we get the among radio technicians well-known *Friis transmission equation*

$$P_R = P_T G_R G_T \Big(\frac{\lambda}{4\pi r}\Big)^2. \tag{5}$$

The meanings of the characters and their units are as follows:

quantity	meaning	unit
P_R	Induced power at the receiving antenna	watt (W)
P_T	Transmitted power at the transmitting antenna	watt (W)
G_R	gain of the receiving antenna	(dimensionless)
G_T	gain of the transmitting antenna	(dimensionless)
λ	wavelength of the radio-wave	meter (m)
r	the distance between transmitter and receiver	meter (m)

The nice thing about Friis equation is that it relates the received power to the distance between transmitter and receiver. When multiple receivers at known locations pick up a signal from the same tag, this equation allows to estimate the location of the tag.

In technical texts, power and gain are often expressed on the logarithmic deciBel (dB) scale, because it can express both large and very small values in a short range of numbers. This decibel scale is simply defined by 10 times the base-10 logarithm of a number. For non-physicist, the use of this scale in technical texts can be confusing, because the quantities in dB-scale are written by the same symbols as the quantities in the ordinary scale. In this way, the Friis equation is noted as

$$P_R = P_T + G_R + G_T + 20\log_{10}\Big(\frac{\lambda}{4\pi r}\Big),$$

where P_R and P_T have unit dBm (1 dBm $= 10\log_{10}(P)$, where P is in milliwatt (mW)), and G_R and G_T have unit measure dBi, which stands for dB(isotropic). To avoid confusion we will use always the ordinary units as in (5) in this writing.

The ideal conditions under which the Friis equation applies, are (according to [6]):

1. Antennas must be in the so-called far field, which means most of the time $r \gtrapprox 2\lambda$. For relatively big antennas (larger than half of the wavelength of the radiation they emit), there should be more distance.
2. The bandwidth is narrow enough that a single value for the wavelength can be assumed.
3. The antennas are correctly aligned and polarized.
4. The space around the antennas is unobstructed: so there is a clear line of sight, and there are no multipath effects.

Conditions 1 and 2 can be guaranteed relatively easy for the Intellifi system: with $\lambda \approx 0.33$ meter for UHF RFID tags and $\lambda \approx 0.12$ met for Bluetooth tags, $r \geq 0.6$ is acceptable. However, in practice, conditions 3 and 4 cannot be satisfied easily. Because we would like to attach RFID chips to moving objects, correct alignment and polarization cannot be guaranteed. Furthermore, we would like to apply the RFID technology in indoor environments, where obstructions and multipath effects are inevitable. In these situations, it is questionable whether Friis equation is still usable, because we can expect many perturbations on the received signal strength.

Fortunately, we can fix condition 3 by introducing an angle $\theta \in [0, \frac{1}{2}\pi]$, which represents the rotation with respect to correct alignment. Here $\theta = 0$ represents no rotation and so correct alignment, and $\theta = \frac{1}{2}\pi$ represents total misalignment. Then the Friis equation (5) becomes

$$P_R = P_T G_T G_R \Big(\frac{\lambda \cos\theta}{4\pi r}\Big)^2. \tag{6}$$

In case that the rotation is not known, we can estimate θ by using two identical antennas which are rotated relative to each other, and compare the signal strength induced on both antennas.

Because we would like to apply the radio communication in indoor environment, we are unable to meet condition 4, and there exists no general fix. Walls, floor and ceiling cause refraction and defraction losses and multipath effects, which can affect the received signal strength intensely. By doing environment analysis, corrections can be derived for specific environments, but this is a intensive task. Moreover, the environment can change during operation, for example when objects are moved in the room where the RFID equipment is situated. Depending on the nature and size of obstructions in the line of sight, the receiver will find a lower signal. And when a clear line of sight is provided, there are always multipath effects.

Concerning multipath, if the route via the wall is some integer wavelengths plus a halve wavelength longer than the route along the line of sight, it is completely out of phase at the receiver. And

hence, it will damp the observed signal strength. Of course, when the route via the wall is exactly some integer wavelengths longer then the route along the line of sight, it is in phase at the receiver, and hence it will increase the observed signal strength.

Concerning absorption losses, these depend on wavelength and the kind of material through which the radio wave propagates. Absorption by air is insignificant for radio frequencies we use in our research, but at extreme high frequencies above 30 GHz, attenuation by oxygen and water vapor can be significant. [14]

2.4 Passive tag path loss

When using passive tags, it is wrong to use simply (6), because the power transmitted by the tag depends on the distance to the reader. As already stated, passive tags need an interrogating signal from the reader, to get activated. Then the tag starts to transmit data by modulating and backscattering the signal from the reader. In this way the power transmitted by the tag is bounded by the power it receives from the reader.

Of course, the power that the tag receives from the reader can be described by (6). And the path loss of the backscattered signal can be described by (6) as well. Now let us assume that the spot transmits $P_{\text{spot},T}$ power, then the tag receives

$$P_{\text{tag},R} = P_{\text{spot},T} G_{\text{tag}} G_{\text{spot}} \left(\frac{\lambda \cos\theta}{4\pi r} \right)^2$$

power. The tag backscatters $P_{\text{tag},T} = T_b P_{\text{tag},R}$ power, where T_b is the backscatter transmission loss factor, which is typical around $\frac{1}{3}$ [6]. Then the spot receives from the tag

$$P_{\text{spot},R} = P_{\text{tag},T} G_{\text{tag}} G_{\text{spot}} \left(\frac{\lambda \cos\theta}{4\pi r} \right)^2 = P_{\text{spot},T} T_b G_{\text{tag}}^2 G_{\text{spot}}^2 \left(\frac{\lambda \cos\theta}{4\pi r} \right)^4. \tag{7}$$

So we find theoretically $P_{\text{spot},R} \sim r^{-4}$. However, here we do not count for nonlinear effects in the tag electronics. A tag needs to receive some minimal power, to get activated. The tags we use, are just optimized for that minimal level. If a tag receives very much power, it protects itself by intentionally losing some energy. Because it is difficult to quantify these effects, we keep temporary the exponent to be an unknown parameter γ. In this way we arrive at

$$P_{R,spot} = P_{T,spot} T_b G_{tag}^2 G_{spot}^2 \left(\frac{\lambda \cos\theta}{4\pi r} \right)^\gamma. \tag{8}$$

2.5 Phase information

Next to the received power, the phase offset with respect to some reference wave is related to the distance between transmitter and receiver as well, see (4). However, this phase information is relative to some reference wave, which is still unknown. Moreover, the solution $r = \lambda\varphi/(2\pi) + k\lambda$ is ambiguous: in the UHF band, λ is approximately 0.33 meter, while the passive tags can operate

at distances r up to 10 meter. For Bluetooth tags, the situation is even worse: here wavelength λ is approximately 0.12 meter, while the active Bluetooth tags can operate at distances up to 100 meter.

To avoid these problems, we can obtain the phase information for two (or more) distinct radio frequencies. So if we get for $i = 1, 2$ at frequency f_i phase (4)

$$\varphi_i = 2\pi\Big(\frac{r}{\lambda_i} - \Big\lfloor \frac{r}{\lambda_i} \Big\rfloor\Big) = \frac{2\pi r f_i}{c} - 2\pi k_i,$$

where $k_i \in \mathbb{Z}$ is some integer, and assuming $\Delta f_{12} \equiv f_1 - f_2 \neq 0$, we can compute the distance r by

$$r = \frac{c(\varphi_1 - \varphi_2)}{2\pi\Delta f_{12}} + \frac{c(k_1 - k_2)}{\Delta f_{12}} = \frac{c\Delta\varphi_{12}}{2\pi\Delta f_{12}} + \frac{ck}{\Delta f_{12}}$$

for some integer $k \equiv k_1 - k_2$. Here $\Delta\varphi_{12} \equiv \varphi_1 - \varphi_2$ is the observed phase difference. In terms of wavelengths, we get

$$r = \frac{\varphi_1 - \varphi_2}{2\pi\big(\frac{1}{\lambda_1} - \frac{1}{\lambda_2}\big)} + \frac{k}{\frac{1}{\lambda_1} - \frac{1}{\lambda_2}},$$

If $c/(\Delta f_{12}) = 1/(\frac{1}{\lambda_1} - \frac{1}{\lambda_2})$ is bigger than the maximum distance at which we can receive a signal from a tag, then we can simply take $k = 0$, and we have an unambiguous estimate of r. So it is attractive to take Δf_{12} small. However, decreasing Δf_{12} amplifies the sensitivity of the phase difference to noise, and thus yields less accuracy. So we should discover which frequency separation is adequate. Note that we can probably use more than two phase measurements to get more accuracy, and use more than two frequencies to cancel the ambiguity. It is also possible to use signal strength measurements to decide which value should be taken for k. Note that in the case of passive (backscattering) tags, we are actually measuring $2r$ instead of r. [10]

2.6 The Intellifi system

As already mentioned, Intellifi uses two types of RFID communication. In the first place, the Intellifi system supports passive UHF RFID tags, which operates at frequencies 860-930 MHz. For this purpose, every Intellifi spot has a square antenna plate, which is designed for linear polarized radio waves. The antenna plate is connected at two sides to the processing unit in the spot, and so the spots have effectively two antennas for UHF RFID, and they do not depend on one specific polarization angle. Concerning the gain of the antenna, signals are better transmitted to the front than to the side of the antenna. And so, signals are also better received from the front than from the side of the antenna.

Next to UHF RFID, Intellifi supports also active Bluetooth tags, which operate at frequencies 2400-2480 MHz. For this purpose, the Intellifi spots are equipped with a separate Bluetooth antenna, which is shaped as a folded wire. This results also linearly polarized radio waves. Because there is only one antenna, we can not correct for the signal strength decrease caused by incorrect alignment. Concerning the gain of the Bluetooth antenna, signals from the front are just less received than signals from the side. This is intentionally designed in this way, because the spots

are usually installed in the ceiling of the room of operation, with the front directed downwards. So transmitters in the front of the spot are always rather close to the spot. In this way, signals from Bluetooth transmitters from the front are easily received anyway. By having more gain to the side directions, signals from transmitters which are further away are better received, and so the effective reach is increased. This argument is not valid for passive UHF RFID tags, because the reach of these tags is much smaller, and so we prefer to focus on the tags which are in front of the spots.

Unfortunately, the Intellifi system is currently only capable to measure signal strengths, we have to focus on this quantity is our research. Future developments aim to detect phase information, and detection of the direction from which a signal originates.

3 Tools from statistics

The field of probability and statistics studies the properties of random variables. Many processes can be modeled by random variables, and in this way, we will try to model the received signal strength. In this chapter, we will present some basic definitions and theorems from probability and statistics. Readers can skip this chapter, and consult it whenever in later chapters we will refer back to the content here presented.

Section 3.1 contains the basic properties of random variables like probability distributions and expected value. Thereafter, Section 3.2 explains how to compute the probability distributions of transformed random variables. Section 3.3 explains how random variables can be simulated by a computer. In Section 3.4 we discuss the histogram, which is a usual method to determine the probability distribution of some dataset. Thereafter, Section 3.5 discusses the method of maximum likelihood estimation, which describes how parameters of a model can be estimated by data samples. The chapter concludes with Section 3.6, which describes the famous normal distribution, and Section 3.7, which discusses the exponential and Gamma distribution, because we will face it in Chapter 5. The information in this chapter relies mainly on [5].

3.1 Basic properties of random variables

In this section we describe shortly the basics of continuous random variables.

A *random variable* is a function from some probability space Ω to $\mathbb{R}$, which describes some numerical properties of outcomes in Ω. Random variables which can take on a continuum of values are called *continuous* random variables. Random variables which can take only a finite or countably infinite number of values are called *discrete* random variables. One can also imagine random variables which mixed continuous and discrete, but these are rather unusual. Here we will only discuss continuous random variables, because these fit the scope of this study.

The *probability space* Ω is a set of all possible outcomes of some experiment, where should think of *experiment* in very general sense. Subsets of Ω are called *events*. The probability space has also some *probability measure* P, which assigns to each event a number in $[0, 1]$ which is the probability that the event occurs.

Continuous random variables are characterized by their *probability density function*, which are defined as follows. Some arbitrary continuous random variable X has probability density function f_X, if for any numbers a and b with $a \leq b$,

$$\mathrm{P}(a \leq X \leq b) = \int_a^b f_X(x)\,\mathrm{d}x.$$

It is clear that the function f_X has to be an integrable function. Moreover, it has to satisfy $f(x) \geq 0$ for all x and $\int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1$.

The *cumulative distribution function* is another characterization of a random variable. This function F_X for some random variable X is defined by

$$F_X(x) = \mathrm{P}(X \leq x).$$

The cumulative distribution function is related to the probability density function by $F_X(x) = \int_{-\infty}^{x} f_X(y)\,\mathrm{d}y$, and so $f_X(x) = \frac{\mathrm{d}}{\mathrm{d}x}F(x)$ for all x when f_X is continuous. Other properties of F_X are $\lim_{x\to\infty} F(x) = 1$ and $\lim_{x\to-\infty} F(x) = 0$. Moreover, $\mathrm{P}(a \leq X \leq b) = F(b) - F(a)$. From this follows simply that F_X is non-decreasing.

Because random variables can be rather complicated, there are some helpful tools to characterize them. The first one is the *expected value*, which is defined by

$$\mathrm{E}(X) \equiv \int_{-\infty}^{\infty} x f_X(x)\,\mathrm{d}x$$

where X is some continuous random variable. The expected value is also often denoted by μ_X.

A second tool to characterize a random variable is the *variance*, which is a measure of the spread of the probability distribution around the expected value. The variance is defined by

$$\mathrm{Var}(X) \equiv \mathrm{E}((X - \mathrm{E}(X))^2) = \int_{-\infty}^{\infty} (x - \mu_X)^2 f_X(x)\,\mathrm{d}x,$$

which is always positive. The variance is often denoted by σ_X^2. In practical situations, we will often consider the *standard deviation* σ_X, which is just the square root of the variance, because it scales similar to the expected value.

3.2 Transformation of random variables

When we have some random variable X, we can obtain a new random variable Y by applying some real-valued function T to X, and taking $Y = T(X)$. But what is now the probability density function of Y in relation to X? This is unfold by following theorem.

Theorem 1. *Let X be a random variable with cumulative distribution function F_X and probability density function f_X, and let T be a real-valued differentiable monotonic function. Then the following applies to random variable $Y = T(X)$.*
If T is increasing, then Y has cumulative distribution function $F_Y(y) = F_X(T^{-1}(y))$ and probability density function $f_Y(y) = f_X(T^{-1}(y))\frac{\mathrm{d}T^{-1}(y)}{\mathrm{d}y}$.
If T is decreasing, then Y has cumulative distribution function $F_Y(y) = 1 - F_X(T^{-1}(x))$ and probability density function $f_Y(y) = -f_X(T^{-1}(y))\frac{\mathrm{d}T^{-1}(y)}{\mathrm{d}y}$.

The probability density function of Y can in general briefly be formulated by

$$f_Y(y) = f_X(T^{-1}(y))\left|\frac{\mathrm{d}T^{-1}(y)}{\mathrm{d}y}\right|. \tag{9}$$

Theorem 1 can be proven rather easily. Without going into detail, we simply have $F_Y(y) = \mathrm{P}(Y \leq y) = \mathrm{P}(T(X) \leq y) = \mathrm{P}(X \leq T^{-1}(y)) = F_X(T^{-1}(y))$ if T is monotonically increasing. The probability density function can be derived by elaborating $f_Y(y) = \frac{\mathrm{d}F_Y(y)}{\mathrm{d}y}$.

3.3 Simulation of random variables

Random variables can be simulated by computers, using a pseudo random number generator. Such generators aim to simulate variable U, distributed according to the standard uniform distribution on $[0, 1]$. Current modern pseudo random number generators like the Mersenne Twister are good enough to pass stringent randomness tests. [11]
The random variable U has probability density function

$$f_U(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

and cumulative distribution function

$$F_U(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } x \geq 1. \end{cases}$$

So simulating U leads to numbers equally distributed on the interval $[0, 1]$. In order to generate a different distribution, we have to find a function T, such that $Y = T(U)$ is distributed according to the desired distribution. The Probability Integral Transform, which is presented below, describes which function T we should choose.

Theorem 2. *(Probability Integral Transform)*
Let X be a continuous random variable with cumulative distribution function F_X, and let U be a continuous random variable with the standard uniform distribution on $[0, 1]$. If F_X is invertible, then random variable $Y \equiv F_X^{-1}(U)$ has the same probability distribution as X.

Proof. For any continuous random variable X with cumulative distribution function F_X, define the random variable $Z \equiv F_X(X)$. Because F_X is invertible and so strictly increasing, for any $z \in [0, 1]$ we have,

$$F_Z(z) = P(Z \leq z) = P(F_X(X) \leq z) = P(X \leq F_X^{-1}(z)) = F_X(F_X^{-1}(z)) = z.$$

Hence, F_Z is the cumulative distribution function of a uniform $[0, 1]$ distributed variable. The relation $U = F_X(X)$ directly implies $X = F_X^{-1}(U) = Y$. □

If F_X is not strictly increasing, we can not use the inverse F_X^{-1}. However, this problem can be circumvented by using a generalized inverse distribution function, but we will not going into detail about these nuances.

So with $T = F_X^{-1}$ we can transform the standard uniform distribution to the distribution of X, and thus this offers a way to simulate it by computer.

3.4 Histograms

The histogram is a classical method to graphically represent the distribution of numerical data. Therefore it is suitable tool to determine the probability distribution of some dataset of observations. In this section we describe how to construct a histogram, and how manipulate it in order to make it easier to recognize a probability density function.

We can obtain a histogram from some dataset as follows. Let $\Omega \subset \mathbb{R}$ be the set of possible values in the dataset. We divide the range of the data into disjoint intervals, which are called *bins*. We denote these bins by B_k for $k \in \mathbb{Z}$, and so $\Omega = \bigcup_k B_k$. The length of an interval is called the *bin width*, which is denoted by $|B_k|$. It is not necessary that all bin widths are equal. Now if $B_k = [b_k^-, b_k^+)$ and $b_k^- < b_k^+$, we have $|B_k| = b_k^+ - b_k^-$. Now we count how many times in the sequence of observations a value in B_k occurs, let us call this $H(B_k)$. In this way, the graph of H is a common histogram.

Thereafter, we normalize the histogram, such that the area of the bins in the histogram sums up to 1, and every measurements represents an equal part of that area. This is obtained by

$$h(B_k) \equiv \frac{H(B_k)}{n|B_k|},$$

where h is the normalized histogram, and n is size of the dataset. In case that all bin widths are equal, this normalization does not affect the shape of the histogram, but only the scaling.

Now note that we can read h as a probability density function. We take $\Omega = \mathbb{R}$ and define

$$h(x) \equiv h(B_k) \quad \text{for } x \in B_k.$$

If b_k is the midpoint of bin B_k, and $(\ldots, b_{k-1}, b_k, b_{k+1}, \ldots)$ forms an increasing series, we can make $h(x)$ more smooth by

$$h(x) \equiv h(B_k) + \frac{x - b_k}{b_{k+1} - b_k}\big(h(B_{k+1}) - h(B_k)\big) \quad \text{for } x \in B_k.$$

Function h can be read as a probability density function, because $h(x) > 0$ for all $x \in \mathbb{R}$, and

$$\int_{\mathbb{R}} h(x)\,\mathrm{d}x = \sum_{k\in\mathbb{Z}} \big|B_k\big|h(B_k) = \sum_{k\in\mathbb{Z}} \frac{H(B_k)}{n} = 1.$$

In case that we cannot easily recognize a proper probability distribution in the shape of h, it could probably help to transform the histogram to some other domain space. This transform should be done carefully, because the bin widths will change. Let us assume that function $T : \mathbb{R} \to \mathbb{R}$

describes the transform of the domain space, such that the transformed bins are $\widehat{B}_k = T(B_k)$. We assume that T is a differentiable monotonic function. Now the most straight-forward way is to calculate the transformed bins $\widehat{B}_k$, and compute the normalized histogram $\widehat{h}$. So if $\widehat{H}$ is the transformed histogram, we have

$$\widehat{h}(\widehat{B}_k) = \frac{\widehat{H}(\widehat{B}_k)}{n|\widehat{B}_k|} = \frac{H(B_k)}{n|\widehat{B}_k|} = \frac{|B_k|}{|T(B_k)|}h(B_k) = \left|\frac{b_k^+ - b_k^-}{T(b_k^+) - T(b_k^-)}\right| h(B_k). \tag{10}$$

This histogram transform is in fact a transform of the bin edges, and the bin heights are altered in such a way that the bin areas stays the same.

A different approach is obtained by applying a probability density transform to h. This transform was described in Section 3.2. In this way we get by (9),

$$\widehat{h}(x) = \left|\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x}\right| h(T^{-1}(x)) = \left|\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x}\right| h(B_k) \qquad \text{for } x \in \widehat{B}_k. \tag{11}$$

Note that this definition of $\widehat{h}$ is in fact no normalized histogram any more, because $\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x}$ is in general not constant. However, the approaches in (10) and (11) will not differ much, for the factor $\left|\frac{b_k^+ - b_k^-}{T(b_k^+) - T(b_k^-)}\right|$ is actually a finite-differences approximation of $\left|\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x}\right|$. Here we assume that the bin widths $|B_k|$ are small enough compared to variations in $\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x}$.

3.5 Maximum likelihood estimation

In previous section, we described how a given probability distribution can be simulated by a computer. However, in practical situations we have often to deal with the inverse problem: given a dataset, we have to determine from which probability distribution it likely is derived.

Often, this problem can loosely be solved by looking at the shape of the histograms of the dataset. However, we would like to fit some probability distribution in some mathematical way to the dataset. This is provided by the method of maximum likelihood. We explain it here as presented in [5] and [8].

Suppose that random variable X has probability density function $f_\lambda(x)$, which depends on one or more parameters denoted by $\lambda \in \mathbb{R}^q$, where q is the number of parameters. Now given the observed value $X_0 = x_0$, the likelihood of λ as function of x_0 is defined to be

$$\mathrm{lik}(\lambda) \equiv f_\lambda(x_0).$$

Next, assume that we have n independent identically distributed random variables $X_0, \ldots, X_{n-1}$, all distributed according to the same probability density function f_λ but with unknown λ. Now given observed values $X_0 = x_0, \ldots, X_{n-1} = x_{n-1}$, the likelihood of parameter λ as a function of

$x_0, \ldots, x_{n-1}$ is

$$\text{lik}(\lambda) = \prod_{k=0}^{n-1} f_\lambda(x_k).$$

Now the *maximum likelihood estimate* $\lambda_{\max}$ of λ is that value of λ that maximizes the likelihood: it makes the observed data "most likely".

We have to notice that the concept of likelihood differs from probability or chance. So $\text{lik}(\lambda)$ should not be read as a kind of probability density function in λ.

Rather than maximizing the likelihood itself, it is usually computationally easier to maximize its natural logarithm, which is called the *loglikelihood*. This is an equivalent problem, because the natural logarithm is a strictly increasing function. Moreover, it turns the product into a summation, which is much more handsome, because its derivative becomes a simple summation as well. So the loglikelihood is given by

$$\mathcal{L}(\lambda) \equiv \log(\text{lik}(\lambda)) = \sum_{k=0}^{n-1} \log\big(f_\lambda(x_k)\big). \tag{12}$$

In this way, the maximum likelihood estimator can be found by solving the equation

$$\sum_{k=0}^{n-1} \frac{1}{f_\lambda(x_k)} \cdot \frac{\partial f_\lambda(x_k)}{\partial \lambda} = 0, \tag{13}$$

for $\lambda = \lambda_{\max}$. In this way we can estimate parameters in our mathematical model. Unfortunately, it is not alway guaranteed whether (13) is uniquely solvable, so we should be careful.

It can be shown that the maximum likelihood estimator does not alway on average approximate the true value of λ. The maximum likelihood estimator may have a *bias*, which is the difference between the true value of λ and the expected value of the estimator. In Section 3.7 we will see an example. However, maximum likelihood estimators have several desirable properties, as stated in [5]. The maximum likelihood estimator may be biased, but it is always *asymptotically* (that is, as the size of the dataset goes to infinity) *unbiased*.

Next, the maximum likelihood estimator is *invariant* under transformations. So if T is the maximum likelihood estimator of a parameter θ, and g is an invertible function of θ, then $g(T)$ is the maximum likelihood estimator for $g(\theta)$.

And finally, the maximum likelihood estimator has *asymptotic minimum variance*. This means that (under mild conditions) the variance of the maximum likelihood estimators attains asymptotically the lowest possible value. So the maximum likelihood estimator is very useful among all unbiased estimators, it approximates the true values the fastest.

3.6 The normal distribution

The most famous probability distribution is the normal distribution. It is defined for parameters μ and σ by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{x-\mu}{\sigma}\Big)^2\Big).$$

The parameters μ and σ are chosen such that μ is the expected value, and σ the standard deviation. With $\mu = 0$ and $\sigma = 1$, we have the standard normal distribution.

The normal distribution is so widely applicable, because sums (and averages) of random variables tend to be approximately normal distributed. This is shown by the central limit theorem, which is stated below. A proof can be found in [8].

Theorem 3. (Central limit theorem) *Let $X_0, X_1, \ldots$ be any sequence of independent identically distributed random variables with finite positive variance. Let μ be the expected value and σ^2 the variance of each of the X_i. For $n \geq 1$, let Z_n be defined by*

$$Z_n = \sqrt{n}\frac{\bar{X}_n - \mu}{\sigma}, \tag{14}$$

where $\bar{X}_n \equiv \frac{1}{n}\sum_{k=0}^{n-1} X_k$. Then for any number $a \in \mathbb{R}$,

$$\lim_{n\to\infty} F_{Z_n}(a) = \Phi(a),$$

where Φ is the cumulative distribution function of the standard normal distribution.

For relatively small n, the distribution of Z_n in (14) can be already rather close to the standard normal distribution, as is shown in Figure 1.

An important property of the normal distribution is that its maximum likelihood estimator of μ appears to be the same as the least squares solution of μ. We can see this as follows. Assume that we have n observed values $x_0, \ldots, x_{n-1}$ which are believed to descend all from the same unknown normal distribution. In order to find out the most likely value of parameter μ for this distribution, we have to maximize the likelihood

$$\text{lik}(\mu) = \prod_{k=0}^{n-1} \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\Big(\frac{x_k-\mu}{\sigma}\Big)^2\Big).$$

We can omit $\frac{1}{\sigma\sqrt{2\pi}}$, because it does not affect μ. So we take

$$\text{lik}(\mu) = \prod_{k=0}^{n-1} \exp\Big(-\frac{1}{2}\Big(\frac{x_k-\mu}{\sigma}\Big)^2\Big).$$

This is equivalent to maximizing the loglikelihood

$$\mathcal{L}(\mu) = \sum_{k=0}^{n-1} -\frac{1}{2}\Big(\frac{x_k-\mu}{\sigma}\Big)^2 = -\frac{1}{2\sigma^2}\sum_{k=0}^{n-1}(x_k-\mu)^2.$$

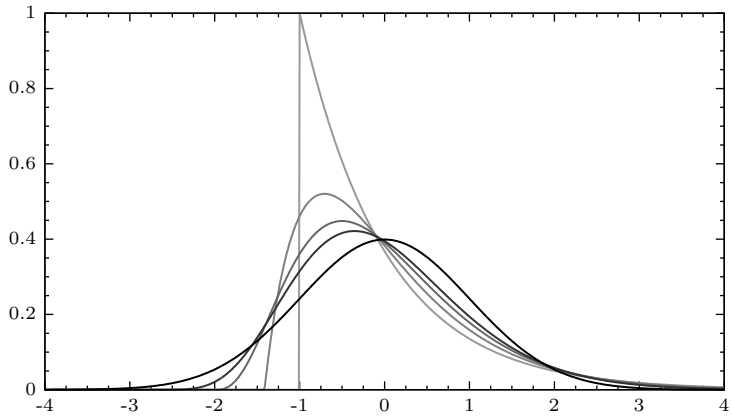


Figure 1: Illustration of the Central Limit Theorem for the exponential distribution. Displayed are the probability distributions for standardized averages of 1, 2, 4, and 8 variables (gray scales), and the standard normal distribution (black).

And this is equivalent to minimizing the sum of squares

$$\sum_{k=0}^{n-1}(x_k - \mu)^2.$$

So the maximum likelihood estimator of μ is identical to the least squares solution of μ.

3.7 The Exponential and Gamma distributions

In Chapter 5 we will see that the exponential distribution models pretty good the received signal strength. Therefore, we discuss here the properties of this probability distribution. It also allows us to demonstrate some properties of the maximum likelihood estimate.

The Exponential distribution

For some $\lambda > 0$, the exponential distribution has probability density function

$$f_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{15}$$

and cumulative density function

$$F_\lambda(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{16}$$

Given some exponential distribution, the expected value is

$$\mu = \int_0^\infty x \cdot \lambda e^{-\lambda x}\,\mathrm{d}x = \left[-\left(x + \lambda^{-1}\right)e^{-\lambda x}\right]_{x=0}^{\infty} = \lambda^{-1}. \tag{17}$$

So λ^{-1} is the average value of the exponential distribution with parameter λ. The variance is

$$\sigma^2 = \int_0^\infty (x - \lambda^{-1})^2 \cdot \lambda e^{-\lambda x}\,\mathrm{d}x = \left[-\left(x + \lambda^{-2}\right)e^{-\lambda x}\right]_{x=0}^{\infty} = \lambda^{-2}, \tag{18}$$

and so the standard deviation σ is λ^{-1}.

The Gamma distribution

The exponential distribution is a special case of the *Gamma distribution* $\mathrm{Gamma}(\alpha, \lambda)$. For some "shape parameter" α and "scale parameter" λ, it has probability density function

$$f_{\alpha,\lambda}(x) = \begin{cases} \dfrac{1}{\Gamma(\alpha)}\lambda^\alpha x^{\alpha-1}e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases} \tag{19}$$

where Γ is the gamma-function, which is defined by

$$\Gamma(\alpha) \equiv \int_0^\infty t^{\alpha-1}e^{-t}\,\mathrm{d}t.$$

The gamma-function satisfies for $\alpha > 0$ the relation $\Gamma(\alpha + 1) = \alpha\Gamma(\alpha)$ and for integer $\alpha \geq 1$ we have $\Gamma(\alpha) = (\alpha - 1)!$.

Some important properties of the Gamma distribution are as follows. The $\mathrm{Gamma}(\alpha, \lambda)$ distribution has expected value $\alpha\lambda^{-1}$ and variance $\alpha\lambda^{-2}$. For $\alpha = 1$, the Gamma distribution turns into the exponential distribution. It can also be shown that for n independent variables, identically distributed according to an exponential distribution with parameter λ, their sum has $\mathrm{Gamma}(n, \lambda)$ distribution. More precise, if X has a $\mathrm{Gamma}(\alpha_X, \lambda)$ and Y has a $\mathrm{Gamma}(\alpha_Y, \lambda)$ probability distribution, then $Z = X + Y$ has a $\mathrm{Gamma}(\alpha_X + \alpha_Y, \lambda)$ distribution.

Maximum likelihood estimation for the exponential distribution

Now let us turn to maximum likelihood estimation for the exponential distribution. Assume that we have observed values $x_0, \ldots, x_{n-1}$, which are believed to come from the same exponential distribution with unknown parameter λ. When we use the method of maximum likelihood to estimate parameter λ, we have to maximize for $\lambda > 0$

$$\mathcal{L}(\lambda) = \sum_{k=0}^{n-1} \log\left(\lambda e^{-\lambda x_k}\right) = n\log(\lambda) - \lambda\sum_{k=0}^{n-1} x_k.$$

Differentiation gives the equation

$$\frac{d\mathcal{L}(\lambda)}{d\lambda} = \frac{n}{\lambda} - \sum_{k=0}^{n-1} x_k = 0.$$

This gives the maximum likelihood estimate

$$\lambda_{\text{mle}} = \frac{1}{\frac{1}{n}\sum_{k=0}^{n-1} x_k}, \tag{20}$$

so the most likely value of λ is the inverse of the mean of the sample set. However, this estimator of λ is biased, which can be seen as follows.

The sum of the sample set $\sum_{k=0}^{n-1} x_k$ has probability distribution Gamma(n, λ), and the mean has probability distribution Gamma$(n, n\lambda)$. This is probability density function

$$f(x) = \frac{1}{\Gamma(n)}(n\lambda)^n x^{n-1} e^{-n\lambda x} \quad \text{for } x \geq 0.$$

Using Theorem 1, we can now compute the probability distribution function of the inverse of the mean of the sample set. We take $T(x) = \frac{1}{x} = T^{-1}(x)$, and so $\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x} = -\frac{1}{x^2}$. It is clear that $T(x)$ for $x > 0$ is a monotonically decreasing function. Hence the inverse mean has probability density distribution

$$f(x) = \frac{1}{\Gamma(n)}(n\lambda)^n x^{-n-1} e^{-n\lambda/x} \quad \text{for } x \geq 0.$$

With some help of symbolic mathematical computation program Mathematica, we find that this probability density has expected value $\mathrm{E}(\lambda_{\text{mle}}) = \frac{n}{n-1}\lambda$. So especially for small n, the estimator λ_{mle} of λ in (20) is biased. We should correct this, and estimate λ as

$$\lambda_{\text{mle,c}} = \frac{1}{\frac{1}{n-1}\sum_{k=0}^{n-1} x_k}. \tag{21}$$

Note that the unbiased estimator $\lambda_{\text{mle,c}}$ only exists if $n > 1$.

So we found that the inverse sample mean is a biased estimator of λ. However, the ordinary sample mean $\frac{1}{n}\sum_{k=0}^{n-1} x_k$ is equal to the expected value $\mu = \lambda^{-1}$, so this is an *unbiased* estimator for λ^{-1}.

Comparison with other estimators for the exponential distribution

To show the optimality of the maximum likelihood estimation, we discuss now the maximum likelihood estimator and two different estimators of λ^{-1}. With observed values $x_0, \ldots, x_{n-1}$, we can find the maximum likelihood estimator by maximizing

$$\mathcal{L}(\lambda^{-1}) = \sum_{k=0}^{n-1} \log\Big(\frac{1}{\lambda^{-1}} \exp\big(-\frac{1}{\lambda^{-1}} x_k\big)\Big) = -n\log(\lambda^{-1}) - \frac{1}{\lambda^{-1}}\sum_{k=0}^{n-1} x_k.$$

Differentiation gives

$$\frac{\mathrm{d}\mathcal{L}(\lambda^{-1})}{\mathrm{d}\lambda^{-1}} = \frac{-n}{\lambda^{-1}} + \frac{1}{(\lambda^{-1})^2}\sum_{k=0}^{n-1} x_k = 0,$$

and so

$$\lambda_{\text{mle}}^{-1} = \frac{1}{n}\sum_{k=0}^{n-1} x_k.$$

We see that the maximum likelihood estimator of λ^{-1} is simply the mean of the sample set. As stated before in this section, the mean of the sample set has probability distribution Gamma$(n, n\lambda)$. So $\lambda_{\text{mle}}^{-1}$ has expected value $\mathrm{E}(\lambda_{\text{mle}}^{-1}) = n(n\lambda)^{-1} = \lambda^{-1}$, and variance $\mathrm{Var}(\lambda_{\text{mle}}^{-1}) = n(n\lambda)^{-2} = n^{-1}\lambda^{-2}$. Because $\mathrm{E}(\lambda_{\text{mle}}^{-1}) = \lambda^{-1}$, the estimator $\lambda_{\text{mle}}^{-1}$ is unbiased.

Now define $m \equiv \min_k x_k$ to be the minimum observed value and $M \equiv \max_k x_k$ to be the maximum observed value in the sample set.

Concerning m, we see that it has cumulative density function

$$F_m(x) = \mathrm{P}(m \le x) = 1 - \mathrm{P}(m \ge x) = 1 - \prod_{k=0}^{n-1} \mathrm{P}(X_k \ge x) = 1 - \prod_{k=0}^{n-1} \big(1 - \mathrm{P}(X_k \le x)\big).$$

In this way we get

$$F_m(x) = 1 - \left(e^{-\lambda x}\right)^n = 1 - e^{-n\lambda x} \qquad (x > 0).$$

This is exactly the cumulative density function of an exponential distribution with parameter $n\lambda$, and so m has expected value $\mathrm{E}(m) = (n\lambda)^{-1}$ and variance $\mathrm{Var}(m) = (n\lambda)^{-2}$. Then we have $\mathrm{E}(nm) = \lambda^{-1}$, and so nm is a unbiased estimator of λ^{-1}. The variance of this estimator is $\mathrm{Var}(nm) = n^2(n\lambda)^{-2} = \lambda^{-2}$. It is a bit strange that the variance of this estimator does not depend on the sample size.

Concerning M, this has cumulative density function

$$F_M(x) = \mathrm{P}(M \le x) = \prod_{k=0}^{n-1} \mathrm{P}(X_k \le x) = \left(1 - e^{-\lambda x}\right)^n \qquad (x > 0).$$

So the probability density function of M is

$$f_M(x) = \frac{\mathrm{d}}{\mathrm{d}x} F_M(x) = n\lambda e^{-\lambda x}\left(1 - e^{-\lambda x}\right)^{n-1} \qquad (x > 0).$$

Now we can compute the expected value of M by

$$\mathrm{E}(M) = \int_0^\infty x\, f_M(x)\, \mathrm{d}x = \int_0^\infty n\lambda x e^{-\lambda x}\left(1 - e^{-\lambda x}\right)^{n-1} \mathrm{d}x = \frac{a_n}{\lambda},$$

where $a_n \equiv \sum_{k=1}^n k^{-1}$ is called the nth harmonic number. So we find that M/a_n is an unbiased estimator of λ^{-1}. It is a bit difficult to compute the variance $\mathrm{Var}(M/a_n)$, but with the help of Mathematica (again) we find with $b_n \equiv \sum_{k=1}^n k^{-2}$

$$\mathrm{Var}(M/a_n) = \int_0^\infty (x - \lambda^{-1})^2\, a_n f_M(a_n x)\, \mathrm{d}x = a_n^{-2} b_n \lambda^{-2}.$$

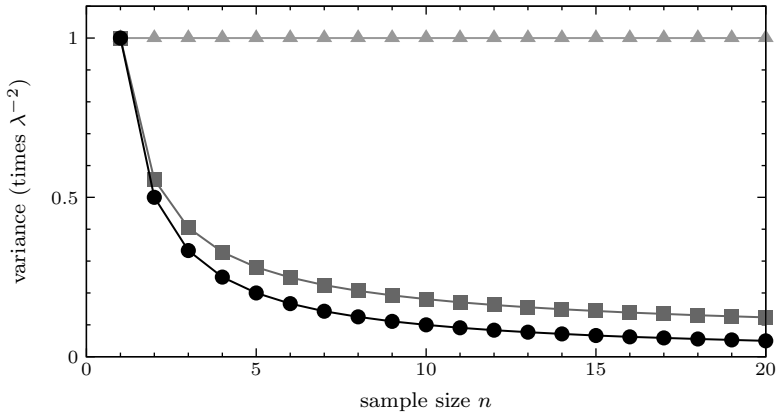


Figure 2: The variances of three estimators as function of sample size n. $\mathrm{Var}(nm)$ is shown by light gray triangles, $\mathrm{Var}(M/a_n)$ is shown by dark gray squares, and $\mathrm{Var}(\lambda_{\mathrm{mle}}^{-1})$ is shown by black circles. Better estimators have smaller variances, and so in this example estimator $\lambda_{\mathrm{mle}}^{-1}$ from the maximum likelihood estimate turns out to be the best estimator.

Comparing the variances of the three estimators, we find

$$\mathrm{Var}(\lambda_{\mathrm{mle}}^{-1}) = \frac{1}{n\lambda^2} \leq \mathrm{Var}(M/a_n) = \frac{b_n}{a_n^2\lambda^2} \leq \mathrm{Var}(nm) = \frac{1}{\lambda^2},$$

as it is illustrated in Figure 2. It is clear that the estimator nm, based on the minimum observed value, is rather useless. Furthermore, for these examples the maximum likelihood estimation gives clearly the best estimator. However, the estimator M/a_n, which is based on the maximum observed variable, is not much worse, especially for small sample sizes.

4 Numerical optimization techniques

In Section 3.5 we met the method of maximum likelihood, which we will use to derive a localization procedure in Chapter 6. In this chapter, we will discuss how to solve optimization problems efficiently, because maximizing likelihood is in fact an optimization problem.

In literature, optimization problems are almost always treated as minimization problems. Any maximization problem can be treated as minimization problem by multiplying the objective by -1. In this chapter we will stay with this convention. So a general optimization problem can be written mathematically as

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{22}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective to be minimized, n is the dimension of the problem, and $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{R}^n$ are the variables. Here we have an *unconstrained* optimization problem. Constrained optimization problems, where the solution should satisfy some equality or inequality conditions, are very common as well. There are also discrete optimization problems, where the solution should be integer-valued. Such problems need to be treated separately, and are not relevant in this research.

In this chapter, we explore first the conditions which a solution of the optimization problem should meet, Section 4.1. Thereafter we will describe the basic structure of an optimization method in Section 4.2.

Section 4.3 handles about local function approximations, because these approximations are a basic ingredient for many optimization methods. In Section 4.4 we show how these function approximations can be applied in optimization methods. In Section 4.6 we will make some final comments.

The information in this chapter is mainly token from [12].

4.1 Optimization solution conditions

Solutions of optimization problem (22) should satisfy a few conditions in order to be a solution. These conditions are very important, because we can use them to find and identify solutions. In this section we will derive these conditions.

While solving minimization problem (22), we would in general be happiest if we find a *global* minimizer: a point $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$. However, it can be very difficult to find a global minimizer. This is because from efficiency considerations we usually want the algorithm to do not visit many points, and so we can never be sure whether f takes a sharp dip in some region that has not been sampled by the algorithm. This is why fast optimization algorithms usually only seek for *local* solutions, and do not always find a global solution. A local minimum can be defined as follows.

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ has a* local minimum *at $\mathbf{x}^* \in \mathbb{R}^n$, if there exists a $\delta > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \{\mathbf{y} \mid \|\mathbf{x}^* - \mathbf{y}\| < \delta\}$.*

So fast optimization algorithms should use some a priori knowledge about the problem in order to find a global solution. Usually, some proper initial guess and an adequate optimization algorithm will be sufficient, and we do not need to care about the distinction between local and global solutions.

Now we will derive some conditions which are satisfied by local solutions. This derivation starts at Taylor's theorem, which is the central theorem in the study of approximations of smooth functions. We write down its multivariate version, the proof can be found in many textbooks on calculus.

Theorem 4. (Taylor's theorem)
If function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, then

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^T \nabla f(\mathbf{x} + s\mathbf{h}) \qquad \text{for some } s \in [0, 1]. \tag{23}$$

Furthermore, if f is twice continuously differentiable, then

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^T \nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T [\nabla^2 f(\mathbf{x} + s\mathbf{h})]\mathbf{h} \qquad \text{for some } s \in [0, 1]. \tag{24}$$

From Taylor's theorem follow two important conditions which a (local) minimizer should meet. We write them down in following theorem.

Theorem 5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuous differentiable. If f has a local minimum at $\mathbf{x}^* \in \mathbb{R}^n$, then the following hold: (i) $\nabla f(\mathbf{x}^*) = \mathbf{0}$, and (ii) $\nabla^2 f(\mathbf{x}^*)$ is positive semidefinite.*

Proof. Part (i). We have $f(\mathbf{x}^*) \leq f(\mathbf{x}^* + s\mathbf{d})$ for all unit vectors $\mathbf{d} \in \mathbb{R}^n$ ($\|\mathbf{d}\|_2 = 1$) and $s > 0$ small enough, because f attains a local minimum at $\mathbf{x}^*$. Then by (23),

$$0 \leq \frac{f(\mathbf{x}^* + s\mathbf{d}) - f(\mathbf{x}^*)}{s} = \mathbf{d}^T \nabla f(\mathbf{x}^* + st\mathbf{d})$$

for some $t \in [0, 1]$. Now by taking limit $s \to 0$, we get

$$\mathbf{d}^T \nabla f(\mathbf{x}^*) \geq 0.$$

Because unit vector $\mathbf{d}$ is arbitrary, we can replace it by $-\mathbf{d}$, and so $\mathbf{d}^T \nabla f(\mathbf{x}^*) = 0$. Again applying the fact that unit vector $\mathbf{d}$ is arbitrary, gives $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

Part (ii). For all unit vectors $\mathbf{d} \in \mathbb{R}^n$ and $s > 0$ small enough, we have by Taylor's theorem for some $t \in [0, 1]$

$$f(\mathbf{x}^* + s\mathbf{d}) - f(\mathbf{x}^*) = s\mathbf{d}^T \nabla f(\mathbf{x}^*) + \frac{1}{2}s^2 \mathbf{d}^T [\nabla^2 f(\mathbf{x}^* + st\mathbf{d})]\mathbf{d} \geq 0.$$

Because $\nabla f(\mathbf{x}^*) = \mathbf{0}$, we see

$$\mathbf{d}^T[\nabla^2 f(\mathbf{x}^* + st\mathbf{d})]\mathbf{d} \geq 0.$$

Unit vector $\mathbf{d}$ is arbitrary, and s can be arbitrary small, and $\nabla^2 f(\mathbf{x})$ is continuous, so we get

$$\mathbf{d}^T[\nabla^2 f(\mathbf{x}^*)]\mathbf{d} \geq 0,$$

which shows that $\nabla^2 f(\mathbf{x}^*)$ is positive semidefinite. □

Note that Theorem 5 formulates necessary conditions for a minimum. These conditions are not sufficient for a minimum, as can be shown by the example $f(x) = x^3$. This function does not have a minimum at $x = 0$, but meets the necessary conditions. To exclude such cases, there are more subtle conditions needed, as $\nabla^2 f$ should be positive semidefinite everywhere in an open neighborhood of $\mathbf{x}^*$. However, it is difficult to check this condition in the numerical practice. Therefore we will simply assume that encountering such points is very rare.

Moreover, it is generally assumed that it is very rare that some minimization algorithm encounters a local maximum, or a saddle point. Therefore, during minimization it is usually sufficient to check only for condition (i) in Theorem 5 during minimization. This has also the benefit that there is no need to compute the second derivative, and check whether it is positive definite, which can be computationally expensive.

4.2 Optimization methods in general

Methods for solving nonlinear optimization problems are in their essence *iterative methods*. They need an initial guess $\mathbf{x}_0$ and generate an infinite sequence $\{\mathbf{x}_t\}$ of approximate solutions. The next *iterate* $\mathbf{x}_{t+1}$ is formed according to certain rules, and based on local information of the problem collected along the previous iterate(s). In this way, the sequence $\{\mathbf{x}_t\}$ is hopefully an improving sequence of approximate solutions.

Iterative methods can be distinguished by the rule they use to compute the next iterate. Because there are many methods, they are categorized by the kind of derivatives they call. Some methods do not call derivatives, but only function values. These methods are useful in case that the analytic form of the function is unknown. However, methods which call function values and derivatives are in general computationally more efficient. Methods which also call second derivatives have high convergence rates near a minimum, but computing the Hessian could be expensive. In Section 4.4 we will discuss some examples.

So iterative methods operate as follows. In their current iterate, they determine a descent direction and an appropriate stepsize to a new iterate, see Algorithm 1. This process is repeated until $\|\nabla f(\mathbf{x}_k)\|_2$ (or an approximation of this quantity) is small enough, which corresponds to the first part of Theorem 5.

Algorithm 1 A general iterative minimization method

Select initial guess $\mathbf{x}_0$

$\mathbf{x} \leftarrow \mathbf{x}_0$

while $\|\nabla f(\mathbf{x})\|_2 > \epsilon$ **do**

Determine at $\mathbf{x}$ descent direction $\mathbf{d}$ and stepsize $t > 0$

Update $\mathbf{x} \leftarrow \mathbf{x} + t\mathbf{d}$

end while

The determination of the descent direction and the stepsize are the two essential ingredients of a iterative method. Now there are basically two strategies. Algorithms from the *line search* strategy determine a descent direction first, and then focus on the determination of an appropriate stepsize. The stepsize is found by loosely solving the one dimensional subproblem

$$\min_{t>0} f(\mathbf{x} + t\mathbf{d}). \tag{25}$$

This subproblem is a so-called *inexact* line search, and can be solved loosely by a simple iterative method. In Section 4.5 we will explain how to do such a line search.

Algorithms from the *trust region* strategy are in some sense dual to line search methods. They first determine a stepsize t, and then try to find an appropriate descent direction $\mathbf{d}$, which is found by loosely solving

$$\min_{\|\mathbf{d}\| \leq 1} f(\mathbf{x} + t\mathbf{d}) \tag{26}$$

for $\mathbf{d}$. If this does not produce sufficient decrease in the objective f, the trust region was too large, and a smaller stepsize t is tried. If good steps are obtained, the size of the trust region can be increased to allow still more ambitious steps. We will not discuss specific trust region methods, but focus on the line search strategy.

Characteristic for methods from the line search strategy is the descent direction which is chosen. We will discuss some important descent directions in Section 4.4. Because these descent directions are based on function approximations, we will discuss some important function approximations in next section.

4.3 Function approximations

As already explained in previous section, function approximations are necessary for good optimization methods. Local function approximations tell some information about the behavior of the original function, and can in this way be used to solve the minimization problem. In this section we consider function approximations by interpolation and by Taylor series approximation.

We will only consider linear and quadratic approximations, because these are simple. Quadratic approximations are useful, because they give generally accurate function approximations near their optima. Linear approximations are useful while finding zeros of some function, especially $\nabla f = \mathbf{0}$. Note that linear approximations on ∇f are in fact quadratic approximations on f.

Further, we will pay special attention to the case of one dimensional functions which arises during line search: assume that we want approximations of f at point $\mathbf{x}$ in one specific direction, say $\mathbf{d} \in \mathbb{R}^n$. Then define

$$\phi(t) \equiv f(\mathbf{x} + t\mathbf{d}),$$

where ϕ is a function of only one variable. Note that we now have $\phi'(t) = \mathbf{d}^T \nabla f(\mathbf{x} + t\mathbf{d})$, and $\phi''(t) = \mathbf{d}^T [\nabla^2 f(\mathbf{x} + t\mathbf{d})]\mathbf{d}$.

For references about interpolation in several variables, see [13] and citations given there.

Linear interpolation

Interpolation is a very basic technique for approximating function value, and linear interpolation is the simplest form of interpolation. Assume that we want to interpolate on function $f : \mathbb{R}^n \to \mathbb{R}$. Then we have to take some starting point $\mathbf{x}_0 \in \mathbb{R}^n$ and n linear independent points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^n$. Now the linear interpolant p is a function which is linear along all directions, and equals f at the points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$. This function p can be determined by solving

$$\det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \mathbf{x} & \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ p(\mathbf{x}) & f(\mathbf{x}_0) & f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_n) \end{pmatrix} = 0. \tag{27}$$

In this way, we force that $p(\mathbf{x}_k) = f(\mathbf{x}_k)$ for $k = 0, \ldots, n$, and $p(\mathbf{x})$ is linear. This is enough to determine $p(\mathbf{x})$. We solve this equation for $p(\mathbf{x})$ by defining

$$A \equiv \begin{pmatrix} 1 & \cdots & 1 \\ \mathbf{x}_0 & \cdots & \mathbf{x}_n \end{pmatrix},$$

and

$$A_k(\mathbf{x}) \equiv \begin{pmatrix} 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ \mathbf{x}_0 & \cdots & \mathbf{x}_{k-1} & \mathbf{x} & \mathbf{x}_{k+1} & \cdots & \mathbf{x}_n \end{pmatrix},$$

where the kth column is replaced by $\mathbf{x}$. Then an explicit formula for $p(\mathbf{x})$ is given by

$$p(\mathbf{x}) = \frac{1}{\det(A)} \sum_{k=0}^{n} f(\mathbf{x}_k) \det(A_k(\mathbf{x})).$$

Here we used the Laplace expansion along the last row to compute the determinant in (27). By changing the column order in the definition of $A_k(\mathbf{x})$, we avoid the need of multiplying alternating

by the co-factors $+1$ and -1. We can find the gradient of p by taking $\mathbf{x}_j = (x_{j,0}, \ldots, x_{j,n-1})^T$, and defining for $k = 0, \ldots, n-1$,

$$B_k \equiv \begin{pmatrix} 1 & \cdots & 1 \\ x_{0,0} & \cdots & x_{n,0} \\ \vdots & \ddots & \vdots \\ x_{0,k-1} & \cdots & x_{n,k-1} \\ f(\mathbf{x}_0) & \cdots & f(\mathbf{x}_n) \\ x_{0,k+1} & \cdots & x_{n,k+1} \\ \vdots & \ddots & \vdots \\ x_{0,n-1} & \cdots & x_{n,n-1} \end{pmatrix},$$

So B_k is equal to A, but the row containing $x_{0,k} \ \cdots \ x_{n,k}$ has been replaced by $f(\mathbf{x}_0) \ \cdots \ f(\mathbf{x}_n)$. Then the gradient $\nabla f(\mathbf{x})$ can be approximated by

$$\nabla p(\mathbf{x}) = \frac{1}{\det(A)} \begin{pmatrix} \det(B_0) \\ \vdots \\ \det(B_{n-1}) \end{pmatrix}.$$

Now we used Laplace expansion along the first column in (27) to get a alternate expression for $p(\mathbf{x})$, namely $p(\mathbf{x}) = \frac{1}{\det(A)} \sum_{k=0}^{n-1} x_k \det(B_k)$. Here, $\mathbf{x} = (x_0, \ldots, x_{n-1})^T$. This can easily be differentiated.

In the one dimensional situation, we need ϕ at two points, say t_0 and t_1. Then we have $A = \left(\begin{smallmatrix} 1 & 1 \\ t_0 & t_1 \end{smallmatrix}\right)$, $A_0 = \left(\begin{smallmatrix} 1 & 1 \\ t & t_1 \end{smallmatrix}\right)$, and $A_1 = \left(\begin{smallmatrix} 1 & 1 \\ t_0 & t \end{smallmatrix}\right)$. So $\det(A) = t_1 - t_0$, $\det(A) = t_1 - t$, and $\det(A) = t - t_0$. Hence,

$$p(t) \approx \frac{t_1 - t}{t_1 - t_0}\phi(t_0) + \frac{t - t_0}{t_1 - t_0}\phi(t_1).$$

To approximate the derivative $\phi'(t)$, we find $B_0 = \left(\begin{smallmatrix} 1 & 1 \\ f(t_0) & f(t_1) \end{smallmatrix}\right)$, and so $\det(B_0) = f(t_1) - f(t_0)$ and

$$p'(t) \approx \frac{\phi(t_1) - \phi(t_0)}{t_1 - t_0}.$$

Quadratic interpolation

Quadratic interpolation is more attractive than linear interpolation, because they can approximate function near a minimum very well. However, the formulas get rather complicated as we will show below. Quadratic interpolation can be done by solving

$$\det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \mathbf{x} & \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{m-1} \\ \mathbf{x^2} & \mathbf{x}_0^2 & \mathbf{x}_1^2 & \cdots & \mathbf{x}_{m-1}^2 \\ p(\mathbf{x}) & f(\mathbf{x}_0) & f(\mathbf{x}_1) & \cdots & f(\mathbf{x}_{m-1}) \end{pmatrix} = 0,$$

where $\mathbf{x^2}$ is a vector containing all quadratic terms. So for $n = 2$, we have $\mathbf{x^2} = (x_0^2, x_0x_1, x_1^2)^T$. For unique quadratic interpolation, there are $m \equiv \binom{n+2}{2} = \frac{1}{2}(n+1)(n+2)$ interpolation points needed. In order to solve this equation, define now

$$A \equiv \begin{pmatrix} 1 & \cdots & 1 \\ \mathbf{x}_0 & \cdots & \mathbf{x}_{m-1} \\ \mathbf{x_0^2} & \cdots & \mathbf{x_{m-1}^2} \end{pmatrix}, \qquad \text{and} \qquad A_k(\mathbf{x}) \equiv \begin{pmatrix} 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \\ \mathbf{x}_0 & \cdots & \mathbf{x}_{k-1} & \mathbf{x} & \mathbf{x}_{k+1} & \cdots & \mathbf{x}_{m-1} \\ \mathbf{x_0^2} & \cdots & \mathbf{x_{k-1}^2} & \mathbf{x^2} & \mathbf{x_{k+1}^2} & \cdots & \mathbf{x_{m-1}^2} \end{pmatrix},$$

where the kth column is replaced by $\mathbf{x}$. Then, similar to the linear case, $p(\mathbf{x})$ is given explicitly by

$$p(\mathbf{x}) = \frac{1}{\det(A)} \sum_{k=0}^{m-1} f(\mathbf{x}_k) \det(A_k(\mathbf{x})).$$

This quadratic approximation can be used for estimating the first and second derivatives. Even though the approximation formula looks elegant, from a computational point of view it is very unattractive. The formula contains many expensive determinants, such that this function approximation method becomes unsuitable for $n \geq 2$. For $n = 1$, we can easily write down an explicit formulas. So if ϕ is known at 3 points, say t_0, t_1, and t_2, we can do quadratic interpolation. We find after computation of the determinants,

$$\phi(t) \approx \frac{t-t_1}{t_0-t_1}\frac{t-t_2}{t_0-t_2}\phi(t_0) + \frac{t_0-t}{t_0-t_1}\frac{t-t_2}{t_1-t_2}\phi(t_1) + \frac{t_0-t}{t_0-t_2}\frac{t_1-t}{t_1-t_2}\phi(t_2).$$

The approximation of the derivative is

$$\phi'(t) \approx \frac{2t-t_1-t_2}{(t_0-t_1)(t_0-t_2)}\phi(t_0) + \frac{t_0-2t+t_2}{(t_0-t_1)(t_1-t_2)}\phi(t_1) + \frac{-t_0-t_1+2t}{(t_0-t_2)(t_1-t_2)}\phi(t_2),$$

and so the approximate solution of $\phi'(t) = 0$ is

$$t = \frac{1}{2}\frac{(t_1^2-t_2^2)\phi(t_0) + (-t_0^2+t_2^2)\phi(t_1) + (t_0^2-t_1^2)\phi(t_2)}{(t_1-t_2)\phi(t_0) + (-t_0+t_2)\phi(t_1) + (t_0-t_1)\phi(t_2)}$$

We can approximate the second derivative as well,

$$\phi''(t) \approx \frac{2\phi(t_0)}{(t_0-t_1)(t_0-t_2)} + \frac{-2\phi(t_1)}{(t_0-t_1)(t_1-t_2)} + \frac{2\phi(t_2)}{(t_0-t_2)(t_1-t_2)}.$$

These interpolation approximations use function values only. It is also possible to derive approximations using derivatives. We will consider them in the next section. Such methods are in general more powerful.

Taylor series approximation

Theorem 4 (Taylor's theorem) gives rise to the following first and second order approximations. With (23) we get a first order approximation,

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\mathbf{x}-\mathbf{x}_0)^T \nabla f(\mathbf{x}_0).$$

With (24) we get a second order approximation,

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^T \nabla f(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T [\nabla^2 f(\mathbf{x}_0)](\mathbf{x} - \mathbf{x}_0).$$

We can also approximate the gradient,

$$\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}_0) + [\nabla^2 f(\mathbf{x}_0)](\mathbf{x} - \mathbf{x}_0). \tag{28}$$

In the one-dimensional case, these Taylor series approximations become as follows. The first order approximation is

$$\phi(t) \approx \phi(t_0) + (t - t_0)\phi'(t_0).$$

The second order Taylor approximation is

$$\phi(t) \approx \phi(t_0) + (t - t_0)\phi'(t_0) + \frac{1}{2}(t - t_0)^2\phi''(t_0).$$

The approximation of the derivative is

$$\phi'(t) \approx \phi'(t_0) + (t - t_0)\phi''(t_0). \tag{29}$$

Using (28) for solving $\nabla f(\mathbf{x}) = \mathbf{0}$ gives Newton's method

$$\mathbf{x} = \mathbf{x}_0 - [\nabla^2 f(\mathbf{x}_0)]^{-1} \nabla f(\mathbf{x}_0) \tag{30}$$

In one dimension, we use (29) and get by $\phi'(t) = 0$

$$t = t_0 - \frac{\phi'(t_0)}{\phi''(t_0)}. \tag{31}$$

We will come back to Newton's method in Section 4.4.2.

4.4 Descent directions

In this section, we will discuss some important descent directions, and explain the underlying ideas.

4.4.1 Steepest descent

A direction vector $\mathbf{d}$ at some point $\mathbf{x}$ is a descent direction of function f if for all sufficiently small $s > 0$ we have $f(\mathbf{x} + s\mathbf{d}) < f(\mathbf{x})$. This is equivalent to

$$0 > \lim_{s \to 0} \frac{f(\mathbf{x} + s\mathbf{d}) - f(\mathbf{x})}{s} = \mathbf{d}^T \nabla f(\mathbf{x}).$$

So if $\mathbf{d}^T \nabla f(\mathbf{x}) < 0$, then $\mathbf{d}$ is a descent direction. That means that the angle θ between $\mathbf{d}$ and $\nabla f(\mathbf{x})$ is in the domain $(\frac{1}{2}\pi, \frac{3}{2}\pi)$, which follows from the relation

$$\mathbf{d}^T \nabla f(\mathbf{x}) = \|\mathbf{d}\|_2 \|\nabla f(\mathbf{x})\|_2 \cos\theta < 0 \qquad \Leftrightarrow \qquad \cos\theta < 0.$$

The steepest descent direction is obtained at $\cos\theta = -1$, so

$$\mathbf{d}_{\text{sd}} = -\nabla f(\mathbf{x}).$$

The steepest descent direction is associated to a first order Taylor series approximation. Steepest descent algorithms do usually converge linearly, which is slow compared to some other methods.

4.4.2 Newton

When we use a quadratic approximation of f to find a minimum, we get by equations (28) and (30) the Newton-direction, which is different from the steepest descent direction,

$$\mathbf{d}_{\text{n}} = -[\nabla^2 f(\mathbf{x})]^{-1}\nabla f(\mathbf{x}).$$

Quadratic function approximations are very accurate near an optimum, and therefore iterative methods which use the Newton-direction converge usually very fast once the iterates are in the neighborhood of an optimum.

Unfortunately, it is not always guaranteed that $\mathbf{d}_{\text{n}}$ is a descent direction. We see that

$$\mathbf{d}_{\text{n}}^T\nabla f(\mathbf{x}) = -\nabla f(\mathbf{x})^T[\nabla^2 f(\mathbf{x})]^{-1}\nabla f(\mathbf{x}) < 0$$

can only be guaranteed if $\nabla^2 f(\mathbf{x})$ is positive semidefinite. Another drawback is that the Hessian $\nabla^2 f(\mathbf{x})$ needs to be computed, which can be error-prone and expensive. Furthermore, we need to solve the matrix-vector equation $\nabla^2 f(\mathbf{x})\mathbf{d}_{\text{n}} = -\nabla f(\mathbf{x})$.

At the other side, if $\mathbf{d}_{\text{n}}$ is a descent direction, there is a "natural" stepsize of 1 associated to it. Therefore it is often not necessary to perform a line search, and simply unit stepsize can be used. If it does not produce satisfactory reduction of the function value, the stepsize can still be adjusted.

4.4.3 Quasi-Newton

Quasi-Newton directions are an attractive alternative for the Newton direction, because they do not require computation of the Hessian solve a matrix-vector system, and still lead to similar convergence properties. Instead of using the true Hessian $\nabla^2 f(\mathbf{x}_k)$ in step k, they use an approximating matrix $\mathbf{B}_k$, which is updated after each step to take account of the additional knowledge of f gained during that step. The updates make use of the fact that changes in the gradient provide information about the second derivative. In this way we use

$$\mathbf{d}_{\text{qn}} = -\mathbf{B}^{-1}\nabla f(\mathbf{x}).$$

In practice, instead of updating the approximating matrix $\mathbf{B}_k$ after every step, it is possible to update its inverse $\mathbf{B}_k^{-1}$, such that the quasi-newton direction can be computed by a simple matrix-vector multiplication.

Quasi-Newton directions can be derived as follows. Observe that by (28) we have for two points $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$

$$\nabla f(\mathbf{x}_{k+1}) \approx \nabla f(\mathbf{x}_k) + [\nabla^2 f(\mathbf{x}_{k+1})](\mathbf{x}_{k+1} - \mathbf{x}_k).$$

We rearrange this to

$$[\nabla^2 f(\mathbf{x}_{k+1})](\mathbf{x}_{k+1} - \mathbf{x}_k) \approx \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k).$$

Now define $\mathbf{s}_k \equiv \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k \equiv \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$, and let $\mathbf{B}_{k+1}$ be our approximation of the Hessian $\nabla^2 f(\mathbf{x}_{k+1})$, which we require to satisfy

$$\mathbf{B}_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{32}$$

Now given some initial approximation $\mathbf{B}_0$, which is often taken to be the identity matrix, we can define an update matrix after every step, such that (32) is satisfied. Because the system is underdetermined, there are many update matrices possible. Therefore we will look for updates which are small in the sense that they maintain the structure of $\mathbf{B}_k$ as much as possible.

Furthermore, it is reasonable to demand that the updates conserve symmetry, because the true Hessian is always symmetric. Positive definiteness of $\mathbf{B}_k$ is also desirable, because if $\mathbf{B}_k$ is not positive definite, we could encounter situations where $\mathbf{d}_{\text{qn}}$ is not a descent direction, or situations where $\mathbf{B}_k$ is not invertible. However, the requirements to be symmetric and positive definite lead to big restrictions on the update matrix. Therefore we will sometimes give up these requirements. In case that the algorithm get stuck we can simply use the steepest descent direction instead.

Now let Δ_k be the update matrix, such that $\mathbf{B}_{k+1} \equiv \mathbf{B}_k + \Delta_k$. Then Δ_k should satisfy $\Delta_k\mathbf{s}_k = \mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k$. The general form of a rank-1 update matrix which satisfies this relation is for some vector $\mathbf{p}$ which is not perpendicular to $\mathbf{s}_k$

$$\Delta_k = \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)\mathbf{p}^T}{\mathbf{p}^T\mathbf{s}_k}. \tag{33}$$

We might think that Δ_k should only carry information in the direction $\mathbf{s}_k$, because in step k we only got information about the variation in ∇f in this direction. So consider $\mathbf{p} = \mathbf{s}_k$,

$$\Delta_k = \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}_k}.$$

Now we have $\Delta_k\mathbf{z} = \mathbf{0}$ whenever $\mathbf{z}$ is perpendicular to $\mathbf{s}_k$. Unfortunately, this update matrix is not symmetric, and therefore it does not lead to a good approximation of the Hessian.

We might ask, does there exist any symmetric update matrix for which $\Delta_k\mathbf{z} = \mathbf{0}$ whenever $\mathbf{z}$ is perpendicular to $\mathbf{s}_k$? The answer is no, except when $\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k$ is a multiple of $\mathbf{s}_k$, which we cannot assume in general. We can see this as follows. Let $\mathbf{z}$ be an arbitrary vector perpendicular to $\mathbf{s}_k$,

and let $\Delta_k \mathbf{z} = \mathbf{0}$, where Δ_k is symmetric. Then

$$0 = \mathbf{s}_k^T \Delta_k \mathbf{z} = \mathbf{s}_k^T \Delta_k^T \mathbf{z} = (\Delta_k \mathbf{s}_k)^T \mathbf{z} = (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{z},$$

but $\mathbf{z}$ is not perpendicular to $\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k$ in general, so there is a contradiction.

Another way to preserve the structure of $\mathbf{B}_k$ is by explicit requiring that the update preserves the relation $\mathbf{B}_k \mathbf{s}_{k-1} = \mathbf{y}_{k-1}$. This means that the update matrix Δ_k should satisfy $\Delta_k \mathbf{s}_{k-1} = \mathbf{0}$. This brings us to

$$\mathbf{p} = \mathbf{s}_k - \frac{\mathbf{s}_k^T \mathbf{s}_{k-1}}{\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}} \mathbf{s}_{k-1}$$

for $\mathbf{p}$ in (33). Unfortunately, the update matrix is not symmetric again.

The only symmetric rank-1 update matrix is given by $\mathbf{p} = \mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k$. This leads to the popular 'symmetric-rank-one' (SR1) formula

$$\Delta_k = \frac{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k}. \tag{34}$$

Unfortunately, because it may happen that $(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k < 0$, positive definiteness is not guaranteed. Moreover, troubles can be expected when $(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k \approx 0$. Because the matrices generated by the SR1 formula appear in practice to be good approximations of the true Hessian, the method is still popular.

When we allow the update matrix to be rank-2, it is possible to ensure positive definiteness. In 1970, the four mathematicians Broyden, Fletcher, Goldfarb and Shanno discovered independently the following formula, which is now known as the *BFGS-formula*,

$$\Delta_k = -\frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}. \tag{35}$$

BFGS generates positive definite approximations whenever the initial approximation $\mathbf{B}_0$ is positive definite, and $\mathbf{y}_k^T \mathbf{s}_k > 0$. This can be seen easily from the inverse update formula below (36). The condition $\mathbf{y}_k^T \mathbf{s}_k > 0$ is very natural: it means that $\mathbf{s}_k^T \nabla f(\mathbf{x}_{k+1}) > \mathbf{s}_k^T \nabla f(\mathbf{x}_k)$, so the directional derivative of f is bigger at the new iterate $\mathbf{x}_{k+1}$ than at the old iterate $\mathbf{x}_k$. If we use a line search procedure to determine the new iterate $\mathbf{x}_{k+1}$, imposing strong Wolfe conditions ensures that $\mathbf{y}_k^T \mathbf{s}_k > 0$ is satisfied.

The BFGS-formula is not the only one which preserves symmetry and positive definiteness. By interchanging the roles of $\mathbf{s}_k$ and $\mathbf{y}_k$ we get the *DFP-formula*, named after Davidon, who proposed this formula in 1959, and subsequently studied, implemented, and popularized by Fletcher and Powell. So the DFP-formula is given by

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

This corresponds to

$$\mathbf{B}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k}\right) \mathbf{B}_k \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{s}_k^T \mathbf{y}_k}\right) + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{s}_k^T \mathbf{y}_k},$$

which can be derived by the formula of Sherman and Morrison [12]. The DFP formula is quite effective, but was soon superseded by the BFGS formula.

Linear combinations of the BFGS and DFP by $\mathbf{B}_{k+1} = (1-\varphi_k)\mathbf{B}_{k+1}^{\text{BFGS}} + \varphi_k \mathbf{B}_{k+1}^{\text{DFP}}$ can also be used. They form the so-called Broyden class of quasi-Newton methods. In practice, BFGS is the most popular method from this class, because it can efficiently be implemented in the form of L-BFGS, as we will show below.

As already noted, smart implementations of these quasi-newton methods do not track and update $\mathbf{B}_k$ itself, but its inverse $\mathbf{B}_k^{-1}$. We can compute them by inversion of the relation (32), so $\mathbf{s}_k = \mathbf{B}_{k+1}^{-1}\mathbf{y}_k$. For the general rank-1 update matrix (34), we see, again by the formula of Sherman and Morrison,

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} + \frac{(\mathbf{s}_k - \mathbf{B}_k^{-1}\mathbf{y}_k)\mathbf{p}^T \mathbf{B}_k^{-1}}{\mathbf{p}^T \mathbf{B}_k^{-1} \mathbf{y}_k}.$$

So we can implement the SR1 formula by

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

The inverse update of the BFGS formula is given by

$$\mathbf{B}_{k+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) \mathbf{B}_k^{-1} \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}. \tag{36}$$

In practice, it is often advantageous to keep the BFGS matrix $\mathbf{B}_{k+1}^{-1}$ factorized, to avoid computationally expensive matrix-matrix multiplications. By saving the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$, the quasi-Newton BFGS direction can be computed by simple inner products and vector updates. To avoid memory issues, it is convenient to keep only a limited number of vectors saved. Moreover, the computation costs per quasi-Newton step would increase at every step if all vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ were saved. So only a limited set of the past vectors should be saved. The method obtained in this way is known as Limited-memory BFGS or L-BFGS. It is particularly well-suited for problems with many unknowns, like training neural networks, Chapter 7.

When we apply quasi-Newton methods to one-dimensional problems, they turn into the *secant method*. In order to satisfy (32), there is only one option,

$$B_{k+1} = \frac{y_k}{s_k} = \frac{f'(x_{k+1}) - f'(x_k)}{x_{k+1} - x_k}.$$

In this way we get

$$x_{k+2} = x_{k+1} - \alpha_k f'(x_{k+1}) \frac{x_{k+1} - x_k}{f'(x_{k+1}) - f'(x_k)},$$

where α_k should be determined by a line search. Observe that if f is quadratic, the minimum is found immediately with $\alpha_k = 1$, so $\alpha_k = 1$ is a good initial guess for the line search.

4.4.4 Conjugate gradients

The Conjugate Gradient (CG) method is well known from solving a linear system of equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric and positive definite matrix. The CG algorithm is given by the recursion

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k & \text{where } \alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{u}_k^T \mathbf{A} \mathbf{u}_k} \\ \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{u}_k & \\ \mathbf{u}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{u}_k & \text{where } \beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \end{cases} \tag{37}$$

with initial guess $\mathbf{x}_0$, and initial values $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{u}_0 = \mathbf{r}_0$. Note that $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ is the residual in step k in this recursion (neglecting rounding errors). To save a matrix-vector multiplication in the algorithm, is chosen to be updated by $-\alpha_k \mathbf{A}\mathbf{u}_k$, for $\mathbf{A}\mathbf{u}_k$ was already in α_k. The vector $\mathbf{u}_k$ is the search direction of the algorithm.

An important property of CG is the fact that the search directions are *conjugate*. This means that $\mathbf{u}_k \mathbf{A} \mathbf{u}_j = 0$ for $j = 0, \ldots, k-1$. Other properties are $\mathbf{r}_k^T \mathbf{r}_j = 0$ and $\mathbf{r}_k^T \mathbf{u}_j = 0$ and $\mathbf{r}_k^T \mathbf{A} \mathbf{u}_j = 0$ for $j = 0, \ldots, k-1$. A proof can be found in [12].

Now observe that solving the equation $\mathbf{Ax} = \mathbf{b}$ is equivalent to minimizing

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x}. \tag{38}$$

Hence, we can interpret the CG algorithm as a minimization algorithm. We see that the gradient of f is equal to the residual of the linear system times -1,

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = -\mathbf{r}.$$

So if we replace in the CG-algorithm the residuals by minus the gradients of f, and if we compute the stepsize α_k by a line search, we can apply the resulting algorithm to nonlinear optimization problems.

The computed search directions of non-linear CG are $\mathbf{d}_{\mathrm{CG},0} = -\nabla f(\mathbf{x}_0)$ and

$$\mathbf{d}_{\mathrm{CG},k} = -\nabla f(\mathbf{x}_k) + \beta_k \mathbf{d}_{\mathrm{CG},k-1} \qquad (k \geq 1),$$

where β_k is such that $\mathbf{d}_{\mathrm{CG},k}$ and $\mathbf{d}_{\mathrm{CG},k-1}$ are conjugate. Following ordinary CG, this means that we can use

$$\beta_k = \frac{\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_{k-1})^T \nabla f(\mathbf{x}_{k-1})}, \tag{39}$$

which was proposed by the of developers of non-linear CG, Fletcher and Reeves. But now there is no matrix $\mathbf{A}$ any more, so it is not obvious what it really means that $\mathbf{d}_{\mathrm{CG},k}$ and $\mathbf{d}_{\mathrm{CG},k-1}$ are

conjugate. In fact, different choices of β_k are possible, it is just which meaning is given to the requirement that $\mathbf{d}_{\mathrm{CG},k}$ and $\mathbf{d}_{\mathrm{CG},k-1}$ should be conjugate. By demanding that the consecutive search directions $\mathbf{d}_{\mathrm{CG},k}$ and $\mathbf{d}_{\mathrm{CG},k+1}$ should be conjugate with respect to the average Hessian over the line segment $[\mathbf{x}_k, \mathbf{x}_{k+1}]$, it can be shown that then we get the Hestenes-Stiefel formula [12]

$$\beta_k = \frac{\nabla f(\mathbf{x}_k)^T(\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1}))}{\mathbf{d}_{\mathrm{CG},k-1}^T(\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1}))}. \tag{40}$$

If we do exact line searches to determine the stepsize, then the directional derivative of f at $\mathbf{x}_k$ in the previous search direction equals zero. So then we have $\mathbf{d}_{\mathrm{CG},k-1}^T \nabla f(\mathbf{x}_k) = 0$. The Polak-Ribière choice applies this relation to the Hestenes-Stiefel formula, which becomes

$$\beta_k = \frac{\nabla f(\mathbf{x}_k)^T(\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1}))}{\nabla f(\mathbf{x}_{k-1})^T \nabla f(\mathbf{x}_{k-1})}. \tag{41}$$

The Polak-Ribière formula is generally considered to be the best choice for β_k, even if we do inexact line searches.

Note that in case we have f as in (38), then the choices of β_k by Hestenes-Stiefel (40) and Polak-Ribière (41) coincide with the choice of Fletcher-Reeves (39), because of the orthogonality of the gradients, and the orthogonality of the gradients with respect to the search directions as we mentioned before. So the differences between these conjugate gradient method lies in the fact how they handle the variations in the Hessian.

Observe that it makes no sense to apply the non-linear conjugate gradients method to one-dimensional problems. This reduces simply to steepest descent.

4.5 Inexact line search

In this section we discuss how to perform an inexact line search, in order to get a proper stepsize t in Algorithm 1. We will formulate some conditions on a solution of the inexact line search, and present a practical algorithm, which is given in Algorithm 2 together with Algorithm 3.

Consider $\phi(t) \equiv f(\mathbf{x}_k + t\mathbf{d})$, where $\mathbf{d}$ is a descent direction of f at $\mathbf{x}_k$. So $\phi'(0) = \mathbf{d}^T \nabla f(\mathbf{x}_k) < 0$. The ideal stepsize t would be the global minimizer of ϕ, but that is in general computationally too expensive. To find even a local minimizer requires normally many (expensive) evaluations of the objective function f. However, minimizers of ϕ are generally no minimizers of f, except when f is a function of only one variable. So if f is a function of several variables, it is not significant to minimize ϕ always exactly. Therefore, it is more practical to perform an *inexact* line search, to identify an adequate stepsize at minimal cost.

It is not directly evident which conditions should be met at t for an inexact line search. For some exact line search, we can simply demand that $\phi'(t)$ is zero. But for inexact line searches we need

a different kind of conditions. A very common inexact line search condition is *sufficient decrease* in ϕ. This can be checked by the inequality

$$\phi(t) \leq \phi(0) + c_1 t\phi'(0), \tag{42}$$

for some $c_1 \in (0, 1)$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$. Next, to rule out unacceptably short steps, there is the *curvature condition*,

$$\phi'(t) \geq c_2\phi'(0), \tag{43}$$

for some constant $c_2 \in (c_1, 1)$. According to [12], typical values are $c_2 = 0.9$ when the search direction is chosen by a Newton or quasi-Newton method, and $c_2 = 0.1$ if the search direction is obtained from a nonlinear conjugate gradient method.

The sufficient decrease (42) and curvature conditions (43) together are known as the *Wolfe conditions*. Because there are stepsizes which satisfy the Wolfe conditions without being particularly close to a minimizer of ϕ, the curvature condition can be modified into

$$|\phi'(t)| \leq c_2|\phi'(0)|, \tag{44}$$

for some constant $c_2 \in (c1, 1)$. Now it is not longer allowed that the derivative ϕ' for stepsize t is too positive. The conditions (42) and (44) together are known as the *strong Wolfe conditions*.

A procedure which finds a stepsize t which satisfies the strong Wolfe conditions (and the ordinary Wolfe conditions) is given in Algorithm 2. It uses the subroutine **zoom**, which is given in Algorithm 3. The algorithm splits the line search in two parts. First the initial stepsize is increased in Algorithm 2 until a acceptable stepsize or a interval bracketing acceptable stepsizes is found. Thereafter, a bisection or interpolation phase computes a acceptable stepsize within this interval. This is carried out by Algorithm 3. Both algorithms are from [12].

A few remarks on Algorithm 2 and Algorithm 3. On the first line of Algorithm 2, there is asked to select an initial guess $t_1 > 0$. For Newton and quasi-Newton methods, always should be used unit stepsize $t_1 = 1$ in order to get the fast convergence properties. For methods which do not have such scaling properties, a popular strategy is to take as initial guess the same as the stepsize obtained in previous step. This is based on the assumption that the first-order change of the function at $\mathbf{x}_k$ is the same as obtained at the previous step.

Next, Algorithm 2 asks to select some $t_{j+1} > t_j$ at the last-but-third line. Here we can for example double the stepsize, and take $t_{j+1} = 2t_j$. Next, in Algorithm 3 we see the variables t_{lo} and t_{hi}). These variables bracket an interval with acceptable stepsizes, and **zoom** reduces this interval, until it finds an acceptable stepsize. On the second line we have to choose some t between t_{lo} and t_{hi},

Algorithm 2 Line search with Wolfe conditions

Set $t_0 \leftarrow 0$, select $t_1 > 0$
$j \leftarrow 1$
repeat
  Evaluate $\phi(t_j)$
  if $\phi(t_j) > \phi(0) + c_1 t_j \phi'(0)$ **or** $[\phi(t_j) \geq \phi(t_{j-1})$ **and** $j > 1]$ **then** ▷ Sufficient decrease
    $t \leftarrow$ **zoom**(t_{j-1}, t_j)
    stop
  Evaluate $\phi'(t_j)$
  if $|\phi'(t_j)| \leq -c_2 \phi'(0)$ **then** ▷ Curvature condition
    $t \leftarrow t_j$
    stop
  if $\phi'(t_j) \geq 0$ **then**
    $t \leftarrow$ **zoom**(t_j, t_{j-1})
    stop
  Select $t_{j+1} > t_j$
  $j \leftarrow j + 1$
end (**repeat**)
return t

Algorithm 3 Subroutine **zoom**$(t_{\text{lo}}, t_{\text{hi}})$

repeat
  Select (by interpolation or bisection) t between t_{lo} and t_{hi})
  Evaluate $\phi(t)$
  if $\phi(t) > \phi(0) + c_1 t \phi'(0)$ **or** $\phi(t) \geq \phi(t_{\text{lo}})$ **then** ▷ Sufficient decrease
    $t_{\text{hi}} \leftarrow t$
  else
    Evaluate $\phi'(t)$
    if $|\phi'(t)| \leq -c_2 \phi'(0)$ **then** ▷ Curvature condition
      return t
    if $\phi'(t)(t_{\text{hi}} - t_{\text{lo}}) \geq 0$ **then**
      $t_{\text{hi}} \leftarrow t_{\text{lo}}$
    $t_{\text{lo}} \leftarrow t$
  end if
end (**repeat**)
return t

for example $t = \frac{1}{2}(t_{\text{lo}} + t_{\text{hi}})$. However, the algorithm can here be refined by using interpolation techniques. Now if this t is not acceptable, the algorithm chooses to cut the interval at this t, such that t becomes the new t_{hi} or t_{lo} in some logical way. Here, the algorithm keeps t_{hi} and t_{lo} in the relation $\phi(t_{\text{lo}}) \leq \phi(t_{\text{hi}})$.

Note that the Wolfe conditions need the value of $\phi'(t)$, and so the gradient of f because $\phi'(t) = \nabla f(\mathbf{x}_k + t\mathbf{d})$. We might think that it is more advantageous to use this gradient to determine a new search direction, than to wait until the inexact line search terminates. Elaborating this idea leads us to trust region methods we already mentioned.

Apart from the Wolfe conditions, it is also possible to use the *Goldstein conditions*. These conditions ensure like the Wolfe conditions that the stepsize achieves sufficient decrease while preventing too small steps. They are given by

$$\phi(0) + (1-c)t\phi'(0) \leq \phi(t) \leq \phi(0) + ct\phi'(0), \tag{45}$$

for some $0 < c < \frac{1}{2}$. The Goldstein conditions are a bit less popular, because the first inequality may exclude all minimizers of f. However, they need not the computation of $\phi'(t)$.

4.6 Concluding remarks

Now we have some choices for descent directions and stepsizes, so we can solve minimization problems. But which method is most efficient? That depends of course partially on the problem we want to solve. Commonly, quasi-Newton and nonlinear CG methods are better than the steepest descent method or the Newton method. When we compare the best nonlinear CG method and the best quasi-Newton method, which are the Polak-Ribière method and the L-BFGS method, there is no clear preference. The quasi-Newton BFGS method converges in general in less iterates to a local minimum than the Polak-Ribière CG method, but BFGS needs more computational work per step. However, concerning problems with many variables, literature indicates that L-BFGS is the more rapid and robust one, based on computational experience. Therefore L-BFGS has become the method of choice for large problems. [12]

5 Modeling received signal strength

In Section 2.3 and Section 2.4 we described theoretically the relation between signal strength and distance, and derived some formulas. But do these theoretical formulas properly describe the practical behavior of the signal strength with respect to distance? And how do the disturbances caused by incorrect alignment, obstructions, multipath fading and anything else look like?

If we could answer these questions completely, and we would know the exact circumstances during a measurement, we could predict exactly the received signal strength by the physical laws of electromagnetic radiation. However, the circumstances are almost always not fully known. Moreover, the signal strength is very sensitive to small changes in the circumstances, so it is usually impossible to predict exactly the received signal strength.

However, while calculation of exact distances is not realistic, we can often be satisfied with approximate results. But which accuracy can be reached? Of course, this depends on the intensity of the disturbances. In this way, we have to find out the intensity and the shape of the disturbances in the signal strength. In this chapter, we will study first the disturbances on the received signal strength, using measurements obtained in a common industrial room with many multipath situations. Here we will see that the exponential probability distribution is very usable to model the disturbances. After this has become clear, we consider the overall relation between distance and the received signal strength.

The structure of this chapter is as follows. Section 5.1 explains the theoretics of the methodology we use, Section 5.2 discusses which probability density function fits to our measurements on active and passive tags. Thereafter, Section 5.3 discusses the relation between received signal strength and distance for both active and passive tags on the basis of measurements. Finally, we will summarize the results in a received signal strength model in Section 5.4. Here, we will also see how we can simulate the received signal strength by a computer pseudo-random number generator.

5.1 Theoretical considerations

We have to investigate the *global* relationship between received signal strength and distance, and the *local* relation between signal strength and disturbances. Because the disturbances are very intense, they obscure the global relationship between the received signal strength and distance. Therefore, we first clarify the possible causes of these disturbances in this section. In Section 5.1.1 we take a closer look to the effects of antenna misalignment.

In Section 3.4 we discussed histograms, which we will use to analyze the distribution of the disturbances. We also discussed already how to transform histograms. Now in Section 5.1.2 we will discuss how to transform histograms of received signal strength data from dBm-space to mW-space.

In Section 2.3 we derived the Friis transmission equation (5), which describes the global relation between signal strength and distance. There we also mentioned a few causes of disturbances. In our application, the following sources of disturbances matter.

1. Incorrect antenna alignment.
2. Multipath fading.
3. Absence of line of sight.
4. For passive tags: nonlinear effects in tag electronics.

Concerning (5), we can safely assume that P_T, the transmitted power is constant. We will also assume that the gain factors G_T and G_R are constant for all directions of interest. The wavelength λ is also approximately constant, so we can rewrite Friis equation to

$$P = Cr^{-2}, \tag{46}$$

where $P \equiv P_R$ is the received signal strength, $C = P_T G_T G_R \left(\frac{\lambda}{2\pi}\right)^2$ is constant. However, as in Section 2.6 the assumption of constant G_R and G_T is not exactly true. But it simplifies the Friis equation a lot, and so we hope that it does not lead to big deviations. We will note it as an extra source cause of disturbances with respect to (46).

5. Antenna gain varies with respect to direction.

Now for some constant C, we can write the expected received signal strength P as function of distance r as

$$P(r) = Cr^{-\gamma}, \tag{47}$$

where $\gamma = 2$ for active tags, and we expect $\gamma = 4$ for passive tags. Because of nonlinear effects in tag electronics (item 4 above), γ can be a bit lower in practice. Therefore we will involve γ to the maximum likelihood estimation in next sections.

Concerning item 3 above, the presence or absence of line of sight, it is almost impossible to predict how this affects the received signal strength. It depends on the size and nature of object which blocks the line of sight. For example, objects made of water or steel block radio wave more than object of plastic or wood. Moreover, we have to count for whether there are secondary paths for the radio wave (for example via some wall). In case that there is a line of sight, these secondary paths are undesirable, for they cause multipath fading, item 2 above. But when the line of sight is blocked, these secondary paths help to receive still a signal.

In Section 2.3 we already explained that the influence of multipath fading is considerably in indoor situations. Unfortunately, multipath fading is rather unpredictable, and therefore it is not possible to change (47) accordingly. We hope that we can model it by some probability distribution. However, it is unclear what this probability distribution should be. Therefore, we will simply rely on measurements.

Concerning the incorrect antenna alignment (item 1 above), in Section 2.3 we proposed a fix by multiplicating Friis equation by $\cos^\gamma \theta$, where $\theta \in [0, \frac{1}{2}\pi]$ is the rotation angle of the spot antenna with respect to the tag antenna (6). However, the angle θ is in general unknown, and so this does not help much. But in case we have multiple receivers, whereof the mutual orientation is known, we can use it. For example, the Intellifi spots can use its UHF antenna in two orientations, where the second orientation is quater-turned with respect to the first orientation. In this way, we have $P_0 = Cr^{-\gamma} \cos^\gamma \theta$ for the first orientation, and $P_1 = Cr^{-\gamma} \cos(\theta - \frac{1}{2}\pi) = Cr^{-\gamma} \sin^\gamma \theta$ for the second orientation. Now observe

$$\left(P_0^{\frac{2}{\gamma}} + P_1^{\frac{2}{\gamma}}\right)^{\frac{\gamma}{2}} = Cr^{-\gamma}\left((\cos^\gamma \theta)^{\frac{2}{\gamma}} + (\sin^\gamma \theta)^{\frac{2}{\gamma}}\right)^{\frac{\gamma}{2}} = Cr^{-\gamma}.$$

So we can eliminate θ for the UHF antenna. We can also estimate the angle θ with

$$\arctan\left(\frac{P_1}{P_0}\right)^{\frac{1}{\gamma}} = \arctan\left(\frac{\sin^\gamma \theta}{\cos^\gamma \theta}\right)^{\frac{1}{\gamma}} = \theta.$$

So in case we do not have multiple receives with known orientations, we can not do better than simply using (47). Consequently, we view the antenna misalignment effects as a source of disturbances on the received signal strength. In Section 5.1.1 we describe how these disturbances look like in the form of some probability distribution. Now if the misalignment effects dominate the other disturbances, our measurements we will be similar to that probability distribution. In the other disturbance sources appear to be more important, we will find a different distribution.

5.1.1 Antenna alignment

In this section, we derive a probability distribution function which describes the disturbances on the received signal strength caused by antenna misalignment. Unfortunately, we are not able to derive a probability density distribution which describes the disturbances by multipath fading. But if the disturbances by antenna misalignment dominate the other disturbances, this imposes some specific structure in a histogram of signal strength measurements, where orientation angle θ has been uniformly sampled. If we find a different structure, apparently something else dominates the disturbances.

Now let us assume that the disturbances on received signal strength P at some fixed distance r are dominated by antenna misalignment. In order to find the probability distribution of these disturbances, we use the Probability Integral Transform, Theorem 2. So we have to write the signal strength P as a function of the standard uniform variable U. This is done by the function F_P^{-1}. Now after inversion and differentiation, we have found the probability density function f_P. Because cumulative density functions and their inverse functions are increasing functions, we should ensure that F_P^{-1} is increasing.

Let us take $P = C \cos^\gamma \theta$, where $\theta \in [0, \frac{\pi}{2}]$, and $C > 0$ is some constant which includes the factor $r^{-\gamma}$, because we have fixed distance. If we treat θ like a random variable with uniform distribution

on $[0, \frac{\pi}{2}]$, then $\frac{\pi}{2}(1-U)$ has the same distribution as θ, and we find

$$P = C\cos^\gamma\theta = C\cos^\gamma\big(\tfrac{\pi}{2}(1-U)\big) = F_P^{-1}(U).$$

Here, F_P^{-1} is increasing in U on $[0,1]$, and so it is a valid inverse function of a cumulative density function. Inversion gives

$$U = F_P(P) = 1 - \tfrac{2}{\pi}\arccos\big((C^{-1}P)^{1/\gamma}\big)$$

on the domain $[0,1]$. After differentiation we find

$$\frac{\mathrm{d}F_P(x)}{\mathrm{d}x} = \frac{2(C^{-1}x)^{\frac{1}{\gamma}-1}}{\pi\gamma C\sqrt{1-(C^{-1}x)^{2/\gamma}}} = f_P(x)$$

for $x \in (0, C)$. So P has probability density function

$$f_P(x) = \begin{cases} \dfrac{2(C^{-1}x)^{1/\gamma}}{\pi\gamma x\sqrt{1-(C^{-1}x)^{2/\gamma}}} & \text{if } x \in (0,C), \\ 0 & \text{otherwise.} \end{cases} \tag{48}$$

For $\gamma = 2$, this is simply

$$f_P(x) = \begin{cases} 1\Big/\Big(\pi\sqrt{x(C-x)}\Big) & \text{if } x \in (0,C), \\ 0 & \text{otherwise.} \end{cases}$$

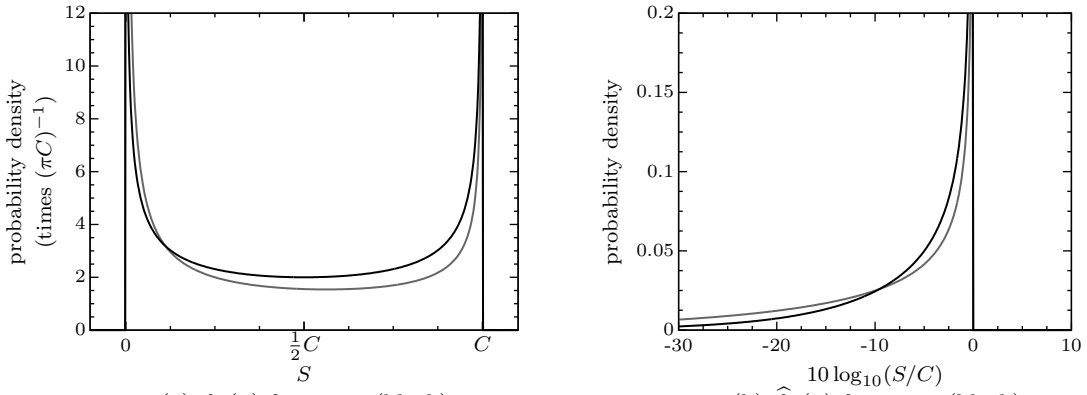
Using Theorem 1 we can easily transform $f_P(x)$ to dBm-space. We get

$$\widehat{f}_P(x) = \frac{\log(10)}{10}10^{x/10}f_P\big(10^{x/10}\big). \tag{49}$$

This probability density function is shown in Figure 3. From this figure, it becomes clear that there is no much difference between the probability density functions for $\gamma = 2$ and $\gamma = 4$. It only draws attention that for $\gamma = 4$, the chance of receiving weaker signal strengths is a bit bigger than in the case we have $\gamma = 2$.

The fact that the probability functions f_P for $\gamma = 2$ and $\gamma = 4$ does not differ much is still more emphasized by the fact that they both induce the same maximum likelihood estimation of C. So when we have a couple of measurements p_k for $k = 0, 1, \ldots, n-1$, it can easily be seen that the maximum likelihood estimation of C in $f_P(x)$ is $C = \max_k p_k$.

So if the tag orientation angle dominates the received signal strength given some fixed distance, then we expect (48) or a very similar probability distribution of the signal strength measurements, in case that we sample θ uniform, and under the condition that all other variables are constant. However, that is never the case. Moreover, the singularity of $f_P(x)$ at $x = C$ is unreliable: there is always some random noise. Next, in (48) we did not take multipath effects into account. Unfor-



(a) $f_P(x)$ for $\gamma = 2$ (black) and $\gamma = 4$ (gray).

(b) $\widehat{f}_P(x)$ for $\gamma = 2$ (black) and $\gamma = 4$ (gray).

Figure 3: Left probability density function f_P from (48) for $\gamma = 2$ and $\gamma = 4$. Right transformed to dBm-space, equation (49). If we sample tag orientation angle θ uniformly and this is the dominating component in the received signal strength, we would find from our measurement a probability distribution which looks like this.

tunately, it is unclear to which extent these effects affect the signal strength. Therefore, we have to rely on measurements.

We can also conclude that it is not a good idea to check γ by sampling the signal strength over the orientation angle. The probability density function f_P is for $\gamma = 2$ and $\gamma = 4$ too similar. It is better to measure γ by taking signal strength samples at distinct distances.

5.1.2 Histograms of received signal strength data

The Intellifi equipment measures signal strength in dBm-space. Because it can be difficult to recognize a proper probability distribution in a histogram on dBm-space, we show here how to transform the histogram to mW-space. In Section 3.4 we derived general transforms in (10) and (11).

Now we see that the transform from dBm-space to mW-space can be described by the function $T(x) = 10^{x/10}$. So (10) becomes

$$\widehat{h}(\widehat{B}_k) = \frac{b_k^+ - b_k^-}{10^{b_k^+/10} - 10^{b_k^-/10}} h(B_k). \tag{50}$$

Furthermore, we see $T^{-1}(x) = 10\log_{10}(x)$, and $\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x} = \frac{10}{x\log(10)}$ for $x > 0$. Then (11) becomes

$$\widehat{h}(x) = \frac{10}{x\log(10)} h(B_k) \qquad \text{for } x \in \widehat{B}_k. \tag{51}$$

Because the Intellifi equipment rounds the received signal strengths to certain discrete points, we

find no continuum of values in our measurements, but all measurements are at discrete points. For active tags, the equipment rounds the signal strength to whole dBm's. For passive tags, the equipment follows a more irregular rounding pattern, but the resulting resolution is comparable. At some parts of the domain the discrete points are only one half dBm apart, and at other points the discrete points are slightly more than a whole dBm apart. Now if we take the bins B_k to have all exactly one such discrete point, and we take the bin edges to be the midpoints of those discrete points, we get well-shaped histograms.

Now also transforming by (50) or (51) differs almost nothing. We can see that as follows. Take $b_k = \frac{1}{2}(b_k^- + b_k^+)$ to be the midpoint of bin B_k. Then $b_k^+ = b_k + \frac{1}{2}|B_k|$ and $b_k^- = b_k - \frac{1}{2}|B_k|$. Now for $x \in \widehat{B}_k$, we have in (51)

$$\frac{\widehat{h}(x)}{h(B_k)} \approx \frac{10}{10^{b_k/10}\log(10)}.$$

At the other side, in (50) we have simply

$$\frac{\widehat{h}(\widehat{B}_k)}{h(B_k)} = \frac{b_k^+ - b_k^-}{10^{b_k^+/10} - 10^{b_k^-/10}} = \frac{|B_k|}{10^{(b_k+\frac{1}{2}|B_k|)/10} - 10^{(b_k-\frac{1}{2}|B_k|)/10}}.$$

Now we take the quotient of both results, and get

$$\frac{10\big(10^{(b_k+\frac{1}{2}|B_k|)/10} - 10^{(b_k-\frac{1}{2}|B_k|)/10}\big)}{|B_k|10^{b_k/10}\log(10)} = \frac{10\big(10^{|B_k|/20} - 10^{-|B_k|/20}\big)}{|B_k|\log(10)}.$$

This depends only on the bin width $|B_k|$. If we take $|B_k| = 1$ dBm, we find that the quotient approximates only 1.002.

Once we have recognized a probability distribution in the histogram data, the next step is to fit this probability distribution to the data. For this we will use the maximum likelihood principle, as discussed in Section 3.5.

5.2 Probability distribution estimation by histograms

In this section we evaluate signal strength measurements in order to determine a probability density function which fits to this data. First we will evaluate signal strength measurements on active tags, thereafter we will evaluate signal strength measurements on passive tags.

Active tags

Concerning active tags, in order to find a probability distribution function which approaches the probability distribution of the signal strength, we have done a large number of measurements at various distances, see Figure 4. These measurements are obtained by 1 tag and 4 receivers used simultaneously to get more data in less time. During the measurements the tag location was changed in a systematic way, at fixed times, in order to get a well-balanced data set. Because of rounding effects, there are many stacked circles in the graphic, which distort the picture a bit. The

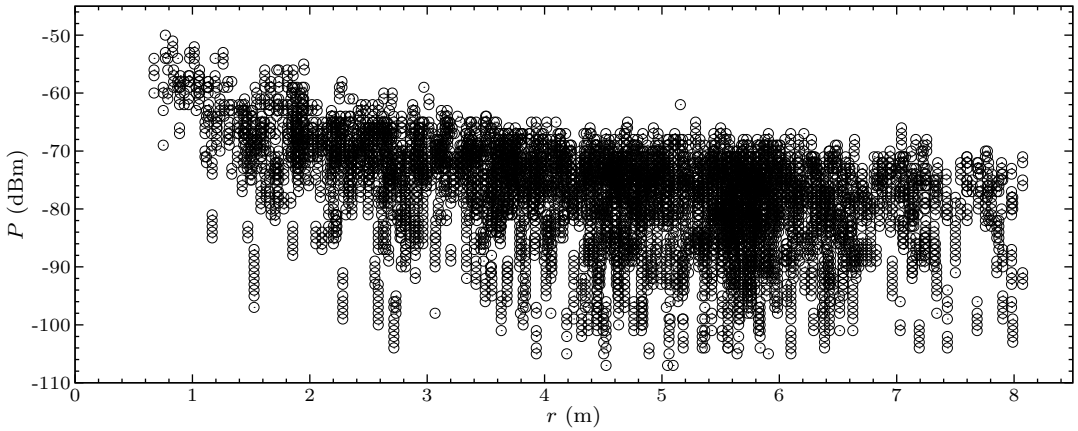


Figure 4: Signal strength measurements (vertical, in dBm) versus distance (horizontal, in meters) from an active Bluetooth tag. Because measurements are rounded to whole dBm's, the graphic consists from many stacked circles.

4 distinct receivers were not exactly identical, but they had the same design, so we assume that they are identical enough to compare their received signal strengths.

From Figure 4 become a few things clear, notwithstanding the rounding effects. As expected, even at small distances ($r < 3$), signal strengths below -100 dBm are possible. In fact, the lower limit of what the equipment is able to receive is about -100 dBm. So the lower limit of the received signal strength is mainly determined by the range of the receiving equipment. Next, apart from a few outliers, there is an obvious maximum to the received signal strength, which slightly decays over bigger distances. However, the presence of the few outliers induce that the maximum is not always a good characterization of the measurement set, and so we likely need a probability density function different from (48).

Now we take a closer look at the measurements obtained at distances between 3.5 and 4.5 meter. We have drawn a normalized histogram of them, see Figure 5a. This shows a skewed Gaussian-like distribution. When we transform to mW-scale as described in Section 5.1, we get the histogram in Figure 5b. Here, we can clearly recognize an exponential probability distribution, which has probability density function f_λ for some parameter λ, as defined in (15),

$$f_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{52}$$

Using Theorem 1 we can transform this to dBm-scale. We take $T(x) = 10\log_{10}(x)$, and so the inverse is $T^{-1}(x) = 10^{x/10}$. Then $\frac{\mathrm{d}T^{-1}(x)}{\mathrm{d}x} = \frac{1}{10}\log(10)10^{x/10}$, and so

$$\widehat{f}_\lambda(x) = \frac{\log(10)}{10}\lambda\, 10^{x/10} e^{-\lambda\, 10^{x/10}}. \tag{53}$$

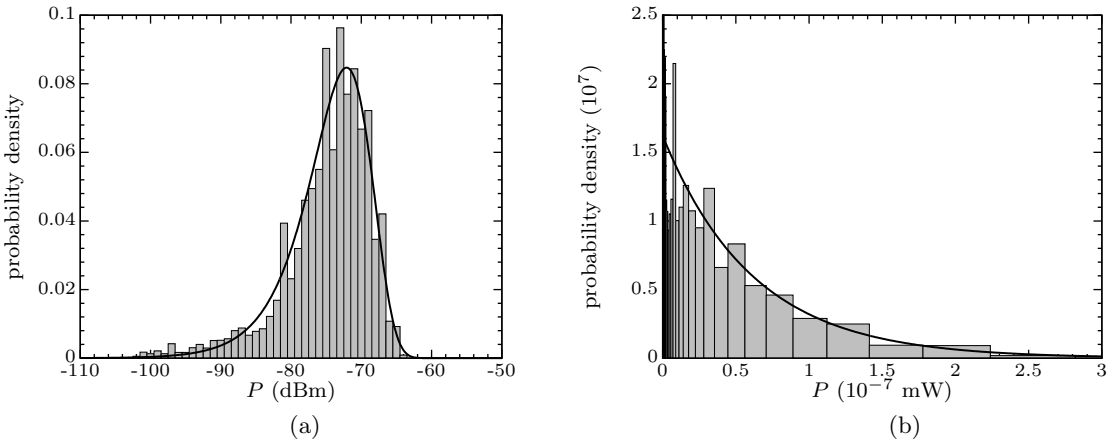


Figure 5: Histograms of 14225 measurements at distances between 3.5 and 4.5 meter. Left is the original histogram on dBm-scale, right is the histogram transformed to mW-scale. The curves are a fit from an exponential distribution, according to the method of maximum likelihood.

The cumulative density function is given in (16),

$$F_\lambda(x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{54}$$

We can also transform this to dBm-scale by Theorem 1,

$$\widehat{F}_\lambda(x) = 1 - e^{-\lambda\, 10^{x/10}}. \tag{55}$$

This is much different from (48), and it is not likely that the orientation angle dominates the received signal strength. Based on the analysis in Section 5.1, multipath fading seems to be a more important disturbance source.

Passive tags

Concerning passive tags, we have unfortunately much less measurements, than we have from active tags. And moreover, these measurements are obtained less systematic. Therefore the results for passive tags are a bit flexible.

Here, all measurements are taken from one spot and one tag. The spot was placed in the ceiling with the antenna directed downwards. The tag was placed in the space below the spot. We took measurements for many different directions and angles, but everywhere the distance between the tag and the spot was 1.0 meter. See Figure 6 for histograms of this data.

Now we have to examine whether the received signal strength of passive tags can be modeled by an exponential probability distribution, like we have for passive tags. The picture in Figure 6 is

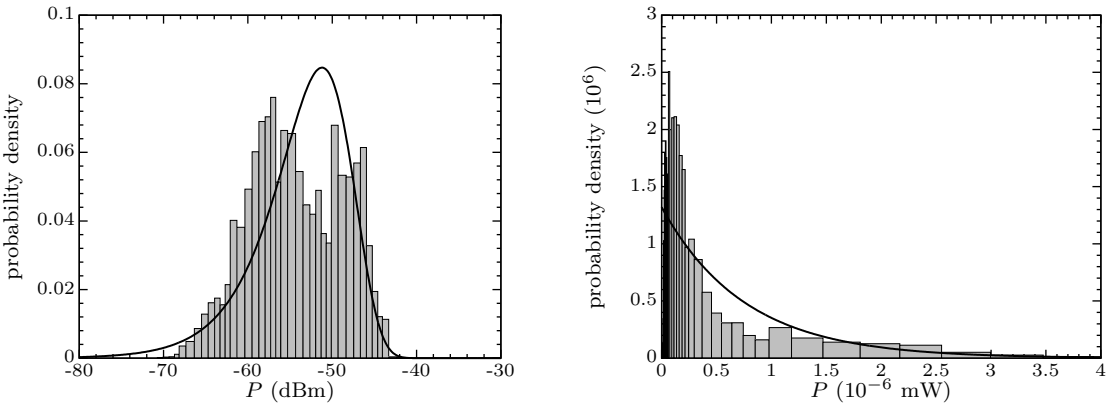


Figure 6: Graphs of 27070 measurements with alternating polarizations. All measurements were obtained at distance 1 meter. Left is the normalized histogram on dBm-scale, right is the histogram transformed to mW-scale. The curves are a fit from an exponential distribution, λ^{-1} is estimated as $7.57 \cdot 10^{-6}$, which corresponds to -51.2 on dBm-scale.

not very convincing, but the deviations are possibly caused by the fact that the measurements did not sample all directions and orientations uniformly. So we can conclude that more research is needed, but for now, we will assume that signal strength measurements from passive tags show an exponential probability distribution.

5.3 Relation received signal strength and distance

In this section we will estimate the relation between parameter λ in (15) and distance r. In Section 5.2 we concluded that the disturbances in the received signal strength can be estimated by an exponential distribution (52) with parameter λ. The expected value of the exponential distribution is λ^{-1}, as derived in (17). Because we expect at distance r to receive signal strength $Cr^{-\gamma}$ as in (47), we would have $\lambda^{-1} = Cr^{-\gamma}$. So we take

$$\lambda(r) = \lambda(r, C, \gamma) = C^{-1} r^{\gamma}, \tag{56}$$

where we expect $\gamma = 2$ for active tags, and $\gamma = 4$ for passive tags. We will estimate C and γ by a maximum likelihood estimation.

This maximum likelihood estimation can be derived as follows. Assume that we have signal strength measurements $p_0, \ldots, p_{n-1}$ at distances $r_0, \ldots, r_{n-1}$. Then we have to optimize

$$\mathcal{L}(C, \gamma) = \sum_{k=0}^{n-1} \log\Big(\lambda(r_k, C, \gamma) e^{-p_k \lambda(r_k, C, \gamma)}\Big) = \sum_{k=0}^{n-1} \log\big(\lambda(r_k, C, \gamma)\big) - p_k \lambda(r_k, C, \gamma).$$

This is

$$\mathcal{L}(C,\gamma) = \sum_{k=0}^{n-1} \log(C^{-1} r_k^\gamma) - p_k C^{-1} r_k^\gamma = -n\log(C) + \sum_{k=0}^{n-1} \Big(\gamma \log(r_k) - \frac{p_k r_k^\gamma}{C}\Big).$$

and so we have to solve

$$\begin{cases} \dfrac{d\mathcal{L}(C,\gamma)}{dC} = -\dfrac{n}{C} + \dfrac{1}{C^2}\displaystyle\sum_{k=0}^{n-1} p_k r_k^\gamma = 0, \\ \dfrac{d\mathcal{L}(C,\gamma)}{d\gamma} = \displaystyle\sum_{k=0}^{n-1} \Big(\log(r_k) - \frac{\log(r_k)}{C} p_k r_k^\gamma\Big) = 0. \end{cases}$$

From the first equation we obtain

$$C = \frac{1}{n}\sum_{k=0}^{n-1} p_k r_k^\gamma. \tag{57}$$

In the second equation we have

$$C\sum_{k=0}^{n-1} \log(r_k) = \sum_{k=0}^{n-1} \log(r_k) p_k r_k^\gamma.$$

Substitution of C gives

$$\left(\frac{1}{n}\sum_{k=0}^{n-1} p_k r_k^\gamma\right)\left(\sum_{k=0}^{n-1} \log(r_k)\right) = \sum_{k=0}^{n-1} \log(r_k) p_k r_k^\gamma.$$

So

$$\frac{\sum_{k=0}^{n-1} \log(r_k) p_k r_k^\gamma}{\frac{1}{n}\sum_{k=0}^{n-1} p_k r_k^\gamma} = \sum_{k=0}^{n-1} \log(r_k). \tag{58}$$

This equation can be numerically solved for γ, for example by the secant method.

In case of our measurements in Figure 4, we used multiple receivers, which were equipped with hand-made antennas. Therefore, there may be a bit difference between the gain factors of these antennas, and so we should estimate the factor C in (56) for each spot apart. The exponent γ is assumed to do not depend on antennas, so we take it to be equal for all spots.

Let m be the number of spots we have in our measurement, and take for spot j the relation $\lambda(r, C_j, \gamma) = C_j^{-1} r^\gamma$. If we have for spot j signal strength measurements $p_{j,0}, \ldots, p_{j,n_j-1}$ at distances $r_0, \ldots, r_{n_j-1}$, then we can estimate $C_0, \ldots, C_{m-1}$ and γ by maximizing

$$\begin{aligned} \mathcal{L}(C_0, \ldots, C_{m-1}, \gamma) &= \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} \log(C_j^{-1} r_{j,k}^\gamma) - p_{j,k} C_j^{-1} r_{j,k}^\gamma \\ &= \sum_{j=0}^{m-1} \Big(-n_j \log(C_j) + \sum_{k=0}^{n-1} \Big(\gamma \log(r_{j,k}) - \frac{p_{j,k} r_{j,k}^\gamma}{C_j}\Big)\Big). \end{aligned}$$

This leads to the system of equations

$$\begin{cases} \dfrac{d\mathcal{L}(C_0,\ldots,C_{m-1},\gamma)}{dC_j} = -\dfrac{n_j}{C_j} + \dfrac{1}{C_j^2}\displaystyle\sum_{k=0}^{n_j-1} p_{j,k} r_{j,k}^{\gamma} = 0 & \text{for } j = 0,\ldots,m-1, \\ \dfrac{d\mathcal{L}(C_0,\ldots,C_{m-1},\gamma)}{d\gamma} = \displaystyle\sum_{j=0}^{m-1}\sum_{k=0}^{n_j-1} \Big(\log(r_{j,k}) - \dfrac{\log(r_{j,k})}{C_j} p_{j,k} r_{j,k}^{\gamma}\Big) = 0. \end{cases}$$

We can solve C_j by

$$C_j = \frac{1}{n_j}\sum_{k=0}^{n_j-1} p_{j,k} r_{j,k}^{\gamma}.$$

Hence we get for γ

$$\sum_{j=0}^{m-1}\Big(\sum_{k=0}^{n_j-1}\log(r_{j,k}) - \frac{\sum_{k=0}^{n_j-1}\log(r_{j,k}) p_{j,k} r_{j,k}^{\gamma}}{C_j}\Big) = \sum_{j=0}^{m-1}\Big(\sum_{k=0}^{n_j-1}\log(r_{j,k}) - \frac{\sum_{k=0}^{n_j-1}\log(r_{j,k}) p_{j,k} r_{j,k}^{\gamma}}{\frac{1}{n_j}\sum_{k=0}^{n_j-1} p_{j,k} r_{j,k}^{\gamma}}\Big) = 0.$$

This can also be solved by the secant method. Now let us first compute the results for measurements on active tags, thereafter we turn to passive tags.

Active tags

For our measurements in Figure 4, we find $\gamma = 1.972$, which is approximately equal to 2, which we expected. Furthermore, we find values of C between $0.65 \cdot 10^{-6}$ mW and $1.51 \cdot 10^{-6}$ mW, which corresponds to -61.9 dBm and -58.2 dBm. In Figure 7 we have drawn the same measurements as in Figure 4, but now rescaled by factor C^{-1}. We also drawn (on the dBm scale) the maximum likelihood estimation of $\lambda^{-1}/C = r^{-\gamma}$ as function of r.

With $C = 10^{-6}$ and $\gamma = 2$ we expect at distance $r = 100$ meter received signal strength $P = 10^{-10}$ mW $= -100$ dBm, which is near the lower limit of what the equipment can measure. So the range of the active tag is approximately 100 meter, which is satisfied by experience.

So the results of these measurements confirm the theoretical analysis we did before. However, it would be imprudent to decide that these results hold for all circumstances. For example, see the Figure 8 for signal strength measurements in a different room in the same building. Here we did measurements in the same way as in Figure 7, but using 9 spots and 4 tags together in order to obtain more measurements in less time. Here we also measured signal strength for multiple tag orientations per location.

The decrease in the received signal over distance is now less than we would expect, and we find $\gamma = 1.328$ by a maximum likelihood estimate. This was surprising, until we realized that we made a mistake in our measurement setup. The spots were installed very close to some metal cable tray, which apperently conducted the electromagnetic signals to the receiving spots at the expense of less loss than free-space path loss.

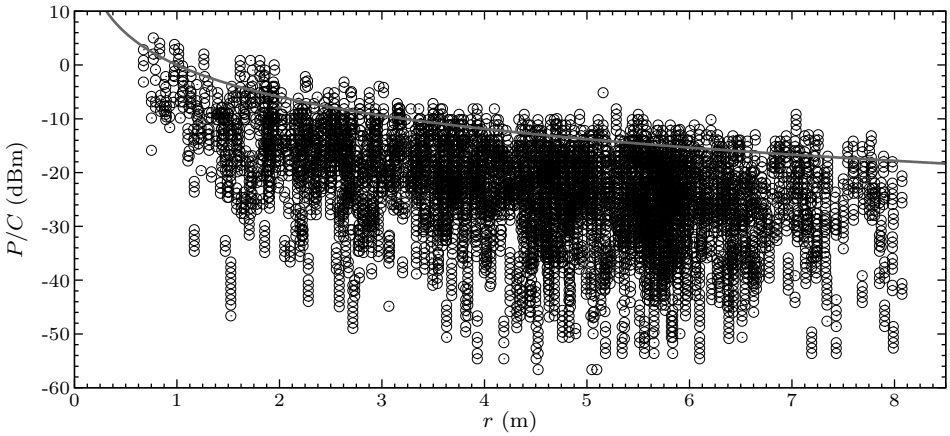


Figure 7: The same as Figure 4, but now the received signal strengths are rescaled by the estimated values of C_j. We also draw the graph of $r^{-\gamma}$ (in dBm), where $\gamma = 1.972$ is obtained by the method of maximum likelihood.

It is noteworthy that the measurements in Figure 8 still display a exponential distribution, as is shown in Figure 9. This confirms our hypothesis that multipath fading dominates the local relation between signal strength and disturbances, although here the global relationship betweeen signal strength and distance is distorted.

Passive tags

To estimate the relation between signal strength and distance for passive tags, we have done measurements where a spot was placed in the ceiling with the antenna directed downwards, and the tag was placed straight downwards in the space below the spot, at various distances. The results are in Figure 10.

The measurements took place for distances r between 0.5 and 2.0 meter. We chose $r \geq 0.5$ because the equipment uses radio waves of approximately 0.33 meter, and so $r = 0.5$ is near the border of the far field (condition 1 of the Friis transmission equation (5)). We chose $r \leq 2.0$, because the spot was placed in the ceiling which height was approximately 2.5 meter. Because the measurements were all straight downwards, for $r > 2.0$ we would place the tag close to the floor, which would affect the measurements more and more intense.

Now we will try to find the relation between the signal strength and distance. As already explained at (56), we take $\lambda(r) = C^{-1}r^{\gamma}$, where ideally we would have $\gamma = 4$ for passive tags, but because of nonlinear effects in the tag electronics we could find a slightly different value. Now solving (58) for γ gives $\gamma = 3.09$. Accordingly, we find $C = 1.97 \cdot 10^{-5}$ with (57), which corresponds to -47.1 dBm.

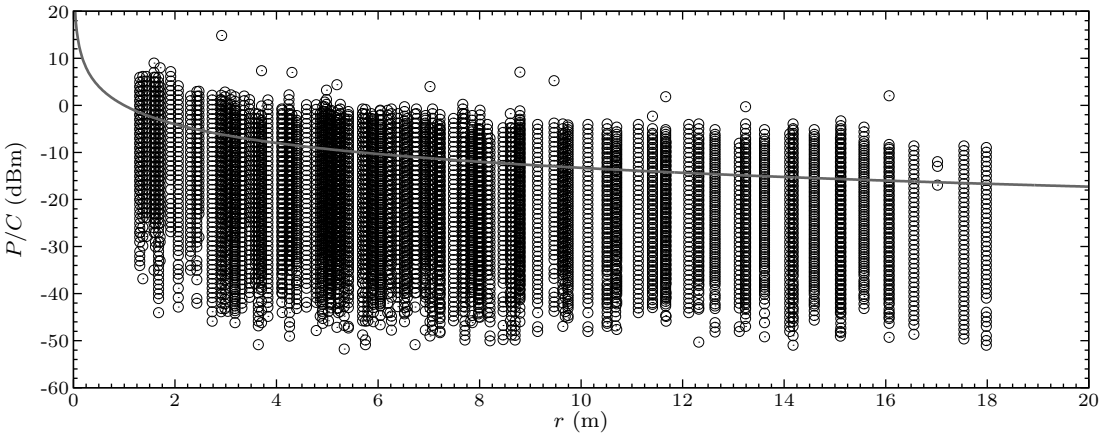


Figure 8: The same as Figure 7, but now with a different data set. The received signal strengths are rescaled by the estimated values of C_j. We also draw the graph of $r^{-\gamma}$ (in dBm), where $\gamma = 1.328$ is obtained by the method of maximum likelihood.

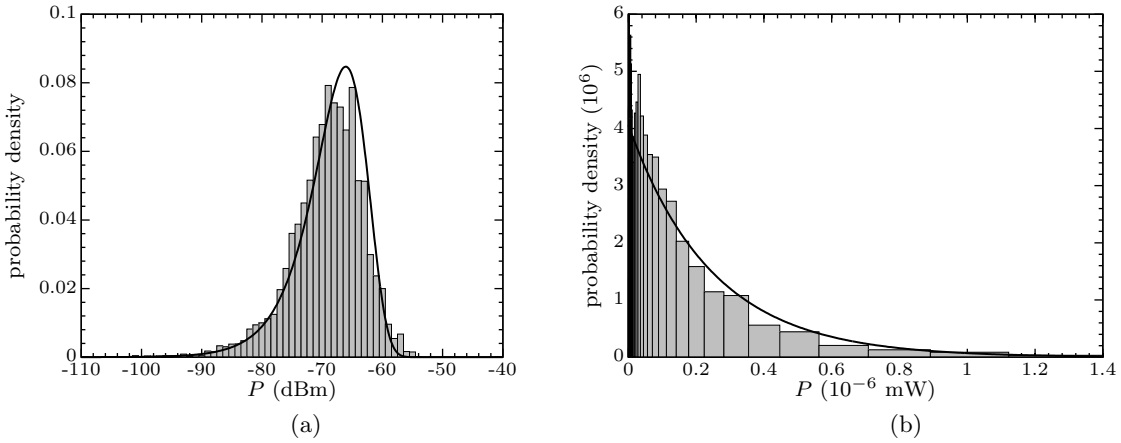


Figure 9: Histograms of 31672 measurements at distances between 5.5 and 6 meter. Left is the original histogram on dBm-scale, right is the histogram transformed to mW-scale. The curves are obtained by a maximum likelihood estimation of an exponential distribution.

This value of C is a bit different from the estimated λ^{-1} in Figure 6, because there we measured also in many other directions than straight downwards. Straight downwards the spot UHF antenna has the biggest gain.

The value of γ is a bit surprising. It means that the nonlinear effects in the tag electronics are really intense. However, these values seem to confirm the claim that the RFID tags can be detected at distances up approximately 10 meters. With these parameters C and γ, we find at $r = 11.7$ an expected received signal strength -80 dBm, which is approximately the lower limit of what the equipment is able to receive.

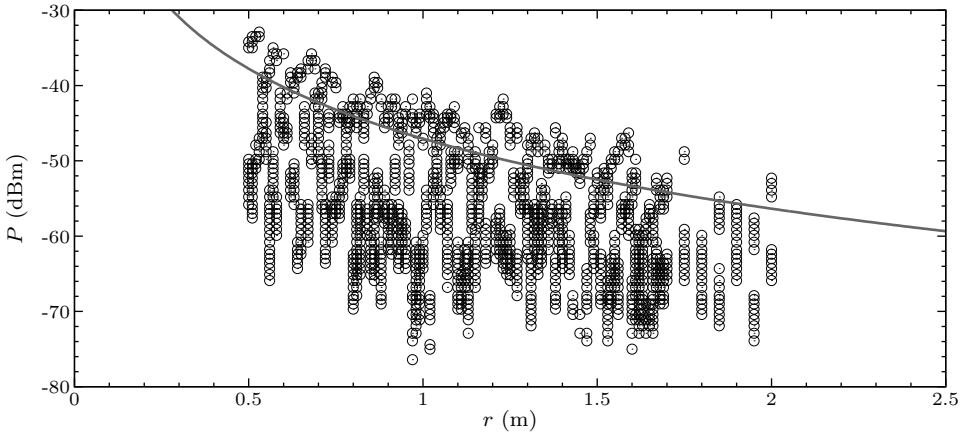


Figure 10: Signal strength measurements versus distance for passive UHF RFID tags. Because the measurements are rounded, the graphic consists many stacked circles. The gray curve is a fit of $\lambda^{-1} = Cr^{-\gamma}$ (in dBm) obtained by the method of maximum likelihood.

5.4 The signal model

From the measurements we evaluated in Section 5.2 we conclude that the signal strength received from a tag at fixed distance can be modeled by a exponential distribution (52). In Section 5.3 we saw that the distribution parameter λ depends on distance r by $\lambda = C^{-1}r^{\gamma}$. Here C and γ represent the spot and tag characteristics. Of course, we should realize that this relation between λ and r applies only if the space between tag and receiver is unobstructed, as it was in our measurements. We can summarize these results in one model for the received signal strength as follows. The probability to get received signal strength p at distance r is proportional to

$$f_\lambda(p) = \lambda e^{-\lambda p} = C^{-1}r^{\gamma}e^{-C^{-1}r^{\gamma}p}. \tag{59}$$

In Section 5.3 we already applied this model to estimate C and γ. But once we know C and γ, we can use measurements to estimate r. This will be worked out in detail in Section 6.1.

Vice versa, if r, C and γ are known, we can simulate the received signal strength, and use these simulations in tests. We only need to generate random numbers according to the exponential distribution with parameter $\lambda = C^{-1}r^{\gamma}$. According to Theorem 2, a computer pseudo-number generator can do this as follows. Let $F_\lambda(x)$ be the cumulative distribution function of the exponential probability distribution as in (54), and let U be a standard uniform $[0, 1]$-distributed variable, simulated by the pseudo-number generator. Then $F^{-1}(U) = -\lambda^{-1}\log(1-U)$ is distributed according to the exponential probability distribution with parameter λ. Because $1 - U$ is like U uniform $[0, 1]$-distributed, $-\lambda^{-1}\log(U)$ also has the exponential probability distribution.

So in terms of C, r and γ, we can simulate the signal strength P by

$$P = -Cr^{-\gamma} \log U. \tag{60}$$

If we would like to simulate the Intellifi equipment precisely, we transform P to dBm-scale, and round the resulting dBm's. We also ignore values below the lower limit of receivable signal strengths.

Finally, we have to realize that in static circumstances, it is impossible to get multiple independent measurements. Only by modifying the circumstances, it is possible to get independent samples from the same probability distribution. In practice, this means that given some static environment, the number independent signal strength measurements is restricted, because there is only a limited number of modifications possible.

Examples of modifications of the circumstances are as follows. Firstly, we can switch between multiple transmitting or receiving antennas, which have slightly different locations and possibly slightly different gain characteristics. The antenna locations should be not too far apart, in order to keep the distance between the transmitter and the receiver approximately equal. Another way to modify the circumstances is obtained by switching between distinct polarizations of the radio waves, or by changing the receiving or transmitting antenna orientation. It is also possible to change the wavelength of the radio waves, and thus choosing a different frequency in the radio communication. In multipath environments, different wavelengths lead to different multipath fading patterns and so to different received signal strengths.

All circumstance modifications should be big enough, in order to get really independent measurements. Concerning the Intellifi equipment we used, communication between the active Bluetooth tags and receivers took place at three distinct radio frequencies. However, the receivers used only one antenna in a fixed orientation. So in natural situations we can obtain three more or less independent signal strength measurements for active Bluetooth tags. Regarding the passive UHF RFID tags, the spots switch between two distinct antenna orientations in order to have radio waves of two distinct polarizations. Here we have only two more or less independent signal strength measurements. Intellifi has also spots which have two or three distinct antennas which all can be used in two distinct orientations, but these spots were not used in this research. However, these spots would enable us to obtain four or six more or less independent signal strength measurements.

So when we simulate the received signal strength, we have to keep in mind that the number of independent signal strength measurements is restricted. In Chapter 6, where we will do tests with simulated signals, we will also investigate the dependence of the localization accuracy on the number of independent signal strength measurements.

6 Solving the localization problem

The purpose of this chapter is to derive and test a practical localization procedure by a maximum likelihood estimation, such as described in Section 3.5. In previous chapter we discovered that the received signal strength can be modeled by an exponential probability distribution (52). In Section 5.4 we used the model to predict and simulate the received signal strength, given some distance. But by the maximum likelihood estimation, we can also do the reverse, and estimate the distance between tag and reader, given some received signal strength. This idea is worked out in Section 6.1.

In Section 6.2 we will push this idea further, and combine signal strength measurements from multiple spots in order to localize the tag. Unfortunately, here arise a couple of difficulties, because this problem can not be solved analytically. Moreover, the loglikelihood function is not smooth, and can contain many distinct local optima. Therefore, in Section 6.3 we will try to obtain a smoother problem, which solves the localization problem approximately. In this way, we will derive a general localization algorithm, which we will present in Section 6.4.

In Section 6.5 we will test the localization algorithm in many simulated situations, in order to investigate the localization accuracy with respect to physical variables. Thereafter, we will finally evaluate the localization algorithm to real world measurements, and inspect whether the results agree with the tests on simulated situations.

6.1 Distance estimation

This section discusses estimation of the distance between tag and reader by signal strength measurements. We will investigate how to estimate the distance, and the accuracy of the estimate. Therefore, we will compute the maximum likelihood estimator, and derive the probability density of that estimator. Then the accuracy can be found by computing the standard deviation.

Assume that we have obtained signal measurements p_k for $k = 0, 1, \ldots, n-1$. According to (59), we can find distance r by maximizing the likelihood

$$\mathrm{lik}(r) = \prod_{k=0}^{n-1} C^{-1} r^{\gamma} \exp\big(-C^{-1} r^{\gamma} p_k\big)$$

with respect to r. Note that we can omit the front factor C^{-1}, because it does not affect r. Therefore we take

$$\mathrm{lik}(r) = \prod_{k=0}^{n-1} r^{\gamma} \exp\big(-C^{-1} r^{\gamma} p_k\big).$$

Maximizing the likelihood function is equivalent to maximizing the loglikelihood $\mathcal{L}$, which is

$$\mathcal{L}(r) = n\gamma \log(r) - C^{-1} r^{\gamma} \sum_{k=0}^{n-1} p_k = n(\gamma \log(r) - P r^{\gamma}), \tag{61}$$

where

$$P \equiv \frac{1}{Cn} \sum_{k=0}^{n-1} p_k.$$

In order to find the maximum, we should solve $\mathcal{L}'(r) = 0$. Differentiation with respect to r gives

$$\mathcal{L}'(r) = n(\gamma r^{-1} - \gamma P r^{\gamma-1}) = n\gamma r^{-1}(1 - Pr^{\gamma}).$$

From this follows simply $1 - Pr^{\gamma} = 0$, and so

$$P^{-\frac{1}{\gamma}} = \left(\frac{1}{Cn} \sum_{k=0}^{n-1} p_k\right)^{-\frac{1}{\gamma}} = r. \tag{62}$$

So we can straight-forward estimate distance r. However, how reliable is such estimation, and is it unbiased? The reliability depends of course on n, the number of measurements. But to get things precise, we compute the probability distribution of the estimate, and compute its expected value and variance.

According to the model, p_k is distributed according to the exponential distribution with parameter $C^{-1}r^{\gamma}$. This is equal to the Gamma$(1, C^{-1}r^{\gamma})$ distribution. From Section 3.7 we know that the sum of n measurements has the Gamma$(n, C^{-1}r^{\gamma})$ distribution. The mean has distribution Gamma$(n, nC^{-1}r^{\gamma})$, and so we find that P is distributed according to Gamma(n, nr^{γ}). So for $x \geq 0$, we have

$$P = \frac{1}{Cn} \sum_{k=0}^{n-1} p_k \quad \sim \quad \frac{1}{\Gamma(n)} n^n r^{\gamma n} x^{n-1} \exp\bigl(-nr^{\gamma}x\bigr), \qquad (x \geq 0)$$

where $\sim$ means "is distributed according to probability density function". Using Theorem 1 we find the probability distribution of $P^{-1/\gamma}$. We take $T(x) = x^{-1/\gamma}$, and so $T^{-1}(x) = x^{-\gamma}$, and $\frac{d}{dx}T^{-1}(x) = -\gamma x^{-\gamma-1}$. Hence,

$$P^{-\frac{1}{\gamma}} = \left(\frac{1}{Cn} \sum_{k=0}^{n_j-1} p_k\right)^{-1/\gamma} \quad \sim \quad \gamma \frac{n^n}{\Gamma(n)} r^{\gamma n} x^{-\gamma n-1} \exp\bigl(-nr^{\gamma}x^{-\gamma}\bigr)$$

Now define for $x \geq 0$, the probability density function

$$\tilde{f}_r(x) \equiv \gamma \frac{n^n}{\Gamma(n)} r^{\gamma n} x^{-\gamma n-1} \exp(-nr^{\gamma}x^{-\gamma}). \tag{63}$$

With the help of symbolic mathematical computation program Mathematica, we find expected value

$$\tilde{\mu}(r) \equiv \int_0^{\infty} x\tilde{f}(x)\,\mathrm{d}x = \frac{\Gamma(n - \frac{1}{\gamma})}{\Gamma(n)} n^{1/\gamma} r. \tag{64}$$

Unfortunately, this is not equal to r, so $P^{-1/\gamma}$ is a *biased* estimate of distance r. Especially for low n, the distance tends to be overestimated, as is shown in Figure 11. The variance of the estimator

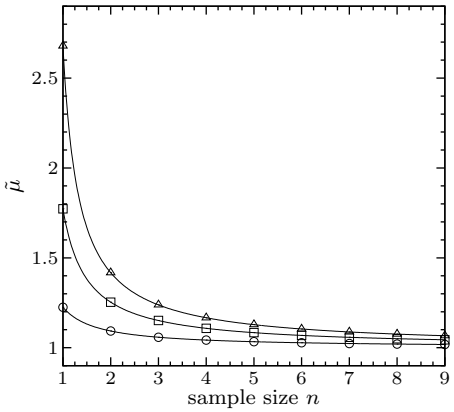


Figure 11: This plot shows the expected value $\tilde{\mu}$ as derived in (64) as function of n for $r = 1$ and different values of γ. Triangles represent $\gamma = 1.5$, squares represent $\gamma = 2.0$, and circles represent $\gamma = 4.0$.

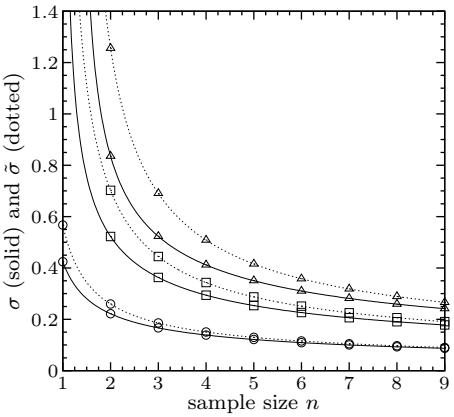


Figure 12: This plots shows standard deviations σ, $\tilde{\sigma}$, which are square root of variance σ^2 as in (68) (solid) and $\tilde{\sigma}^2$ as in (65) (dotted). Both are drawn as function of n for $r = 1$. The triangles, squares and circles represent the same values of γ as left.

is

$$\tilde{\sigma}^2(r) \equiv \int_0^\infty (x-\tilde{\mu})^2 \tilde{f}_r(x)\,\mathrm{d}x = \left(\frac{n^{\frac{2}{\gamma}}\Gamma(n-\frac{2}{\gamma}) - 2n^{\frac{1}{\gamma}}\Gamma(n-\frac{1}{\gamma})}{\Gamma(n)} + 1\right) r^2. \tag{65}$$

Fortunately, we can correct for the bias by multiplying $P^{-\frac{1}{\gamma}}$ by $n^{-1/\gamma}\Gamma(n)/\Gamma(n-\frac{1}{\gamma})$. In this way we get the *unbiased* estimator

$$\frac{\Gamma(n)}{n^{\frac{1}{\gamma}}\Gamma(n-\frac{1}{\gamma})} P^{-1/\gamma} = \frac{\Gamma(n)}{\Gamma(n-\frac{1}{\gamma})} \left(\frac{1}{C}\sum_{k=0}^{n-1} p_k\right)^{-1/\gamma}. \tag{66}$$

This estimator is subject to probability density function

$$f_r(x) \equiv \frac{n^{1/\gamma}\Gamma(n-\frac{1}{\gamma})}{\Gamma(n)}\, \tilde{f}_r\left(\frac{n^{1/\gamma}\Gamma(n-\frac{1}{\gamma})}{\Gamma(n)}\, x\right). \tag{67}$$

Calculations by Mathematica confirm that it has expected value $\mu(r) = r$. The variance $\sigma^2(r)$ of this estimator is, again calculated with the aid of Mathematica, (assuming $n \geq 2$ and $\gamma > 1$)

$$\sigma^2(r) \equiv \int_0^\infty (x-\mu)^2 f_r(x)\,\mathrm{d}x = \left(\frac{\Gamma(n)\Gamma(n-\frac{2}{\gamma})}{(\Gamma(n-\frac{1}{\gamma}))^2} - 1\right) r^2. \tag{68}$$

Figure 12 shows clearly that especially for small n and small γ, the unbiased estimator in (66) is better than the biased estimator in (62).

6.2 The general localization problem

Now we turn to the situation with multiple receivers. Again, we will use the principle of maximum likelihood. Let the locations of the receivers be $\mathbf{a}_0, \ldots, \mathbf{a}_{m-1}$ (commonly in 3-dimensional Euclidean space), and we would like to find the location of the tag. This location can be found by maximizing the likelihood

$$\mathrm{lik}(\mathbf{x}) = \prod_{j=0}^{m-1} \prod_{k=0}^{n_j-1} \frac{1}{C} \|\mathbf{x} - \mathbf{a}_j\|_2^\gamma \exp\Big(-\frac{1}{C}\|\mathbf{x} - \mathbf{a}_j\|_2^\gamma p_{j,k}\Big)$$

with respect to $\mathbf{x}$. The global maximizer of the likelihood whould ideally be the location of the tag. Note that the front factor $\frac{1}{C}$ can be omitted: it does not affect $\mathbf{x}$. So we take

$$\mathrm{lik}(\mathbf{x}) = \prod_{j=0}^{m-1} \prod_{k=0}^{n_j-1} \|\mathbf{x} - \mathbf{a}_j\|_2^\gamma \exp\Big(-\frac{1}{C}\|\mathbf{x} - \mathbf{a}_j\|_2^\gamma p_{j,k}\Big).$$

This is equivalent to maximizing the loglikelihood

$$\mathcal{L}(\mathbf{x}) = \sum_{j=0}^{m-1} n_j \Big(\frac{\gamma}{2}\log(\|\mathbf{x} - \mathbf{a}_j\|_2^2) - P_j\|\mathbf{x} - \mathbf{a}_j\|_2^\gamma\Big), \tag{69}$$

where

$$P_j \equiv \frac{1}{C_j n_j} \sum_{k=0}^{n_j-1} p_{j,k}.$$

In order to find the maximum, we should solve $\nabla\mathcal{L}(\mathbf{x}) = \mathbf{0}$. We see that differentiation with respect to $\mathbf{x}$ gives the gradient vector

$$\begin{aligned}\nabla\mathcal{L}(\mathbf{x}) &= \sum_{j=0}^{m-1} n_j \Big(\gamma\frac{\mathbf{x} - \mathbf{a}_j}{\|\mathbf{x} - \mathbf{a}_j\|_2^2} - \gamma P_j(\mathbf{x} - \mathbf{a}_j)\|\mathbf{x} - \mathbf{a}_j\|_2^{\gamma-2}\Big) \\ &= \gamma \sum_{j=0}^{m-1} n_j \frac{\mathbf{x} - \mathbf{a}_j}{\|\mathbf{x} - \mathbf{a}_j\|_2^2}\Big(1 - P_j\|\mathbf{x} - \mathbf{a}_j\|_2^\gamma\Big).\end{aligned} \tag{70}$$

There does not exist an analytic technique to solve $\nabla\mathcal{L}(\mathbf{x}) = \mathbf{0}$ in general. Therefore, we have to use numerical methods. In Chapter 4 we discussed the most important numerical methods.

Unfortunately, numerical methods can only find local optima, and the loglikelihood function (69) may have many local optima. Moreover, the singularities at $\mathbf{x} = \mathbf{a}_j$ create an unsmooth behavior. So it is very important to take an appropriate initial guess. We will obtain this guess by first solving some "smoother" relaxed problems. This is the subject of next section. We have also to realize that $\mathcal{L}$ probably has no unique global maximizer. In this case, we have to be satisfied with the choice of the numerical method we use.

Another problem is the fact that that maximizing $\mathcal{L}$ does not give an unbiased estimate, which we for distance estimation in Section 6.1. There we also saw that we could correct this by replacing

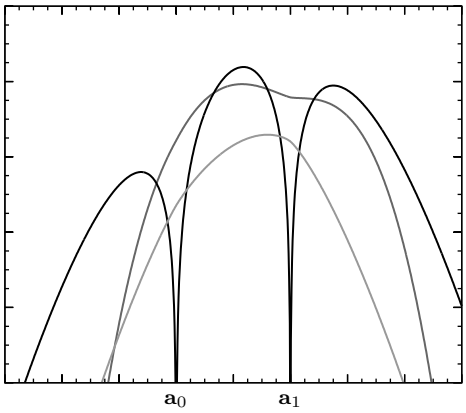


Figure 13: Some typical 1-dimensional maximum likelihood estimation problem. Plotted in black is $\mathcal{L}$ as function of x with $m = 2$, and some chosen values of other parameters. There are multiple local optima, and singularities at $\mathbf{a}_0$ and $\mathbf{a}_1$, such that it is not easy to find efficiently the global optimum by a computer algorithm. Plotted in darker gray is $-L$, which is clearly smoother and easier to optimize. Plotted in lighter gray is $\mathcal{L}_0$, which shows that its optimum is a proper initial guess for optimizing $-L$. Vertical scalings are modified to simplify comparison.

P_j by

$$S_j \equiv \left(\frac{\Gamma(n_j - \frac{1}{\gamma})}{\Gamma(n_j)}\right)^{\gamma} \frac{1}{C} \sum_{k=0}^{n_j-1} p_{j,k}. \tag{71}$$

However, it is not clear whether this is a real improvement while doing localization.

6.3 Problem relaxations

In this section we will derive two problem relaxations, which are easier to solve by numerical means. The loglikelihood function (69) can be split in two parts,

$$\mathcal{L}_0(\mathbf{x}) = -\sum_{j=0}^{m-1} n_j P_j \|\mathbf{x} - \mathbf{a}_j\|_2^{\gamma}, \tag{72}$$

and

$$\mathcal{L}_1(\mathbf{x}) = \sum_{j=0}^{m-1} \frac{1}{2} \gamma n_j \log\left(\|\mathbf{x} - \mathbf{a}_j\|_2^2\right), \tag{73}$$

such that $\mathcal{L}(\mathbf{x}) = \mathcal{L}_0(\mathbf{x}) + \mathcal{L}_1(\mathbf{x})$. Under assumption $\gamma > 1$, we see that if $\|\mathbf{x}\|_2 \to \infty$, then we observe that $\|\nabla\mathcal{L}_0(\mathbf{x})\|_2 \to \infty$ while $\|\nabla\mathcal{L}_1(\mathbf{x})\|_2 \to 0$. So $\mathcal{L}_0$ describes more or less the *global* behavior of $\mathcal{L}$, while $\mathcal{L}_1$ describes more or less the *local* behavior of $\mathcal{L}$. Moreover, for $\gamma \geq 1$ is $-\mathcal{L}_0$ a convex function, and for $\gamma > 1$ is $-\mathcal{L}_0$ everywhere differentiable. So $\mathcal{L}_0$ can easily be optimized. For $\gamma = 2$, we can solve $\nabla\mathcal{L}_0(\mathbf{x}) = \mathbf{0}$ even analytically by

$$\mathbf{x} = \frac{\sum_{j=0}^{m-1} n_j P_j \mathbf{a}_j}{\sum_{j=0}^{m-1} n_j P_j}. \tag{74}$$

We will use this as initial guess for cases where $\gamma \neq 2$.

In general, it is not wisely to use this point as initial guess to optimize $\mathcal{L}$, especially when the global optimum can clearly not be written as a weighted mean of the spot locations. Therefore, we first consider a similar optimization problem where we relaxed an assumption.

From the analysis in Section 6.1 we know that P_j is distributed according to Gamma$(n_j, n_j r_j^\gamma)$ distribution. So P_j has expected value $\mu_j = r_j^{-\gamma}$ and variance $\sigma_j^2 = (n_j r_j^{2\gamma})^{-1}$. Now, by Theorem 3, the central limit theorem, the variable

$$Z_j \equiv \sqrt{n_j}\frac{P_j - \mu_j}{\sigma_j} = \sqrt{n_j}\frac{P_j - r_j^{-\gamma}}{(\sqrt{n_j} r_j^\gamma)^{-1}} = n_j(P_j r_j^\gamma - 1)$$

has standard normal distribution if $n_j \to \infty$. In Section 3.6 we saw that the maximum likelihood estimation for normal-distributed variables is equivalent to a least-squares fit. Therefore, minimizing $\sum_{j=0}^{m-1} Z_j^2$ solves the localization problem if $n_j \to \infty$ for $j = 0, \ldots, m-1$. This is

$$\sum_{j=0}^{m-1} n_j^2(P_j r_j^\gamma - 1)^2 = \sum_{j=0}^{m-1} n_j^2(P_j \|\mathbf{x} - \mathbf{a}_j\|_2^\gamma - 1)^2.$$

We call this function $L(\mathbf{x})$,

$$L(\mathbf{x}) \equiv \sum_{j=0}^{m-1} n_j^2(P_j \|\mathbf{x} - \mathbf{a}_j\|_2^\gamma - 1)^2. \tag{75}$$

Now we hope that optimizing L optimizes $\mathcal{L}$ approximately if $n_j \not\to \infty$, and even if n_j is rather small. The gradient of $L(\mathbf{x})$ is

$$\nabla L(\mathbf{x}) = \gamma \sum_{j=0}^{m-1} n_j^2 P_j(P_j \|\mathbf{x} - \mathbf{a}_j\|_2^\gamma - 1)\|\mathbf{x} - \mathbf{a}_j\|_2^{\gamma-2}(\mathbf{x} - \mathbf{a}_j).$$

For solving $\nabla L(\mathbf{x}) = \mathbf{0}$ we need also numerical techniques, but this problem does not have singularities (again we assume $\gamma > 1$) which are present in $\nabla\mathcal{L}$. Unfortunately, the function L is not necessarily convex, so it can have multiple local optima. However, in practice the solution of $\nabla\mathcal{L}_0(\mathbf{x}) = \mathbf{0}$ is a quite good initial guess for solving $\nabla L(\mathbf{x}) = \mathbf{0}$.

In Figure 13 is shown a comparison of $\mathcal{L}$, $\mathcal{L}_0$ and L. This illustrates nicely the analysis in this section.

6.4 A general localization algorithm

In this section we discuss a general localization algorithm, using the analysis in previous sections. As described in Section 6.2, localization is equivalent to optimizing the loglikelihood function $\mathcal{L}$ in (69). Optimization can be done by a numerical method from Chapter 4, but then we need a proper initial guess. Such a proper guess can be obtained by first optimizing $\mathcal{L}_0$ in (72) and L in

(75), where we use (74) as initial guess for optimizing $\mathcal{L}_0$, and the resulting point of this problem as initial guess for optimizing L.

This is the basis structure of our localization algorithm, but it has two important disadvantages. The first one is due to the fact that L is not necessarily convex. Hence, in practice we do not always find the global optimum of L. The second disadvantage comes from the fact that the optimum of $\mathcal{L}_0$ is always a convex combination of the spot locations. Therefore the algorithm performs poorly if the real location of the tag which has to be localized is no convex combination of the spot locations.

A remedy for these disadvantages is to optimize L two times, but at the second time with a different starting point. So we take the first starting point to be equal to the solution of $\nabla\mathcal{L}_0(\mathbf{x}) = \mathbf{0}$, which we call $\mathbf{x}_0$. In this way we find a local optimum of L at $\mathbf{x}_1$. The second starting point we take at $2\mathbf{x}_0 - \mathbf{x}_1$, which is opposite of $\mathbf{x}_1$ with respect to $\mathbf{x}_0$. If we find $\nabla L(\mathbf{x})$ to be very close to $\mathbf{0}$ at an initial guess, we add an small random vector, to make sure we do not get stuck at a bad local maximum. If the two initial guesses for L led to two distinct optima of L, we take the best in terms of $\mathcal{L}$ as initial guess for optimizing $\mathcal{L}$.

The disadvantage of this remedy is that we make the localization accuracy worse for situations where the real location of the tag to be localized is a convex combination of the spot locations.

There are probably also other refinements for this algorithm possible, but we will only adapt the choice in (71) while optimizing $\mathcal{L}$. Perhaps some future research can look for some improvements. While optimizing L or $\mathcal{L}$, computing the Hessian matrices seems to be unattractive because of the mixed terms. Therefore, we use the BFGS quasi-Newton method as optimization method. This method was discussed in Section 4.4.3, and is paired up with inexact line search, discussed in Section 4.5. Let us abbreviate this as BFGSW$(f, \mathbf{x})$, where f is the function to be minimized, and $\mathbf{x}$ is the initial guess. Then our localization algorithm is schematically lined out in Algorithm 4. We implement it in MATLAB, the code is in Appendix A.

6.5 Localization tests

In this section we will de some localization tests. We are interested in the dependence of the localization accuracy on the physical parameters in the localization problem. Therefore we will do many tests and track the localization accuracy. In order to check our model assumptions, we will also compare results by simulated signals and real world measurements.

We determine the localization accuracy by the mean error. Given some estimated tag location $\mathbf{x}_\mathrm{e}$ and the real tag location $\mathbf{x}_\mathrm{r}$, we measure the error by

$$E \equiv \|\mathbf{x}_\mathrm{e} - \mathbf{x}_\mathrm{r}\|_2.$$

Algorithm 4 Algorithm to solve the localization problem

Compute $P_j = \frac{1}{C_j n_j} \sum_{k=0}^{n_j-1} p_{j,k}$ for $j = 0, \ldots, m-1$
$\mathbf{x}_0 \leftarrow \big(\sum_{j=0}^{m-1} n_j P_j \mathbf{a}_j\big) / \big(\sum_{j=0}^{m-1} n_j P_j\big)$
if $\gamma \neq 2$ **then**
    $\mathbf{x}_0 \leftarrow \text{BFGSW}(-\mathcal{L}_0, \mathbf{x})$
$\mathbf{x}_1 \leftarrow \text{BFGSW}(L, \mathbf{x}_0)$
$\mathbf{x}_2 \leftarrow \text{BFGSW}(L, 2\mathbf{x}_0 - \mathbf{x}_1)$
$P_j \leftarrow P_j \cdot n_j(\Gamma(n_j - 1/\gamma)/\Gamma(n_j))^\gamma$ for $j = 0, \ldots, m-1$
if $\mathcal{L}(\mathbf{x}_2) > \mathcal{L}(\mathbf{x}_1)$ **then**
    $\mathbf{x}_1 \leftarrow \mathbf{x}_2$
$\mathbf{x} \leftarrow \text{BFGSW}(-\mathcal{L}, \mathbf{x}_1)$
return $\mathbf{x}$

If we do N tests, where in test k we find E_k, the mean error is

$$\overline{E} \equiv \frac{1}{N} \sum_{k=0}^{N-1} E_k.$$

If N is large, we hope that this converges

It is also possible to take a different measure for the localization accuracy, for example the median of the computed errors, or a percentile score, but we will simply stay at the mean error.

In our simulations, we assume that the receivers are are arranged on some equally spaced grid. For tests in two dimensional space, this means that the spot locations are on some square grid, and in three dimensional space we have a cubical grid. We will perform tests which show the dependence of the mean error on the position of the tag in the grid, the grid spacing, and the number of measurements per spot. We will do tests for different numbers of dimensions of the localization problem, and for $\gamma = 2.0$ (active tags) and $\gamma = 3.09$ (passive tags).

Figure 14 shows the dependence of the mean error on the location of the tag in the grid. We see that we can expect smaller errors near spots than further away. This suits with the observation in Section 6.1 that the standard deviation of the distance estimation increases with distance.

In following tests, we will take the tag at the middle point between the spots. Figure 14 shows that this point is fairly representative.

In order to test the dependence of the localization accuracy on the number of measurements n per spot, we take the same grid spacings as in Figure 14, and vary n. The results of tests in 1-D, 2-D, and 3-D are in Figure 15. From these graphs follow approximately $\overline{E} \sim n^{-1/2}$.

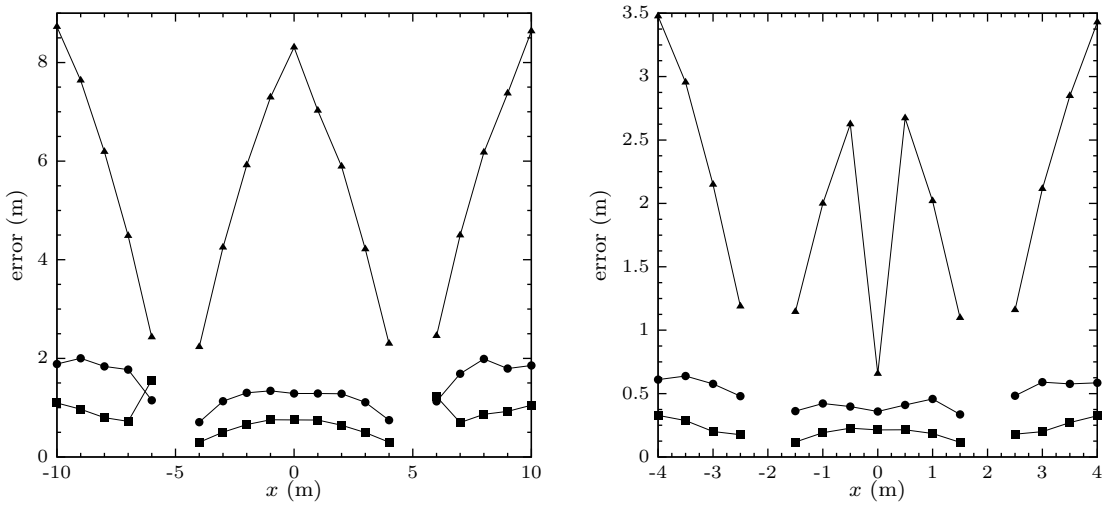


Figure 14: Dependence of localization error on the position of tag in grid, based on simulated received signal strength. The grid is here 1-dimensional, and there are only two spots. The plot on the left is from active tags, the plot on the right is for passive tags. The graphs with circles are the mean error $\overline{E}$, the squares are the median of the errors, and the triangels are the 95th percentile of the errors. At the left, the spots were located at $x = -5$ and $x = 5$, on the right the spots were located at $x = -2$ and $x = 2$.

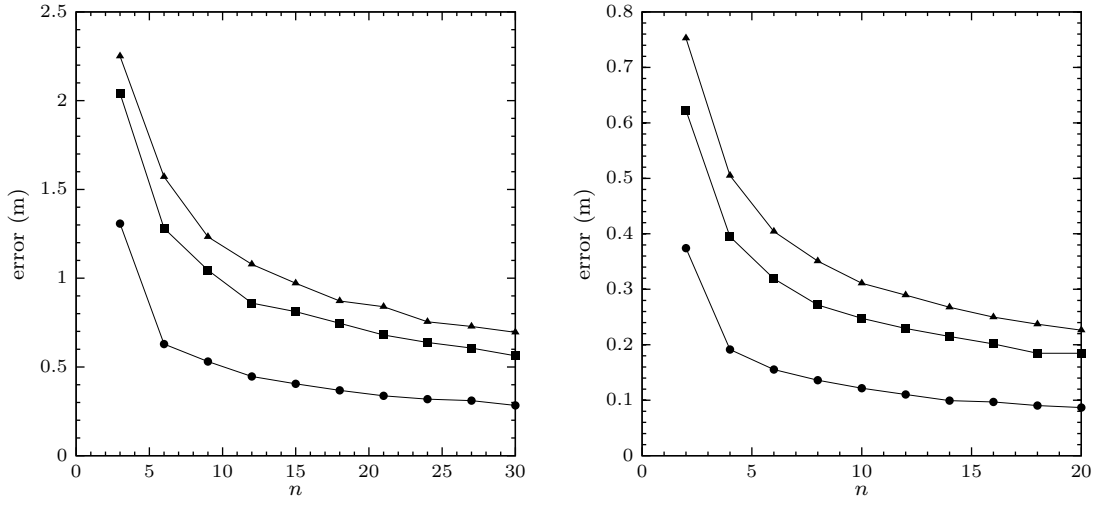


Figure 15: Dependence of mean localization error $\overline{E}$ on n, the number of measurements per spot, based on simulated received signal strength. On the left for active tags with grid spacing of 10 meter, and on the right for passive tags, with grid spacing 4 meter. The circles represent one-dimensional simulations, the squares represent two-dimensional simulations, and the triangles represent three-dimensional simulations. From these graphs follow approximately $\overline{E} \sim n^{-1/2}$.

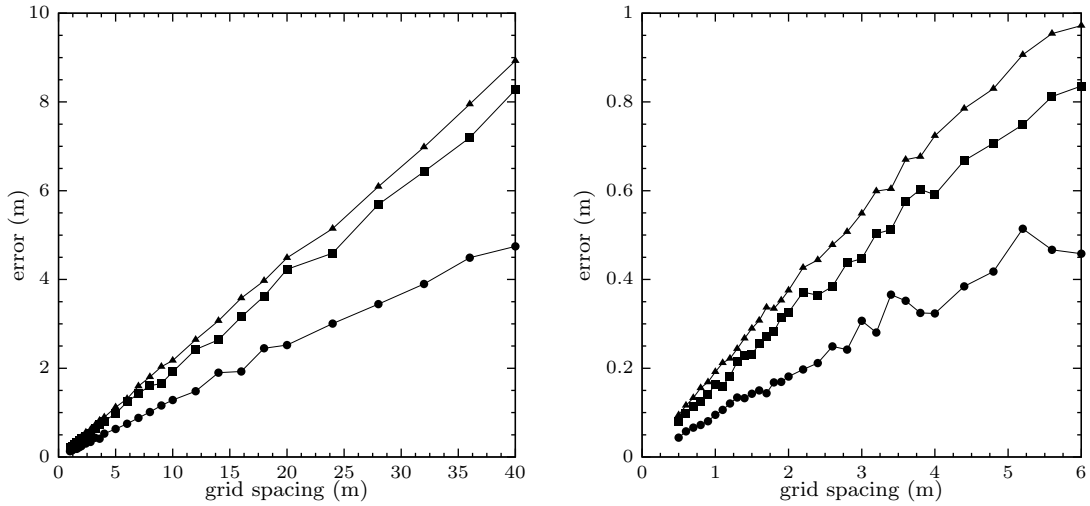


Figure 16: Dependence of mean localization error $\overline{E}$ on the grid spacing for simulated measurements. On the left for active tags with $n = 3$ measurements per spot. On the right for passive tags with $n = 2$ measurements per spot. The circles represent one-dimensional simulations, the squares represent two-dimensional simulations, and the triangles represent three-dimensional simulations. From these graphs follow a linear relationship.

Next, we test the dependence of the localization accuracy on the number grid spacing. The results of tests in 1-D, 2-D, and 3-D are in Figure 16. From these graphs follow approximately linear relationship between the grid spacing and the mean error.

The question is now whether these results are an improvement of zone detection. The graphs suggest it is an improvement, especially if we work in one dimensional space. However, the improvement is only slightly. For example, the mean error for active tags in 2-D or 3-D space is approximately equal to the grid spacing devided by 5. But that means that we can roughly distinguish $5/2 = 2.5$ zones between two spots. (We need to divide by 2, because the error can be positive or negative). For passive tags in 2-D or 3-D space we have in this way approximately 3 zones between 2 spots. However, for real measurements, the results are likely a bit worse.

In order to test the reliability of our results, we simulated the setup of the measurements in Figure 4. There we had 4 spots on a two-dimensional grid with grid spacing 4.8, and the tags were in 3-D space. With simulated received signal strengths, we found mean error $\overline{E} = 2.0$. However, with real measurements we found $\overline{E} = 2.2$. This shows that our simulations are a bit too optimistic, altough they represent the real-world pretty well.

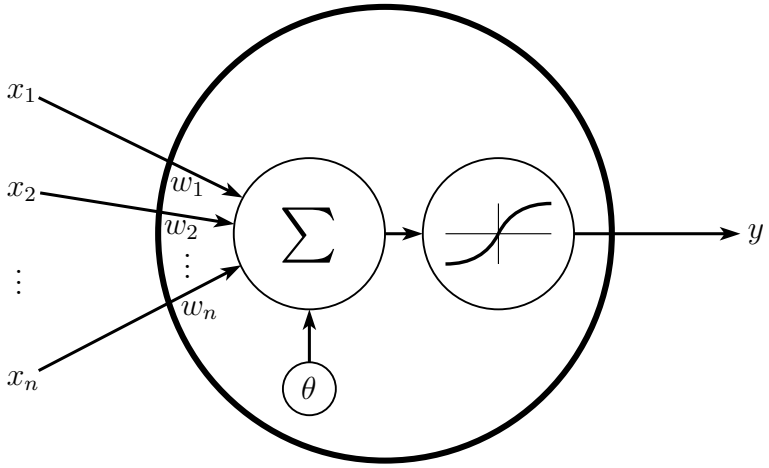


Figure 17: A schematic overview of a network unit. At the left are the inputs $x_1, \ldots, x_n$, which are weighted by $w_1, \ldots, w_n$. The bias θ, and activation function ϕ are displayed as well.

7 Localization by artificial neural networks

The localization problem can also be solved by neural networks. In this chapter, we discuss neural networks in general, we discuss the localization results by neural networks, and we compare them to the results in Section 6.5. The information about neural networks in this chapter is mainly from [16].

Neural network are inspired by the human brain, which can do highly complex cognitive tasks like visual and auditory pattern recognition. However, most current neural network models do not try to imitate the biological brain closely. They should rather be regarded as a class of interpolation algorithms.

7.1 Basic definitions

The basic building blocks of neural networks are 'neurons', which we will call *units*. These units are in essence simple functions of some special form, see Figure 17. They have an input vector $\mathbf{x} \equiv (x_1, x_2, \ldots, x_n)^T$, and scalar output y. The inputs are weighted by $\mathbf{w} \equiv (w_1, \ldots, w_n)^T$.

The unit generate its output as follows. The weighted inputs are summed, and a *bias* θ is added. Thereafter, *activation function* ϕ protects the unit of being 'over-activated'. In this way we get

$$y = \phi(\mathbf{w}^T\mathbf{x} + \theta). \tag{76}$$

Activation functions are typical increasing functions with range $[0, 1]$ or $[-1, 1]$, and often feature some symmetry. Some popular choices are the sigmoid $\phi(z) = 1/(1 + e^{-z})$, and the hyperbolic tangent $\phi(z) = \tanh(z)$, which are neat smooth functions, see Figure 18. Note that $1/(1 + e^{-z}) = \frac{1}{2}(\tanh(\frac{1}{2}z) + 1)$. It is also possible to use a clipping or a threshold function as activation function,

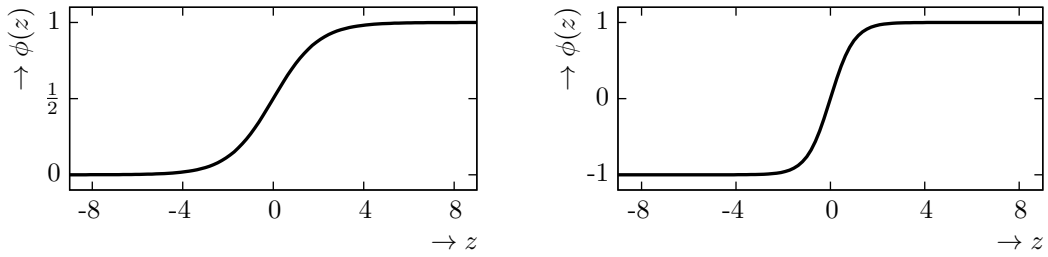


Figure 18: Plots of activation functions $\phi(z) = 1/(1 + e^{-z})$ (left) and $\phi(z) = \tanh(z)$ (right).

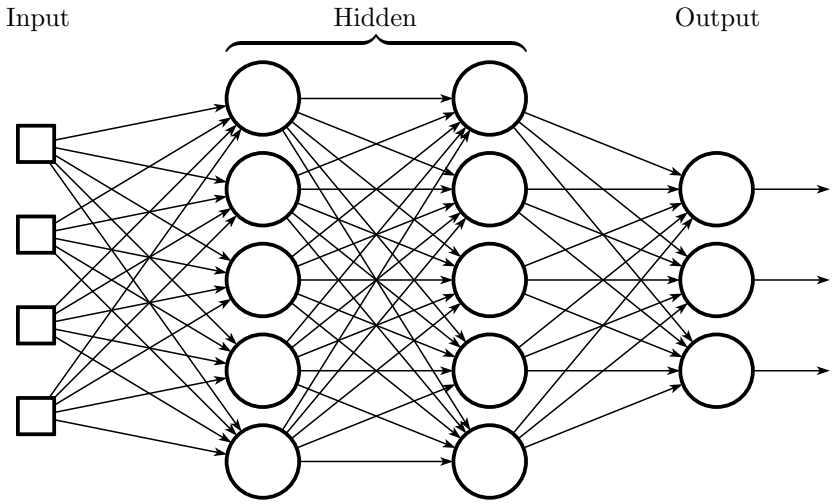


Figure 19: A feed-forward network with two hidden layers.

but they are less attractive because they are not everywhere differentiable. The purpose of bias θ in the activation function is that it represents the horizontal displacement of the activation function.

The units are connected to each others to form a neural network. There are several ways of organizing a neural network, but the most successful networks appears to be the multilayer networks with only forward directed links, like drawn in Figure 19.

Furthermore, there is an input layer, with input units or *input sites*. These are no ordinary units, because they do no computation at all. Here we use simply the problem input as output of this layer. We have drawn these units as squares instead of circles.

As designated, the outermost layers are used for input and output. Due to the behavior of the activation functions, it is often necessary to preprocess input and post-process output. The inner layers are called 'hidden' layers, because of the background of neural networks as black box solvers.

Let us now introduce some notation. We number the layers from 0 to l, where there are $l + 1$

layers. Layer 0 is the input layer, and layer l is the output layer. Next we define for layers $1, \ldots, l$,

$$\begin{array}{lll} x_{i,k} & \equiv y_{i-1,k} & \text{input } k \text{ in layer } i, \\ w_{i,j,k} & & \text{weight of input } k \text{ in unit } j \text{ in layer } i, \\ \theta_{i,j} & & \text{bias of unit } j \text{ in layer } i, \\ s_{i,j} & \equiv \theta_{i,j} + \sum_k w_{i,j,k} x_{i,k} & \text{net input of unit } j \text{ in layer } i, \\ y_{i,j} & \equiv \phi(s_{i,j}) & \text{output of unit } j \text{ in layer } i, \\ y'_{i,j} & \equiv \phi'(s_{i,j}) & \text{derivative of the output of unit } j \text{ in layer } i. \end{array}$$

Here the derivative ϕ' is taken with respect to the net input. Using vector and matrix notation, we get clearer formulas. So we define for layers $1, \ldots, l$

$$\begin{array}{lll} \mathbf{x}_i & \equiv \mathbf{y}_{i-1} & \text{input of layer } i, \\ \mathbf{w}_{i,j} & \equiv (w_{i,j,1}, w_{i,j,2}, \ldots)^T & \text{vector of weights } w_{i,j,k} \text{ in unit } j \text{ in layer } i, \\ \boldsymbol{W}_i & \equiv (\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, \ldots)^T & \text{matrix of weights in layer } i, \\ \boldsymbol{\theta}_i & \equiv (\theta_{i,1}, \theta_{i,2}, \ldots)^T & \text{vector of biases } \theta_{i,j} \text{ in layer } i, \\ \mathbf{s}_i & \equiv \boldsymbol{\theta}_i + \mathbf{W}_i \mathbf{x}_i & \text{vector of net inputs in layer } i, \\ \mathbf{y}_i & \equiv \boldsymbol{\phi}(\mathbf{s}_i) & \text{output of layer } i. \\ \mathbf{y}'_i & \equiv \boldsymbol{\phi}'(\mathbf{s}_i) & \text{derivative of output of layer } i. \end{array}$$

Here, $\boldsymbol{\phi}$ is the vector-valued equivalent of activation function ϕ. So $\boldsymbol{\phi}(\mathbf{s}_i)$ means that ϕ is evaluated separately over all vector components of $\mathbf{s}_i$. In this way we can compute the output $\mathbf{y} = \mathbf{y}_l$ of the network for input $\mathbf{x} = \mathbf{x}_0$ recursively by

$$\begin{cases} \mathbf{y}_0 = \mathbf{x}_0 \\ \mathbf{y}_i = \boldsymbol{\phi}\big(\boldsymbol{W}_i \mathbf{y}_{i-1} + \boldsymbol{\theta}_i\big), \qquad i = 1, \ldots, l \end{cases} \tag{77}$$

In order to get neater formulas, we will consider the biases $\theta_{i,j}$ as ordinary weights which are attached to an input of fixed value 1. In this way, we extend the input vectors, weight vectors, and weight matrices as follows,

$$\overline{\mathbf{x}}_i \equiv \begin{pmatrix} 1 \\ \mathbf{x}_i \end{pmatrix}, \qquad \overline{\mathbf{w}}_{i,j} \equiv \begin{pmatrix} \theta_{i,j} \\ \mathbf{w}_{i,j} \end{pmatrix}, \qquad \overline{\boldsymbol{W}}_i \equiv (\boldsymbol{\theta}_i \ \boldsymbol{W}_i).$$

Then (77) becomes for some input $\mathbf{x} = \mathbf{x}_0$,

$$\begin{cases} \mathbf{y}_0 = \mathbf{x}_0, \\ \overline{\mathbf{x}}_i = (1, \mathbf{y}_{i-1}^T)^T, \qquad i = 1, \ldots, l, \\ \mathbf{y}_i = \boldsymbol{\phi}\big(\overline{\boldsymbol{W}}_i \overline{\mathbf{x}}_i\big), \qquad i = 1, \ldots, l. \end{cases} \tag{78}$$

In the sequel, we will sometimes omit the bar in $\overline{\mathbf{x}}_i$, but it will be clear from context what is meant.

7.2 Network learning

Given some so-called multilayer feed-forward network, the objective is to find the right weights in the units. In a common situation, a training set $\{(\mathbf{x}_j^t, \mathbf{y}_j^t)\}$, $j = 0, \ldots, N-1$, is provided to which the network is fitted. The superscript t is to distinguish these vectors from the layers input and output. Now let f be the underlying function of the dataset such that $\mathbf{y}_j^t = f(\mathbf{x}_j^t)$, and let F be the network function as defined in (78),

$$\mathbf{y} \equiv \mathbf{y}_l = F(\mathbf{x}) \equiv F(\mathbf{x}; \overline{\boldsymbol{W}}_1, \ldots, \overline{\boldsymbol{W}}_l).$$

Now define the residuals

$$E_j \equiv \frac{1}{2} \left\| F(\mathbf{x}_j^t; \overline{\boldsymbol{W}}_1, \ldots, \overline{\boldsymbol{W}}_l) - \mathbf{y}_j^t \right\|_2^2, \tag{79}$$

and total residue

$$E \equiv \sum_{j=0}^{N-1} E_j. \tag{80}$$

In order to get $F(\mathbf{x}_j^t) \approx \mathbf{y}_j^t = f(\mathbf{x}_j^t)$ for all j, we have to minimize E, that is to solve

$$\underset{W_1, \boldsymbol{\theta}_1, \ldots, W_l, \boldsymbol{\theta}_l}{\arg\min} \sum_{j=0}^{N-1} \frac{1}{2} \left\| F(\mathbf{x}_j^t; \overline{\boldsymbol{W}}_1, \ldots, \overline{\boldsymbol{W}}_l) - \mathbf{y}_j^t \right\|_2^2. \tag{81}$$

By minimizing this error, we hopefully get $F \approx f$ also outside the training set. In practice, it appears that strictly minimizing leads to over-fitting: the performance at the data points of the fit is very good, but the performance for unseen data is less. To avoid over-fitting, the available dataset is divided in three parts: the *training set*, the *validation set*, and the *test set*. For training, only data from the training set is used, but during the training process, both errors in the training set and errors in the validation set are tracked. In the beginning of the training process, both errors will typically decrease very fast. After a while, the error curves become more flat, and finally the error in the validation set will start to increase slowly. Then it is time to stop the training process to avoid over-fitting. Finally, the test set is used to check the performance of the fit to unseen data.

So by implementing a complex stopping criterion we are more sure about $F \approx f$. In this research we choose the sizes of the three subsets to be such that the training set holds 60 percent of the available data, and the validation and test sets each hold 20 percent of the available data.

The training itself can be done by ordinary iterative minimization algorithms. We can use minimization algorithms which use gradients, because the gradients can be computed relatively easily by the backpropagation algorithm, which is explained in Section 7.3. However, because the training set is commonly very large, it is very expensive to compute E and ∇E. But in general, many training patterns are very similar, so it is often unnecessary to compute E_j and ∇E_j for all $j = 0, \ldots, N-1$ to get an impression of E. Many training algorithms exploit this property.

A general simple training algorithm can be formed by considering the training set pattern by pattern, and update the unit weights a bit accordingly. The size of the updates is controlled by a parameter which is called the 'learning rate'. Doing a line search is in this case not very sensible.

It is also possible to take some patterns together and update after every batch of patterns. In order to avoid oscillation problems like in gradient descent, we can insert momentum terms.

7.3 The backpropagation algorithm

The backpropagation algorithm is a method to compute derivatives in a neural network. It works basically as follows. First, every unit except the input units, are divided into two parts. The right parts compute the ordinary output of the units, while the left parts compute the derivatives of the output values. Furthermore, both values are saved. So if we propagate an input vector through the network in the ordinary way, unit j in layer i will hold both $\mathbf{y}_{i,j}$ and $\mathbf{y}'_{i,j}$.

After this feed-forward step, the links in the network are reversed and there is the backpropagation step. Now there are 1's fed to the network at the output layer, and instead of the function evaluation of ϕ at every unit, there is a simple multiplication with the derivative $y'_{i,j}$. The result $\delta_{i,j}$ is saved at every unit. So we have $\delta_{i,0} = 1$, and for $j > 0$

$$\delta_{i,j} \equiv \begin{cases} y'_{i,j} & \text{if } i = l, \\ y'_{i,j} \sum_j w_{i+1,j,k} \delta_{i+1,j} & \text{if } i < l. \end{cases}$$

Now the derivatives of the network can be computed by

$$\frac{\partial F}{\partial w_{i,j,k}} = \delta_{i,j} x_{i,k} = \delta_{i,j} y_{i-1,k}.$$

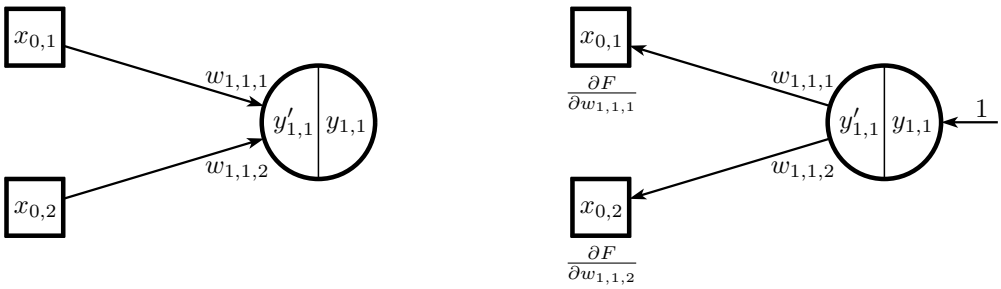


Figure 20: The backpropagation algorithm applied to a simple one-layer network. Left is the feed-forward step, right is the backpropagation step.

We will explain it by two simple examples. Consider Figure 20. The output of the network is

$$F = y_{1,1} = \phi(s_{1,1}) = \phi(w_{1,1,0} + w_{1,1,1} x_{0,1} + w_{1,1,2} x_{0,2}).$$

The derivatives are

$$\frac{\partial F}{\partial w_{1,1,0}} = \phi'(s_{1,1}) = y'_{1,1}$$
$$\frac{\partial F}{\partial w_{1,1,1}} = \phi'(s_{1,1})x_{0,1} = y'_{1,1}x_{0,1}$$
$$\frac{\partial F}{\partial w_{1,1,2}} = \phi'(s_{1,1})x_{0,2} = y'_{1,1}x_{0,2}$$

In this way we see

$$\frac{\partial F}{\partial \overline{\mathbf{w}}_{1,1}} = y'_{1,1}\overline{\mathbf{x}}_0 = y'_{1,1}\overline{\mathbf{x}}_1.$$

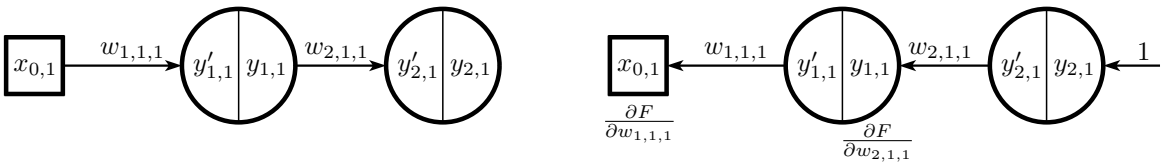


Figure 21: The backpropagation algorithm applied to a simple two-layer network. Left is the feed-forward step, right is the backpropagation step.

Now consider Figure 21. The output of the network is

$$F = y_{2,1} = \phi(s_{2,1}) = \phi(w_{2,1,0} + w_{2,1,1}y_{1,1}) = \phi(w_{2,1,0} + w_{2,1,1}\phi(w_{1,1,0} + w_{1,1,1}x_{0,1})).$$

The derivatives are

$$\frac{\partial F}{\partial w_{1,1,0}} = \phi'(s_{2,1})w_{2,1,1}\phi'(s_{1,1}) = y'_{2,1}w_{2,1,1}y'_{1,1}$$
$$\frac{\partial F}{\partial w_{1,1,1}} = \phi'(s_{2,1})w_{2,1,1}\phi'(s_{1,1})x_{0,1} = y'_{2,1}w_{2,1,1}y'_{1,1}x_{1,1}$$
$$\frac{\partial F}{\partial w_{2,1,0}} = \phi'(s_{2,1}) = y'_{2,1}$$
$$\frac{\partial F}{\partial w_{2,1,1}} = \phi'(s_{2,1})y_{1,1} = y'_{2,1}y_{1,1}$$

Ans so we see

$$\frac{\partial F}{\partial \overline{\mathbf{w}}_{2,1}} = y'_{2,1}\overline{\mathbf{x}}_2$$

$$\frac{\partial F}{\partial \overline{\mathbf{w}}_{1,1}} = y'_{2,1}w_{2,1,1}y'_{1,1}\overline{\mathbf{x}}_1$$

7.4 Training using quasi-Newton or conjugate gradient steps

Now let us consider the quasi-Newton and the conjugate gradient methods for training neural networks. Because both algorithms assume inexact line searches, it is not possible to take the training set pattern by pattern and update the weights a bit accordingly. But when we divide the training set randomly in some batches of patterns, and consider the batches apart, an inexact line search can make sense, if the batches contain enough patterns.

Now concerning quasi-Newton, it is questionable whether the matrix updates are very useful. It is sure that it is not possible to ensure positive definiteness for BFGS. When the neural network

contains many units, it can be advantageous to use the low-memory variant of BFGS.

Concerning the method of conjugate gradients, there can be problems because the CG-direction is not necessarily a descent direction. In such cases, we have to use the gradient direction instead.

7.5 Localization results by neural networks

To perform localization tests, we used the SNNS implementation of neural network simulaters, as described in [16]. Again, we used the measurement set from Figure 4. We splitted it in a training set (60 percent of the data), and a validation set and a test set (each 20 percent of the data).

We used a neural network with two hidden layers. The first hidden layer consisted from 8 units, and the second hidden layer consisted from 6 units. As input we used the mean of the received signal strengths for each spot in mW, rescaled by taking a logarithm. In this way we found mean error $\overline{E} = 0.54$ in the training set, $\overline{E} = 1.63$ in the validation set, and $\overline{E} = 1.51$ in the test set.

Although the difference with the results we had previously by the maximum likelihood estimation is not too big, we conclude that neural networks give significantly better results. However, we should notice that the neural network should be trained by enough representative data to be applicable. We can improve the results of the maximum likelihood estimation by using some more a priori knowledge about the room where the measurements took place. For example, we can clip the solutions of the maximum likelihood estimation by the room dimensions. There is more research needed at this point.

8 Conclusions and recommendations

The most important result in this research is the localization algorithm based on received signal strength, as presented in Section 6.4. This algorithm can be used to localize both active Bluetooth tags and passive UHF RFID tags. On the way to get this algorithm, we studied the received signal strength in detail, and derived a model in Section 5.4. This model relates the received signal strength to the distance between tag and the RFID reader, and describes the shape of disturbances in multipath environments in which the space between the tag and reader is unobstructed.

Concerning the relation between received signal strength and distance, we found that the Friis transmission equation, as written in (47), dominates the signal strength decay over distance. Nevertheless, we have to take into account the nonlinear effects in the electronics of passive tags, as we saw in Section 5.4. Moreover, big metal structures in the direct surrounding of RFID readers may also deform the signal strength decay over distance.

In Section 5.2 we saw that the received signal strength suffers from very intense disturbances, but these disturbances were modeled well by an exponential probability distribution, as was shown by our measurements and analysis by histograms. Here the measurements on passive tags were less convincing, and so here is more research needed.

The method of maximum likelihood allowed us to formulate the localization problem as an optimization problem in Section 6.2. Because this optimization problem could not be solved analytically, we depended on numerical methods. In Chapter 4 we discussed some common numerical methods, and we chose to apply the BFGS method, combined with inexact line search with Wolfe conditions.

The signal model from Section 5.4 allowed us also to simulate the received signal strength, and in this way we were able to research easily the dependence of the localization accuracy on the physical parameters in the localization problem, as was carried out in Section 6.5. Analysis by artificial neural networks in Chapter 7 confirmed our results.

Unfortunately, the localization results reveal that it is not possible to do much better than zone detection with the used Intellifi equipment. However, if the equipment is adapted to obtain more independent signal strength measurements, our localization procedure will likely outperform zone detection.

The literature study in Section 1.6 suggests to use different techniques for better localization results. If possible, a time-based method should be used, because these are believed to be most accurate. Furthermore, the angle of arrival method and the phase difference of arrival method can also be used to improve the localization results. Especially the latter one is attractive, because

the radio communication between active Bluetooth tags and spots already take place at multiple frequencies, which is essential in this method.

References

[1] A. Bhatia, B. Mehta, and R. Gupta, *Different Localization Techniques for Real Time Location Sensing using passive RFID*, Indian Institute of Technology, Kanpur, India, 2007.

[2] M. Bouet and G. Pujolle, *3-D Localization Schemes of RFID Tags with Static and Mobile Readers*, in *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, Springer Berlin Heidelberg, pp. 112-123, 2008.

[3] M. Bouet and A.L. dos Santos, *RFID tags: Positioning principles and localization techniques*, in Proceedings of Wireless Days 2008 (WD '08), 2008.

[4] R. Bridelall and A. Hande, *Performance Metrics and Operational Parameters of RFID Systems* in *RFID Systems: Research Trends and Challenges*, edited by M Bolić, D. Simplot-Ryl, and I. Stojmenović, John Wiley & Sons Ltd, Chichester, United Kingdom, 2010.

[5] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, and L.E. Meester, *A Modern Introduction to Probability and Statistics: Understanding Why and How*, Springer-Verlag London, 2005.

[6] Daniel M. Dobkin, *The RF in RFID: UHF RFID in Practice*, second edition, Newnes, Waltham, Massachusetts, 2013.

[7] Y. Gu, A. Lo, and I Niemegeers, *A survey of indoor positioning systems for wireless personal networks*, in Communications Surveys & Tutorials, IEEE, Volume 11, Issue 1, 2009.

[8] J.A. Rice, *Mathematical Statistics and Data Analysis*, third edition, Thomson Brooks/Cole, Belmont, California, 2007.

[9] E.D. Kaplan et al. *Fundamentals of Satellite Navigation* in *Understanding GPS: Principles and Applications*, edited by E.D. Kaplan and C.J. Hegarty, second edition, Artech House Boston London, 2006.

[10] X. Li, Y. Zhang, and M. Amin, *Multifrequency-based range estimation of RFID Tags*, in *IEEE International Conference on RFID*, pp. 147-154, 2009.

[11] M. Matsumoto, and T. Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, in *ACM Transactions on Modeling and Simulation*, vol. 8, no.1, pp. 3-30, 1998.

[12] J. Nocedal, and S.J. Wright, *Numerical Optimization*, Springer-Verlag New York, 1999.

[13] K. Saniee, *A Simple Expression for Multivariate Lagrange Interpolation*, SIAM Undergraduate Research Online, vol. 1, no. 1, 2008.

[14] H. Sizun, *Radio Wave Propagation for Telecommunication Applications*, Springer-Verlag Berlin Heidelberg, Germany, 2005.

[15] M. Stella, M. Russo, and D. Begušić, *RF Localization in Indoor Environment*, in *Radioengineering*, vol. 21, no. 2, pp. 557-567, 2012.

[16] A. Zell, G. Mamier, M. Vogt, et. al., *SNNS Stuttgart Neural Network Simulator User Manual*, version 4.2, 2000. http://www.ra.cs.uni-tuebingen.de/SNNS/

A Appendix: Program code for MATLAB

This appendix contains a localization program code for MATLAB. It consists from 5 seperate m-files. We also give the program code which we used for simulating the received signal strength.

localize.m

```
function x = localize(a,ss,C,g)
% a is a cell-array with the spot locations
% ss is a cell-array with the received signal strengths in dBm per spot
% C is an array with the constant C per spot
% g is the path loss exponent gamma

    m = numel(a); % number of receivers (spots)

    n = zeros(m,1);
    p = zeros(m,1);
    for j=1:m
        n(j) = numel(ss{j});
        p(j) = mean(10.^(ss{j}/10))/C(j);
    end

    x = zeros(size(a{1}));
    for j=1:m
        x = x + n(j)*p(j)*a{j};
    end
    x = x / (n'*p);


    %optimize logl_0
    %if(abs(g-2)>1e-10)
        [x0,~] = optimize(@func_logl0,x,a,p,n,g);
    %end

    %optimize least squares
    [x1,~] = optimize(@func_ls,x0,a,p,n,g);
    [x2,~] = optimize(@func_ls,2*x0-x1,a,p,n,g);

    if(func_logl(x2,a,p,n,g)<func_logl(x1,a,p,n,g))
        x1=x2;
    end

    %optimize loglikelihood function
    [x,~] = optimize(@func_logl,x1,a,p,n,g);


end
```

optimize.m

```
function [x,f] = optimize(fun,x,a,s,n,g)

    %plot_fun(fun,x,a,s,n,g); pause;

    [f,df] = fun(x,a,s,n,g);

    Binv = eye(numel(x));
    t = 1;

    tol=1e-12;

    k=1;
    kmax=1000;
    while ( k<=kmax && df'*df > tol )

        %determine Quasi-newton BFGS direction
        d = -Binv*df;

        xp = x;
```

```
        dfp = df;

        %inexact line search
        [~,x,f,df] = line_search(fun,x,a,s,n,g,d,1,f,df'*d);

        %update Binv
        dx = x - xp;
        ddf = df - dfp;
        dxddf = dx' * ddf;
        Binv = ( eye(numel(x)) - dx*ddf'/dxddf ) * Binv * ( eye(numel(x)) - ddf*dx'/dxddf ) + dx*dx'/dxddf;

        k=k+1;

    end
end

function plot_fun(fun,x,a,s,n,g)
    if(numel(x)==1)
        t=linspace(x-20,x+20);
        ft=zeros(size(t));
        dft=zeros(size(t));
        for i=1:numel(t)
            [ft(i),dft(i)] = fun(t(i),a,s,n,g);
        end
        figure(1);
        plot(t,ft);
        figure(2);
        plot(t,dft,'r');
    end
    if(numel(x)==2)
        t1=linspace(x(1)-10,x(2)+10,30);
        t2=linspace(x(2)-10,x(2)+10,30);
        [T1,T2]=meshgrid(t1,t2);
        T=zeros(size(T1));
        for u1=1:numel(t1)
            for u2=1:numel(t2)
                [T(u1,u2),~]=fun([T1(u1,u2);T2(u1,u2)],a,s,n,g);
            end
        end
        figure
        contourf(T1,T2,T,80);
    end

end


function [t,x,f,df] = line_search(fun,x0,a,s,n,g,d,t0,f0,fa0)

    c1=1e-4;
    c2=0.9;

    tp=0.0;
    t=t0;
    x=x0;

    fp=f0;
    f=f0;

    k=1;
    kmax=100;
    while k<=kmax

        [f,df] = fun(x0+t*d,a,s,n,g);
        fa=df'*d;

        if( (f > f0 + c1*t*fa0) || ( (f >= fp) && (k>1) ) )
            [t,x,f,df] = zoom(fun,x,a,s,n,g,d,tp,t,f0,fa0,fp,c1,c2);
            return;
        end
        if abs(fa)<=-c2*fa0
            x=x0+t*d;
            return;
```

```
        end
        if fa>=0
            [t,x,f,df] = zoom(fun,x,a,s,n,g,d,t,tp,f0,fa0,fp,c1,c2);
            return;
        end
        tp=t;
        fp=f;

        t=2*t;
        k=k+1;

    end
end

function [t,x,f,df] = zoom(fun,x0,a,s,n,g,d,t_lo,t_hi,f0,fa0,f_lo,c1,c2)
    k=1;
    kmax=40;
    while k<=kmax
        t = (t_lo + t_hi) / 2;

        x = x0+t*d;

        [f,df] = fun(x,a,s,n,g);
        fa = df'*d;

        if ( (f > f0 + c1 * t * fa0) || (f >= f_lo))
            t_hi = t;

        else
            if abs(fa)<=-c2*fa0
                return;
            end
            if fa * (t_hi - t_lo) >= 0
                t_hi = t_lo;
            end
            t_lo = t;
            f_lo = f;
        end
        k = k+1;

    end
end
```

func_logl0.m

```
function [f,df] = func_logl0(x,a,s,n,g)

    m = numel(a);
    dist = cell(size(a));
    r2=zeros(m,1);

    for j=1:m
        dist{j} = x - a{j};
        r2(j) = dist{j}'*dist{j};
    end

    %compute f and df (diff and r2 depend on x)
    f=fx(r2,s,n,m,g);
    df=dfx(r2,s,n,m,g,dist);

end

function [f] = compute_fx(x,a,s,n,g)

    m = numel(a);
    r2=zeros(m,1);

    for j=1:m
        dist = x - a{j};
        r2(j) = dist'*dist;
    end
```

```
    %compute f and df (diff and r2 depend on x)
    f=fx(r2,s,n,m,g);

end

function f = fx(r2,s,n,m,g)
    %objective, to be optimized

    g2 = g/2;

    f = 0;
    for j=1:m
        f = f + n(j) * s(j)*(r2(j)^g2);
    end

end

function df = dfx(r2,s,n,m,g,dist)
    %gradient, with precomputed distances
    g2 = g/2;

    df = zeros(numel(dist{1}),1);
    for j=1:m
        df = df + dist{j} * n(j) * s(j) * r2(j)^(g2-1);
    end
    df = g * df;
end
```

func_ls.m

```
function [f,df] = func_ls(x,a,s,n,g)

    m = numel(a);
    dist = cell(size(a));
    r2=zeros(m,1);

    for j=1:m
        dist{j} = x - a{j};
        r2(j) = dist{j}'*dist{j};
    end

    %compute f and df (diff and r2 depend on x)
    f=fx(r2,s,n,m,g);
    df=dfx(r2,s,n,m,g,dist);

end

function f = fx(r2,s,n,m,g)
    %objective, to be optimized

    g2 = g/2;

    f = 0;
    for j=1:m
        f = f + ( n(j) * ( s(j) * r2(j)^g2 - 1) )^2;
    end

end

function df = dfx(r2,s,n,m,g,dist)
    %gradient, with precomputed distances
    g2 = g/2;

    df = zeros(numel(dist{1}),1);
    for j=1:m
        df = df + dist{j} * n(j)^2 * s(j) * ( s(j) * r2(j)^g2 - 1 ) * r2(j)^(g2-1);
    end
    df = g * df;
end
```

func_logl.m

```
function [f,df] = func_logl(x,a,s,n,g)

    m = numel(a);
    dist = cell(size(a));
    r2=zeros(m,1);

    for j=1:m
        dist{j} = x - a{j};
        r2(j) = dist{j}'*dist{j};
    end

    %compute f and df (diff and r2 depend on x)
    f=fx(r2,s,n,m,g);
    df=dfx(r2,s,n,m,g,dist);

end

function [f] = compute_fx(x,a,s,n,g)

    m = numel(a);
    r2=zeros(m,1);

    for j=1:m
        dist = x - a{j};
        r2(j) = dist'*dist;
    end

    %compute f and df (diff and r2 depend on x)
    f=fx(r2,s,n,m,g);

end

function f = fx(r2,s,n,m,g)
    %objective, to be optimized

    g2 = g/2;

    f1 = 0;
    f2 = 0;
    for j=1:m
        f1 = f1 + n(j) * s(j)*(r2(j)^g2);
        f2 = f2 - n(j) * log(r2(j));
    end
    f = f1+g2*f2;

end

function df = dfx(r2,s,n,m,g,diff)
    %gradient, with precomputed distances
    g2 = g/2;

    df = zeros(numel(diff{1}),1);
    for j=1:m
        df = df + diff{j}/r2(j) * n(j) * (s(j) * r2(j)^g2 - 1);
    end
    df = g * df;

end
```

simulate.m

```
function s = simulate( n, r, C, g )
s = 10*(log10(C)-g*log10(r)+log10(-log(rand(n,1))));
if(g>2.1) %passive tag
    s = s(s>-80);
else %active tag
    s = s(s>-110);
end
s = round(10*s)/10;
end
```