UNIVERSITY OF GRONINGEN

MASTER THESIS

Imaging Lightning with the Extended Kalman Filter

Author:
Alex PEL

Supervisor:
Prof. Dr. Olaf SCHOLTEN

June 28, 2019

rijksuniversiteit
groningen

Abstract

In this Master Thesis we are aiming to improve current lightning imaging techniques in the radio spectrum by means of a Kalman Filter. Current LOFAR-based methods fail whenever the inter-pulse distance reaches 150 ns, and the Kalman Filter has the potential to lower this boundary. It is shown that the Kalman Filter can predict the location in time of a given pulse accurate to a few nanoseconds in most cases. Exceptions remained within a few tens of nanoseconds and were only seen in the case of a crude initial estimate or deviating shapes of pulses, for which the technique still has to be optimized in the second case. Also, we show that even though we use a linearized version of the Kalman Filter in our non-linear model, the imaging can still be done close to optimality.

Contents

1 Introduction **1**

2 Lightning Physics **4**
2.1 Charge separation 4
2.2 Initiation 5
2.3 Propagation 6
2.4 Measurements in the Netherlands 7

3 The LOFAR Telescope **8**
3.1 LOFAR signal processing 9
3.2 Processed block example 10

4 The Kalman Filter **14**
4.1 The linear filter 16
4.1.1 Applying the filter 17
4.1.2 Two identical carts 22
4.2 The non-linear filter 24
4.2.1 EKF simulation: spherical wave emitted from point source . 27
4.3 Methods used with LOFAR data 29

5 Results **36**

6 Conclusions **37**

7 Acknowledgements **38**

A Derivations **39**
A.1 State vector evolution 39

B Kalman Filter Scripts **40**
B.1 Linear Kalman Filter applied to one or two carts 40
B.2 EKF simulation: spherical wave emitted from point source 42
B.3 EKF to LOFAR data 46

Chapter 1
Introduction

As frequent as lightning occurs (about (44 ± 5) strikes hit the earth per second [1]), as little is known about the exact mechanism driving this natural phenomenon. From the ancient times when it was believed the gods (Zeus, Thor, Indra etc.) were responsible for the phenomenon, to the first known scientific experiment involving lightning (the "*Kite experiment*" by Benjamin Franklin 1752 [2]); lightning is a phenomenon that has fascinated us for thousands of years, but still remains elusive. Therefore, research is continuing in this area, to unravel more pieces of the puzzle and hopefully coming to a complete understanding of the enchanting spectacle in the future.

As mentioned before, the first revelation of lightning research came presumably from Benjamin Franklin with his Kite experiment (although there is some controversy around this [3]). A few years before the actual experiment several pioneer engineers and electricians separately (working independently from different countries such as England, Germany, America) noted the similarity between lightning and electricity, and suggested theories for it. The famous experiment Franklin then carried out, was presumed to be performed in 1752 by flying a kite into a thunderstorm. To demonstrate that electricity was at place during a thunderstorm, Franklin mounted an iron key to the kite to store the charge. A wet hemp rope would ensure that the wire attached to the kite became conductive, and hence could transfer charge from the air to the key. Another wire was connected to a Leyden Jar, an early form of a capacitor, who could also register that charge had been accumulated. As the experiment was carried out by his son flying the kite, the kite wasn't (luckily) directly hit by lightning, but did receive a charge as a result from the charge in the air, and hence the electric nature of thunderclouds was confirmed.

The century after Franklin demonstrated his experiment, not much news came to light in this field, except for a so called "lightning rod", which Franklin invented to keep structures safe from lightning damage (a grounded rod mounted on top of the structure, leading the lightning current into the ground). Franklin also showed that in some cases, the lower part of the thundercloud was positively charged, but in most cases negatively charged. Today we know this is true in about 90% of the cases [4]. It was only until the late nineteenth century that new tools became available to study the short-lasting lightning flashes.

As an average lightning flash lasts about 0.2 seconds [5], which is made up of shorter "strokes" or flashes which last for about 60-70 *microseconds*, it was extremely difficult to qualitatively study lightning without the use of photography or spectroscopy. When these tools arrived in the late nineteenth century, new advances came to the area, and it was shown that lightning often contains two or more strokes, as can be seen from figure 1.1. This discovery was led by the invention

of the streak camera [6], after which multiple advances were made. One of which was the discovery that lightning strikes lowering negative charge to the ground, the "negative leader", was for the first stroke "stepped", and was followed by an upward "return stroke"[7]. This was also the start of the lightning terminology, which will be elaborated on in the next section. "Thunder" on the other hand is the product of lightning, which is the sound emitted by the rapidly expanding air, due to the sudden increase in temperature within a lightning flash. As the gas in the air expands at a very fast rate, it creates a pressure shock, leading to a sound ranging from a loud crack to a low "rolling" rumble.

FIGURE 1.1: A lightning flash captured with a horizontally moving camera. It appears to have four strikes hitting the ground along the same channel, where the duration between two separate strikes is of the order of tens of milliseconds. The picture was taken from [8].

Today, lightning research is in a completely different ballpark. Because lightning does not only emit radiation in the visible spectrum but also produces radio pulses, it can be studied by radio telescopes as well. One telescope performing radio observations, namely "LOFAR" which stands for LOw-Frequency-ARray, has shown to image lightning processes to unprecedented precision[9]. The current accuracy of lightning imaging with LOFAR has reached the meter/nanosecond level in the 30-80 MHz range, and current algorithms handle a relatively low-populated spectrum very well. This means the radio pulses emitted by lightning are observed separately enough such that they can be identified individually. The spectra during thunderstorms however can get extremely dense (e.g. multiple pulses arriving at the same time). Such a large population of pulses is not yet handled with the existing methods, and the hope is that this work can improve on this by means of the so called "Kalman Filter".

In the next section we will give a general overview of lightning development. After that we will shortly go over the LOFAR telescope and how it relates to our

research. In the sections following that, we dive into the mechanism of the (Extended) Kalman Filter, which is an algorithm combining measurements of a system with the expected results based on physical evolution of the system. We derive the proper form of the linear and non-linear filter, and how it should be used in our research. After that we will discuss our results which were made by using the filter.

Chapter 2
Lightning Physics

Lightning is the discharge of electric energy through the atmosphere. It is the phenomenon which can brilliantly light up the sky, although not all lightning processes emit visible radiation. The brilliant light either comes from thermal radiation of the extremely high (near *30000 K*[4]) temperature air, which gets transformed into a plasma as the electrons are accelerating through it with great energies, or from ohmic heating due to large numbers of electrons with low energies. The acceleration of these electrons are due to (strong) electric fields. The specific picture in figure 1.1 is a demonstration of a so called "cloud-to-ground" (CG) lightning, which speaks for itself. Other variants during a thunderstorm can be cloud-cloud (CC) or intra-cloud (IC) lightning, when either a lightning discharge occurs from one cloud to another or inside the same cloud respectively. Intra-cloud lightning is the most occurring type during a thunderstorm.

2.1 Charge separation

Charge separation inside thunderclouds is the first important process to generate thunderstorms. The ingredients for development are moist, (rapid) rising air (convection streams inside the clouds) and the presence of ice crystals and graupel (soft hail). In general the warmer air is near the ground and colder air higher in the atmosphere, and as a result the warmer air becomes buoyant. This "updraft" of air leads the charging process. In the central region of the storm, where temperatures range from -10 to -25 degrees Celsius [10], the upward draft carries small ice crystal upwards. At the same time the graupel tends to fall down or be suspended in the air due to its larger volume and density. These particles then collide inside the cloud. At each collision, charge is transferred from the ice crystals to the graupel, with as result that the graupel becomes negatively charged and the ice crystals positively charged. As the graupel tends to fall down, the bottom or central part of the cloud accumulates negative charge and the top part of the cloud positive charge. Also, there is some positive charge near the bottom of the cloud, but where this charge comes from is not yet fully understood. See figure 2.1 for a schematic picture of the charge distribution of a thundercloud.

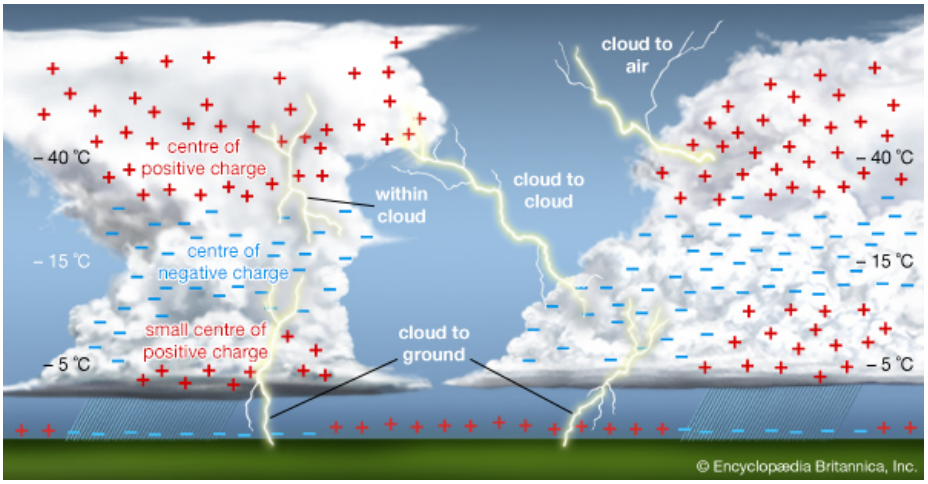

FIGURE 2.1: A picture of charge distribution inside a thundercloud. Inside the central region the charging process takes place by means of collisions of ice crystals and graupel. The updraft carries the positively charged ice crystals upwards while the negatively charged graupel remains suspended in the air or sinks towards the lower part of the cloud. The figure is credited to [11].

The separation of these charges makes the cloud act like a capacitor and generates an electric field between them. If the fields are strong enough, "dielectric breakdown" (transformation of an insulator to a partial conductor) of the air would occur, making the air conductive and able to transport a current, leading to the initiation of lightning.

2.2 Initiation

The electrical breakdown of air at sea level is on the order of 10^6 V/m [12]. Therefore, if the fields would exceed this threshold, lightning would initiate automatically. The fields measured during thunderstorms however, are more than an order of magnitude too weak to induce breakdown [4], and it is still an open question how initiation exactly occurs. One hypothesis for initiation of lightning involves extraterrestrial origins: cosmic rays. It is suggested [13] [14] that initiation could be stimulated by cosmic rays (secondaries). Others propose the idea that hydrometeors (ice crystals on the order of centimeters) can enhance the electric field inside thunderclouds locally, leading to local breakdown around the tip of these hydrometeors, or a combination of both of the two hypotheses[12]. Hopefully the detailed imaging provided by LOFAR can resolve this issue. One big problem obstructing the current imaging process is that the measured spectrum has a very high density of (radio) pulses, to such an extent that the current imaging procedure fails. The hope is that the new procedure discussed in this work can resolve some of these problems.

2.3 Propagation

As the initial breakdown occurs somewhere in the thundercloud, a conductive channel is established and the charge residing in the surrounding region will start to move due to the electric field. This polarizes the (conductive) channel, which leads to a positively charged "tip" (charge at the end of the conductive channel) at one end, and a negatively charged tip at the other end. How these charges propagate through the clouds or atmosphere depends on the polarity of the charge. We distinguish the two by the "negative leader" having a negatively charged tip, and the "positive leader" by having positive charge at the tip. The positive leader propagates differently from the negative leader, but why this happens is not yet understood.

Negative Leaders
The negative leader is often called "stepped leader", and gets its name from sequentially moving and stopping as it propagates. This stepping process, which is shown in figure 2.2, proceeds in steps of ten to hundreds of meters with inter-step intervals ranging from several microseconds to tens of microseconds [15]. Why this happens is not yet understood. It is known that among visible light, this stepping process also emits VHF ("Very High Frequency", 30-300MHz) radiation. This happens at the tip of the leader, and it is currently (by LOFAR-based research) under investigation why this process emits this VHF radiation. The research discussed in this paper has the potential to also contribute to this.

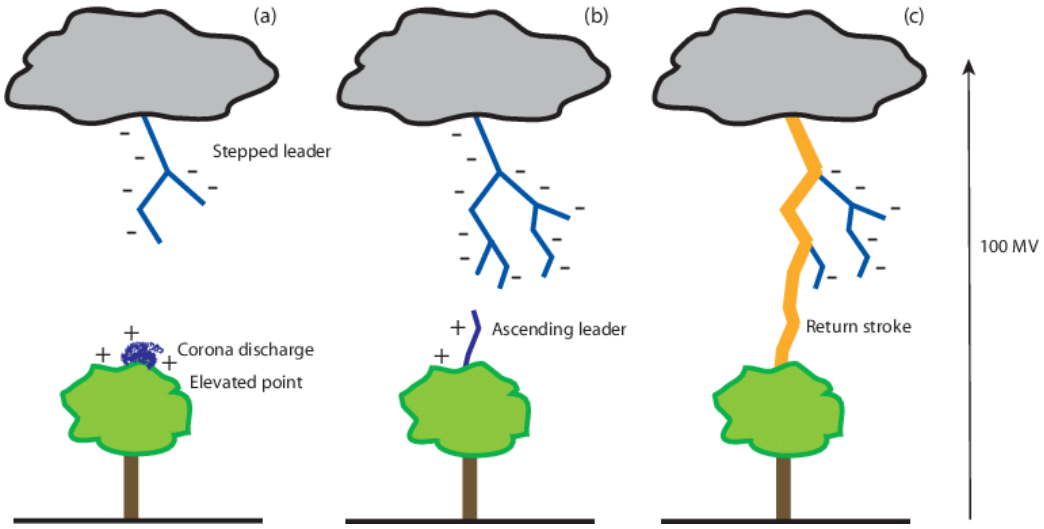

FIGURE 2.2: The stepping process shown in a simple schematic picture. The individual branches do not flow continuously but make one step, stop for a short amount of time, and then make the next step again. Also the "ascending leader" is shown which can develop from ground to air (in this case a tree). Soon as the two connect with each other, all the negative charge will flow trough the channel into the ground, with velocities around 1/3 the speed of light [10]. This bright flash of current is called the "return stroke", and is observed to be able to occur several times in the same channel in a short window of time, as seen earlier in figure 1.1. Therefore lightning does strike the same place twice! Figure credit: [16].

Positive leaders
Positive leaders have been studied much less than negative leaders. A simple reason for this is that the positive flashes occur much less than negative flashes. Also, positive flashes are less luminous in the visible and VHF spectrum than negative flashes, making it harder to study them[10]. The propagation of positive leaders which *has* been observed, is also different from that of the negative leaders. In general positive leaders propagate more continuously, but can also propagate in steps, which is still different from their negative counterpart.

2.4 Measurements in the Netherlands

Contrary to the lightning research mentioned previously, thunderstorms in the region of the Netherlands have been studied with much less detail. A simple reason for this is that in this region, occurrences of thunderstorms are much less frequent (and usually also less violent). The ones that have been studied are yet to show if the thundercloud structure is the same as mentioned before. In the research done until today (source: private discussions with Olaf Scholten), it is shown that (lightning) activity is only seen in the lower two layers of the thunderclouds. In the upper part (above 8 km), no activity in either the radio or visible spectrum was measured during the thunderstorms. This could be an indication that dutch thunderstorms behave differently than thunderstorms in and around the tropics, or that their top part is much less developed. It is known that in tropical regions the cloud tops usually exceed heights of 12 km. In Dutch storms only discharges from the center or the lower part of the thundercloud have been observed (by either "Lightning mapping arrays" or LOFAR), which would indicate that there would be only two layers of charge. There is however research done [17], where the electric field structure was probed by means of studying radio emissions from cosmic ray air showers using LOFAR. This study indicated that thunderclouds in the Netherlands indeed have the three-layer (charge) structure.

Even though thunderstorms have been studied much less frequently in the Netherlands, they have been recorded in great detail. LOFAR, which will be discussed in the next section, has proven to image lightning to unprecedented precision [9]. This telescope, from which data is also used in this research, together with the current imaging techniques, have already lead to newly discovered structures on positive leaders [18], and may very well lead to other new insights in the field of lightning.

Chapter 3

The LOFAR Telescope

LOFAR (LOw-Frequency-ARray) is currently one of the largest radio telescopes in the world, operated by the Dutch radio observatory ASTRON. It consists of a large number of antennas spread out over a large area, which are combined to function as one large telescope. The dutch "stations", a collection of antennas, cover an area of about 3,200 km^2. There are also stations Germany, France, UK, Ireland, Poland and Sweden, but these will not be used in this work. Stations (where the locations of the dutch stations are shown in figure 3.1) consist of two types of antennas; low-band antennas (LBA) recording in the 10-90MHz frequency range and high-band antennas (HBA) recording in the 110-250 MHz frequency range[19]. Together they are sensitive to several VHF phenomena such as cosmic-ray air showers, lightning but also astronomical events. In this work only the LBAs will be used.

FIGURE 3.1: The distribution of the LOFAR antennas over the North-East of the Netherlands with remote stations (RS) tagged with their name. The LOFAR core is located somewhere around the center of the remote stations (piled up green markers). The antennas are spread out over distances up to 100 km from the core and are combined by optical fibre cables to work as one single large telescope. Together the dutch antennas span an area of about 3,200 km^2, but there are antennas in other countries as well. The figure was taken from [9].

The LBAs consist of two (inverted) V-shape dipole antennas which have a pyramid shape (figure 3.2). There are 96 LBAs per station who record in the 10-90 MHz

frequency range and are sampled at 200MHz. They are positioned orthogonal to each other such that they are sensitive to two orthogonal polarizations. To limit the data, about five LBAs per station are used in this work. Although LOFAR was originally designed for astronomical and cosmic ray air shower observations, anomalies found in measurements during thunderstorms led to expectations that LOFAR was also suited to probe lightning phenomenon. These expectations were later shown to be correct, as it is shown that LOFAR can detect pulsed VHF emission from lightning with sub-ns precision [9].

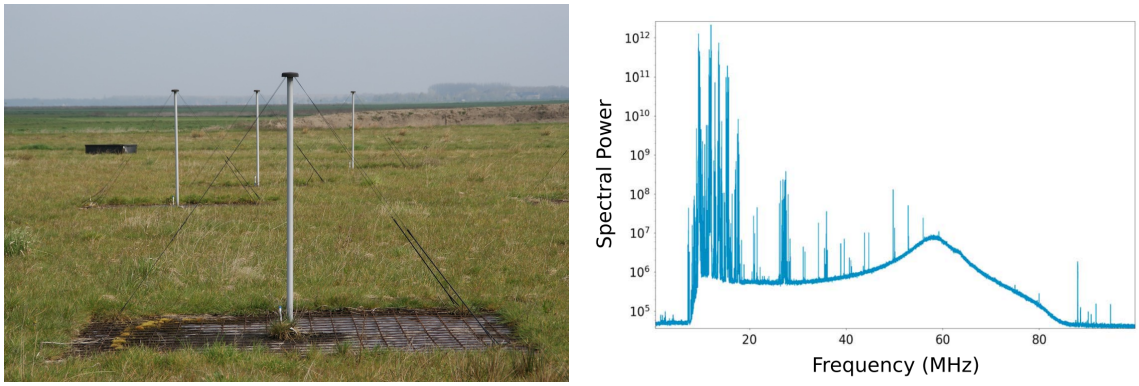

FIGURE 3.2: A picture of the low-band antennas (LBAs) of LOFAR (left), and the typically measured spectral power per frequency averaged over a large period of time (right). The sharp peaks in the spectrum are caused by human radio frequency interference (RFI), which are mainly below 30MHz and above 80 MHz, and are due to local radio stations. The resonance peak is located around 58 MHz, which is partly due to the size of the antenna and the balun. Image credit: [20],[9] for left and right figures respectively.

3.1 LOFAR signal processing

As lightning develops, it emits radio pulses which can be observed in a wide frequency band. We limit ourselves to part of the VHF-band of which the LOFAR LBAs are sensitive to, and where human radio activity is minimal, which is the frequency interval of 30-80 MHz. In this frequency range, there are still human-generated phenomena which are measured by the LOFAR antennas, and are of no use to our research. These influences are minimized by the following procedures discussed.

As shown in figure 3.2, the data recorded by LOFAR suffers from human radio frequency interference (RFI), which are due to (local) radio stations (sharp peaks in the spectrum). To remove this influence we perform "RFI mitigation" on every active LBA. In this process the first and last 10% of the block is multiplied by a "half-Hann window" (a simple de-/increasing function from 1 to 0 or vice versa) to remove edge effects. Also, the data is Fourier transformed and band-pass filtered between 30-80 MHz since most human generated RFI is outside this range. For a more complete description of the RFI filtering process done on LOFAR data (e.g. filtering in 30-80 MHz region based on stability of relative phases), we refer

to [9].

Also, the antennas at LOFAR are sampled at 200 MHz which means the sampling time is 5 ns. With upsampling methods however, the resolution can be brought under one ns, as is done in this work with an upsampling factor of eight (which gives 5/8 ns resolution). Lightning can also saturate the antenna during violent emissions. The spectrum data points are set to zero for all the points which saturate the antennas up until 50 data points after the end of the saturation. In the final stage the Hilbert envelope of the signal is plotted, which can be thought of as representing the energy deposited by the electromagnetic wave as a function of time. The result is shown in figure 3.3.

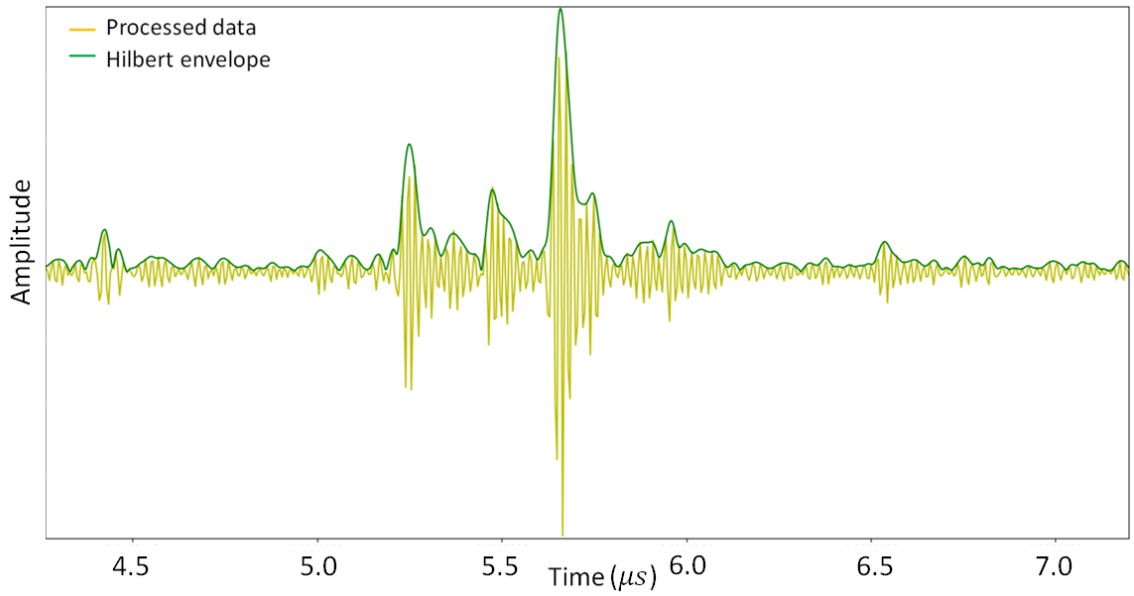

FIGURE 3.3: A small part of a measured spectrum, recorded by a core station antenna. In the processing of the spectrum, saturation has been removed, RFI mitigation has been applied and the real component of the signal is then shown (yellow). The spectrum then has been upsampled by a factor of eight (5/8 ns resolution) together with taking the Hilbert envelope (green).

3.2 Processed block example

Signal processing is done in "blocks" of 2^{16} data points (before upsampling), or equivalently in time intervals of about 327 μs. These blocks are the data which is worked with, and an example of a block of data recorded during a thunderstorm is shown in figure 3.4.

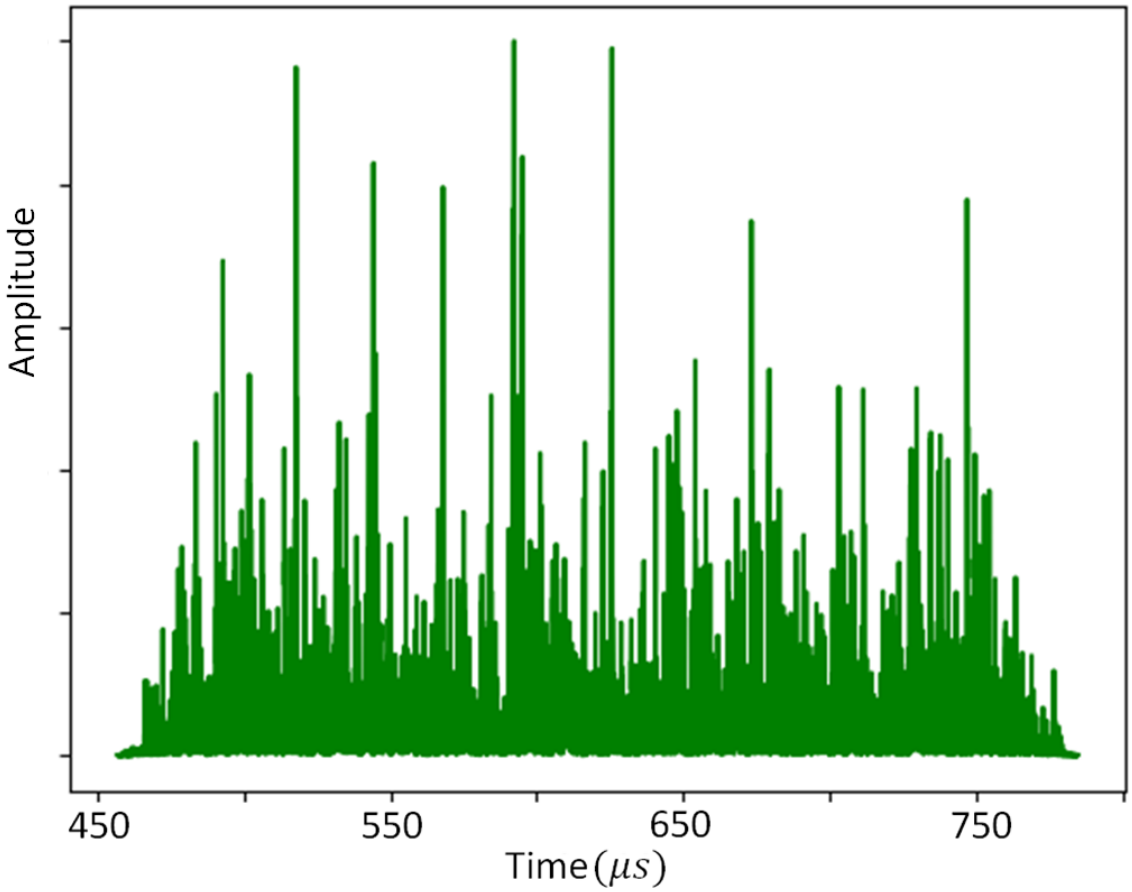

FIGURE 3.4: A moderately dense, processed spectrum recorded by a LOFAR core station LBA during a thunderstorm. Time is plotted against normalized intensity of the Hilbert Envelope of the signal for a full block. As one can see there are many pulses per time interval of 50 microseconds. Time-varying currents in the atmosphere are assumed to be the physical source for the emission of these VHF radio pulses. Our goal is to find the location in time in each antenna, for a specifically chosen pulse.

As one can see the measured spectrum is filled with many pulses. As we move from one antenna to the next (another antenna in either the same station or the next), the spectrum will look quite similar, with probably some deviation in the amplitudes and positions in time of the pulses w.r.t. other antennas. For some pulses, recognizing the same event in different antennas can be obvious, as shown in figure 3.5. Here the recorded spectra of four antennas are plotted above each other. Some pulses are isolated and have a very distinct, high amplitude, for example around $t = 555\mu s$ in figure 3.5. If we obtain the location in time for this pulse in all recording antennas (around 100 antennas), and the location in space of the corresponding antennas (which are known to us), we can use a non-linear fitter to obtain the location in time and space for the specific event which has emitted the pulse. This relies on the assumption that the event is produced by a point source emitting a spherical wave. The physical sources assumed responsible for emitting these spherical radio-waves are time-varying currents in the atmosphere. How these are developed is not yet known, and this research hopes to improve current imaging of these sources to gain more insight in the corresponding processes.

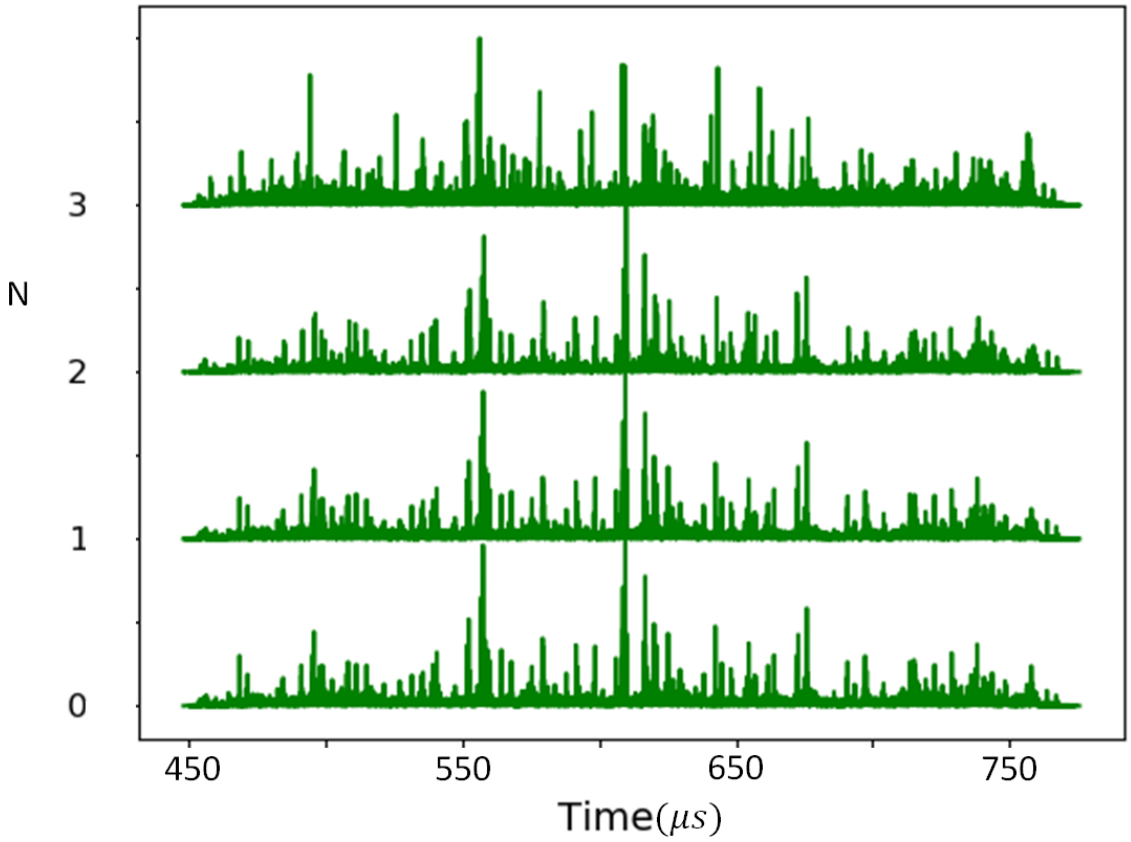

FIGURE 3.5: The same events measured by four different antennas plotted above each other. N indicates the antenna number, and as we can see the spectra are almost identical for these four antennas. Differences in space between the antennas recording the top and bottom spectrum are about 80 km, with the spectra in between recorded at intermediate distances. Some pulses can be recognized in all antennas, and to obtain the location in time of these specific pulses (for each antenna) is not a difficult task. If this is done, the location in space and time can be obtained for the specific event, and this is how lightning is imaged in this research.

The imaging is done by determining the locations in time and space of the individual events (which correspond to the individual pulses observed in the spectrum). If we imagine doing this for every pulse in the total spectrum (a part of the total spectrum shown here is about 327 microseconds long, but the total recording time is around 1-2 seconds), a very detailed picture of lightning processes can be given. This will probably lead to new discoveries in the field of lightning, as is already done by earlier LOFAR-based research [18]. The current problem however, is that the presently existing methods are not able to image all spectra. As the separation between two neighbouring pulses reaches 150 ns, the current imaging method fails to analyze them. Example spectra are shown in figure 3.6:

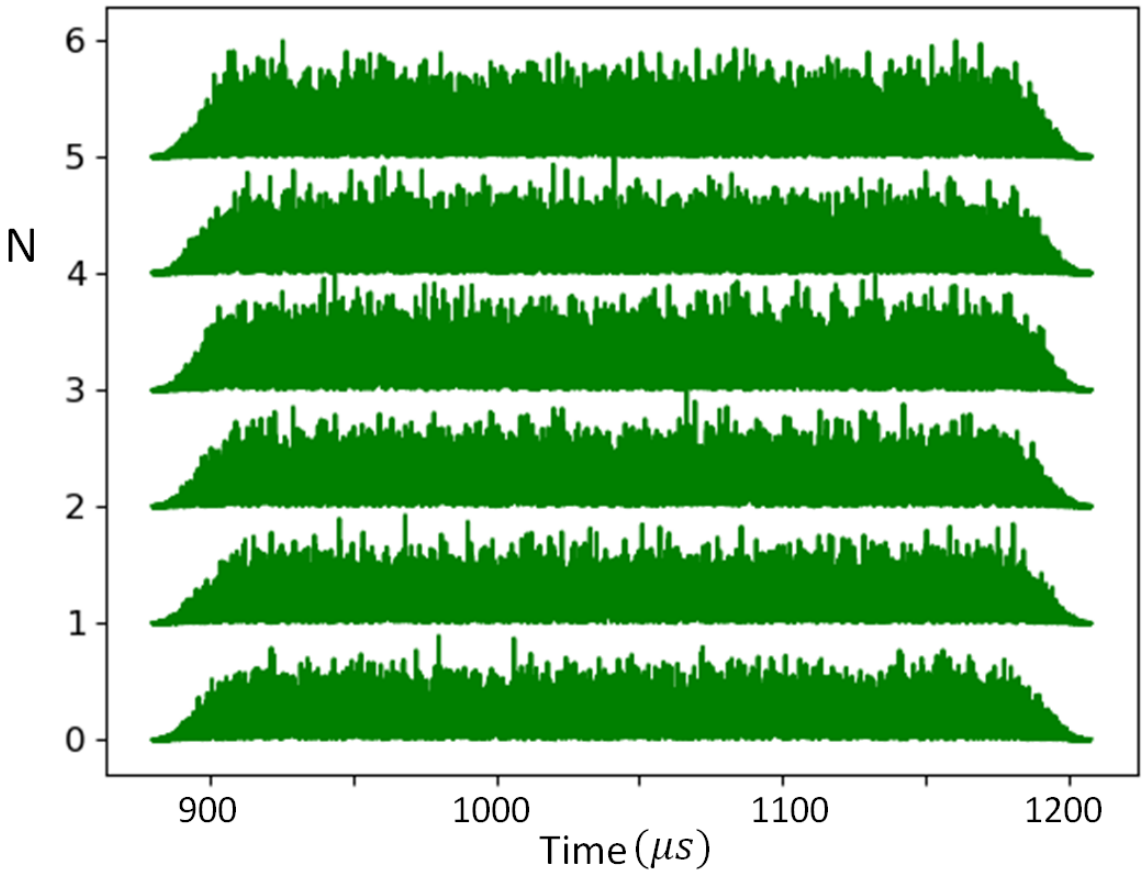

FIGURE 3.6: High-density spectra recorded by several LOFAR antennas during a thunderstorm. In these regions it is non-trivial to determine which pulse in one antenna belongs to which pulse in other antennas. Existing methods fail to analyze these parts, and the hope is that this can be done by means of a Kalman Filter.

As one can see, the spectra shown contain a lot more events per block than the previous spectra shown. It is impossible to image these regions with current methods used, since it has no way to determine which events belong to each other as we compare the spectra from different antennas. There is however potential to make a very detailed picture of the processes involved in lightning (which emit these pulses), if these region could be analyzed. We will see in the next chapter that the Kalman Filter has the ability to predict the location of a pulse we are interested in, in the antenna we are interested in. This is shown to be done with good accuracy, which promises to improve the current imaging techniques.

Chapter 4
The Kalman Filter

The Kalman Filter (named after the Hungarian electrical engineer and mathematician Rudolf E. Kálmán[21]) is an algorithm that uses a series of measurements (usually observed over time) containing statistical noise. In its early years it was used at the Apollo missions program, to estimate the trajectory of manned spacecraft going to the moon and back. Later it was shown to be applicable in many situations, for example in the field of particle tracking at the LHC. The working of the Kalman Filter relies on the combination of a calculated state of the system with a measured state of the system, with each its corresponding uncertainties. This calculated state is based on the physical laws governing the system. The combined estimate tends to be more accurate than the measurements alone, and in the case of a linear model it can be shown that the Kalman Filter is actually an optimal estimator. In this work we show that the so called "state prediction", which is the calculated estimate based on physical evolution of the system, can help us locate our pulse in the antenna we are interested in. To do this we will use the non-linear extension of the Kalman Filter (the "Extended Kalman Filter"), since we are dealing with a non-linear problem. We show by comparison with a least squares estimator that our filter is still close to optimality, even though linearization is used in the Extended Kalman Filter.

Before applying the filter, we will go through the details of the idea behind the filter in a one dimensional case. In the iterative process, there are actually four values to be considered during each time step k:

1. The estimate of the state at time step k-1
2. The predicted value of the state at time step k (from the value at k-1, we calculate the estimate for the next time step k according to the physical laws)
3. The measured value of the state at time step k
4. The combination of the predicted and measured value at time step k

If we consider the "noise" (deviations from the true value by inaccuracy of our measurements) in our measurements to be Gaussian, the property will merge that the procedure is recursive. This means that after we have completed step 4, we can use that estimate for step 1 of our next time step and go through the same process again, which is shown in the flowchart below.

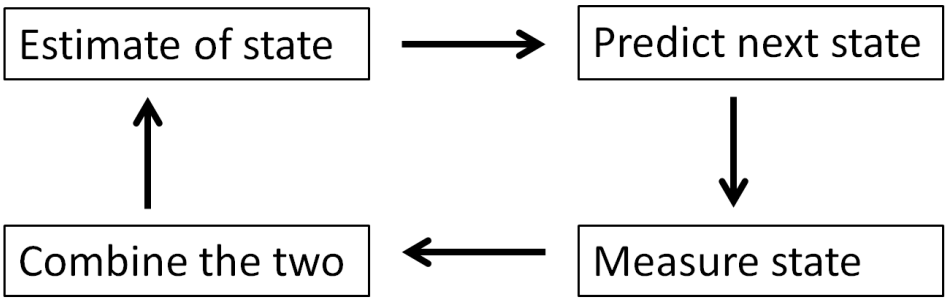

FIGURE 4.1: A flowchart of the Kalman Filter procedure. We start by having a previous estimate of our system, and then calculate what the state will be for our next time step. This is then combined with our measurement which gives our next estimate of the state, and the procedure repeats itself again.

The recursiveness originates from the fact that combining two Gaussian functions gives us another Gaussian function, which simplifies the problem and decreases the computation time. We display the process in figure 4.2 for the example of estimating the location of a cart on a rail.

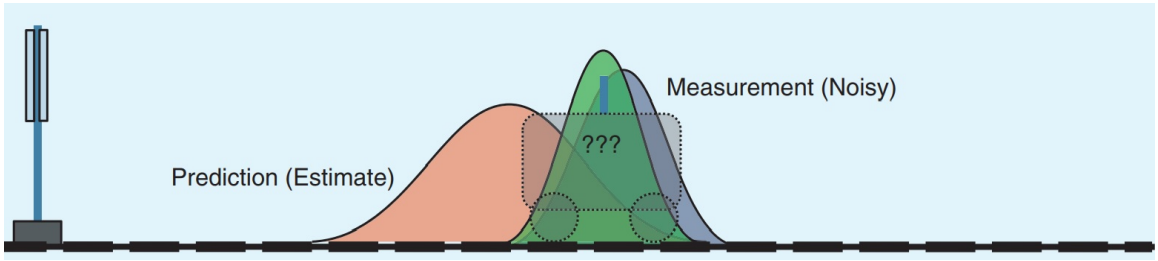

FIGURE 4.2: The predicted location of the cart based on the previous state (location, velocity and acceleration) of the cart (pink), and the measurement of the location of the cart (purple), together with their uncertainties. They are combined into a new update of the location (green), resulting in a better estimate. The figure was taken from [22].

We will now quickly show for the 1D case that the product of two Gaussian functions is again a Gaussian function. The calculation is straightforward, but will help us gain insight into the equations governing the Kalman filter. Consider two Gaussion functions y_1 and y_2, with means μ_1,μ_2 and standard deviations σ_1,σ_2 respectively:

$$y_1 = \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} \quad , \quad y_2 = \frac{1}{\sqrt{2\pi\sigma_2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} .$$

We can combine them into a new probability density function by taking the product $y_1 \cdot y_2$. We get

$$y_{fused} = \frac{1}{\sqrt{(2\pi)^2\sigma_1\sigma_2}} e^{-\frac{(x-\mu_1)^2\sigma_2^2+(x-\mu_2)^2\sigma_1^2}{\sigma_1^2\sigma_2^2}}$$
$$= \frac{1}{\sqrt{(2\pi)^2\sigma_1\sigma_2}} e^{-\frac{(\sigma_1^2+\sigma_2^2)x^2-2(\sigma_2^2\mu_1+\sigma_1^2\mu_2)x+\sigma_2^2\mu_1^2+\sigma_1^2\mu_2^2}{2\sigma_1^2\sigma_2^2}}$$

where we can recognize after a little work that we now have a Gaussian distribution again. The new "fused" mean and standard deviation expressed in terms of the old ones take on the form

$$\mu_{fused} = \mu_1 + \frac{\sigma_1^2(\mu_2-\mu_1)}{\sigma_1^2+\sigma_2^2} \quad , \quad \sigma_{fused} = \sigma_1^2 - \frac{\sigma_1^4}{\sigma_1^2+\sigma_2^2}. \tag{4.1}$$

In this simple case, y_1 could for example correspond to the state prediction (pink estimate in figure 4.2), y_2 to the measurement (purple) and y_{fused} to the estimate combining that data (green). μ_{fused} then gives the best estimate for the position of the cart, given the data.

A notation useful for the next section is

$$\mu_{fused} = \mu_1 + K(\mu_2 - \mu_1) \tag{4.2}$$

where $K \equiv \sigma_1^2/(\sigma_1^2+\sigma_2^2)$ is the "Kalman gain". It tells us how much weight we should give our measurement (μ_2). A high uncertainty in our measurement (σ_2^2 large) would mean $K \to 0$, giving $\mu_{fused} \approx \mu_1$ such that our state prediction was the most valuable to our state estimation. Otherwise if $\sigma_2^2 \to 0$ (highly accurate measurement), $K \approx 1$ and we have $\mu_{fused} \approx \mu_2$ (state estimate equals measurement).

This simple but powerful technique is the basic idea behind the Kalman filter.

4.1 The linear filter

In the following section we will derive the one dimensional linear Kalman Filter, together with applying it in an example. We start by looking at measurements without the filter and then continue with applying the filter. The filter is linear in the sense that it assumes linear dependence of the present state on the previous state. To demonstrate this we will remain at our problem of tracking a cart on a rail, whereby our measurement equipment has some noise to it. We show that the Kalman filter increases our chances of correctly determining the cart's position.

In our example there is a cart on a rail located at $x = 0$. At time $t = 0$ the cart starts to accelerate along the x-axis at a constant rate. We measure the cart's location at specific time intervals with some measurement device, which has some

systematic (Gaussian) noise to the measurement. With this the first 10 seconds of the actual position of the cart, and the measured position of the cart is shown in figure 4.3.

As we can see, due to the large uncertainty in the measurement of our device, we determine the cart to be in the wrong location most of the time. Therefore we are going to introduce the Kalman filter, to improve our estimate of the cart's location.

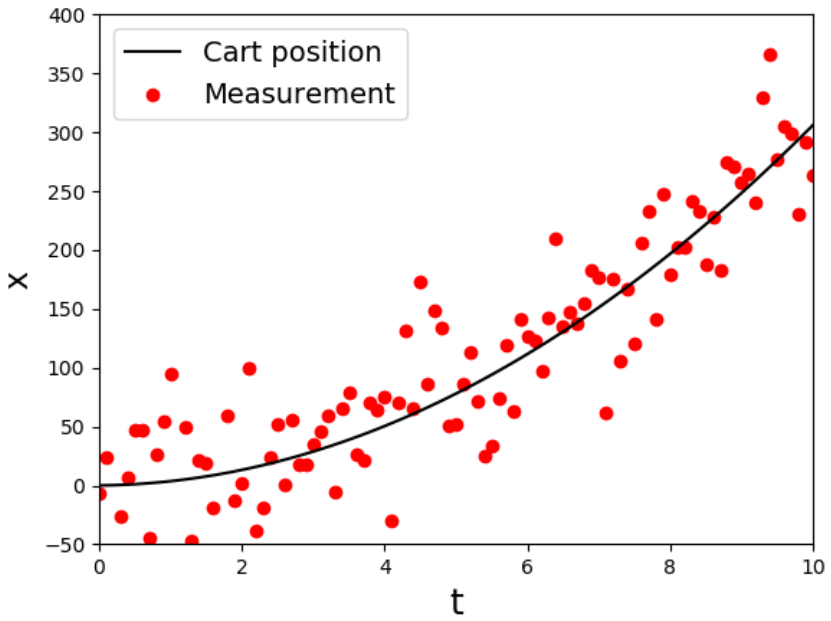

FIGURE 4.3: Indication of the actual cart path (black) and what our device sees over time (red), with the cart accelerating at a constant rate. As we can see our measurement device locates the cart at the wrong place for most of the time. A standard deviation of measurement σ_x=40, acceleration a=6 and time intervals of dt=0.1 were used in the example.

4.1.1 Applying the filter

We are now going to apply a Kalman Filter to improve our chances for determining the actual position of the cart. Therefore we derive the correct form of the equations to apply the Kalman filter to our data. We summarize a few important details:

- the cart has a constant acceleration a
- our device only measures position
- the data noise has a standard normal distribution.

With this information, we can describe the evolution of the path of the cart, based on the physics of objects in motion. We use $\mathbf{x}$ to denote the state vector of the cart,

which now has three components:

$$\mathbf{x} = \begin{pmatrix} \text{position} \\ \text{velocity} \\ \text{acceleration} \end{pmatrix} \tag{4.3}$$

(we only measure position, but we estimate also velocity and acceleration at each step). We distinguish different state vectors for the moment for which the estimate is made, and for which data is available to us for that estimation.

We introduce the notation:

z_k : the measurement of our device at time step k
$\hat{\mathbf{x}}_{k|k-1}$: the state prediction at time step k, using data up to time step k-1
$\hat{\mathbf{x}}_{k|k}$: the best estimate of the state at time step k,
using (measurement and state prediction) data up to time step k

As one can see we use the hat notation ($\hat{\mathbf{x}}$) for the estimations, and use normal boldface ($\mathbf{x}$) for the *true* state vector (which is only "known" to the cart itself).

z_k
Our measurement z_k describes what the we obtain each time we make a measurement, which is just position. Therefore:

$$z_k = \mathbf{H}_k \mathbf{x}_k + \sigma_x \tag{4.4}$$

where σ_x is the uncertainty of our measuring device, $\mathbf{x}_k$ is the true state vector of the cart at time step k and $\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$ maps the state vector parameters into the measurement domain.

$\hat{\mathbf{x}}_{k|k-1}$
The evolution of the state vector (position, velocity, acceleration) for an accelerating object can be described by the simple equations (short derivation in appendix A.1):

$$s_k = s_{k-1} + v_{k-1} \cdot dt + \frac{a_{k-1} \cdot dt^2}{2} \tag{4.5}$$

$$v_k = v_{k-1} + a_{k-1} \cdot dt \tag{4.6}$$

$$a_k = a_{k-1} \tag{4.7}$$

where in our case the time evolution is in steps of time interval dt (where $k \cdot dt = t$). It is then a matter of trivial algebra to obtain the form for the state prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1} \hat{\mathbf{x}}_{k-1|k-1} \tag{4.8}$$

where

$$\mathbf{F}_{k-1} = \begin{pmatrix} 1 & dt & dt^2/2 \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{pmatrix}.$$

Note that we do not take into account here that the acceleration might have noise; we don't have control over the cart and assume the cart has an unflawed acceleration (no "process noise"). The noise according to the state prediction (σ_1 in section 4) is now stored in a 3x3 covariance matrix $\mathbf{P}_{k|k-1}$, since we now have three variables with each its corresponding uncertainty. The evolution of the state prediction (error) covariance matrix $\mathbf{P}_{k|k-1}$ is given by

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{\mathrm{T}} \tag{4.9}$$

and can be derived as follows. We note that for our *true* state vector the following holds:

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1}.$$

The variance of our estimate w.r.t. this true value is then calculated as follows:

$$\begin{aligned}
\mathbf{P}_{k|k-1} &= \mathrm{Var}(\mathbf{x}_k) \\
&= E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T] \\
&= E[\mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^T\mathbf{F}_{k-1}^T] \\
&= E[(\mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^2\mathbf{F}_{k-1}^T] \\
&= \mathbf{F}_{k-1}E[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^2]\mathbf{F}_{k-1}^T \\
&= \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T.
\end{aligned}$$

With this we have completed our state prediction step with the corresponding error covariance, and move on to combine this data with our measurement.

$\hat{\mathbf{x}}_{k|k}$

We now want to fuse the previous data and update our estimate of the state (just as we fused the two estimations in section 4). Our goal is to make a linear combination of the state prediction and the measurement, with each its corresponding weights while minimizing the error covariance of our combined (weighted) estimate (minimizing the mean squared error). Since $\sigma_1^2 \to \mathbf{P}_{k|k-1}$ (state estimate error covariance), the equations are less obvious, but we can still compare them to our derivation in section 4.

The result for the estimation update after fusing the data from (4.8) and (4.4) is:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}) \tag{4.10}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}. \tag{4.11}$$

The new matrix $\mathbf{K}_k$ is the Kalman gain of the system, as the one we already encountered before in a simpler form (to repeat: it tells us how much weight we should give our state prediction relative to our performed measurement). $\mathbf{P}_{k|k}$ is

the updated (combined) covariance of the estimate. The Kalman gain is defined as

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}. \tag{4.12}$$

Here, $\mathbf{R}_k$ is the (error) covariance matrix of the measurement, which in our case is just σ_x^2. The derivations of equations (4.10) to (4.12) are detailed as follows.

As we can see equation (4.10) is of the same form as (4.2), and takes on this specific form as a consequence of two assumptions;

1. We assume that the updated state estimate $\hat{\mathbf{x}}_{k|k}$ is a linear combination of the state prediction $\hat{\mathbf{x}}_{k|k-1}$ and the observed measurement z_k
2. We assume an unbiased error $E[\epsilon] = 0$ where ϵ is the error in our estimate

From assumption 1. we write

$$\hat{\mathbf{x}}_{k|k} = \mathbf{A}_k\hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k z_k \tag{4.13}$$

with $\mathbf{A}_k, \mathbf{K}_k$ to be determined. Using assumption 2. we continue by looking at the expected deviation ϵ from the true state, which is expected to be zero for all cases: $E[\epsilon_{j|k}] = 0\ \forall j,k$ where $\epsilon_{j|k} \equiv \mathbf{x}_j - \hat{\mathbf{x}}_{j|k}$, which we can expand as follows:

$$\begin{aligned}
\epsilon_{k|k} &= \mathbf{x}_k - \hat{\mathbf{x}}_{k|k} \\
&= \mathbf{x}_k - \mathbf{A}_k\hat{\mathbf{x}}_{k|k-1} - \mathbf{K}_k z_k \\
&= \mathbf{x}_k - \mathbf{A}_k(\mathbf{x}_k - \epsilon_{k|k-1}) - \mathbf{K}_k(\mathbf{H}_k\mathbf{x}_k + \sigma_x) \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k - \mathbf{A}_k)\mathbf{x}_k - \mathbf{A}_k\epsilon_{k|k-1} - \mathbf{K}_k\sigma_x.
\end{aligned}$$

We then compute the expectation value of this, and set it equal to zero:

$$E[\epsilon_{k|k}] = E[((\mathbf{I} - \mathbf{K}_k\mathbf{H}_k - \mathbf{A}_k)\mathbf{x}_k)] - \mathbf{A}_k E[\epsilon_{k|k-1}] - \mathbf{K}_k E[\sigma_x] = 0.$$

From our unbiased assumption, $E[\epsilon_{j|k}] = 0$ and $E[\sigma_x] = 0\ \forall j,k$, leading to the result that

$$\begin{aligned}
&(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k - \mathbf{A}_k)E[\mathbf{x}_k] = 0 \quad \forall k \\
&\Rightarrow \mathbf{A}_k = \mathbf{I} - \mathbf{K}_k\mathbf{H}_k.
\end{aligned}$$

If we substitute this result into our assumption (4.13) we arrive at the proper form of (4.10). We are then left with determining $\mathbf{P}_{k|k}$and $\mathbf{K}_k$. We first calculate $\mathbf{P}_{k|k}$, under the assumption of our linear estimate (4.10) which we've just calculated:

$$\begin{aligned}
\mathbf{P}_{k|k} &= E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^T] \\
&= E[(\mathbf{x}_k - (\hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})))(\mathbf{x}_k - (\hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})))^T] \\
&= E[((\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) - \mathbf{K}_k\sigma_x)^2] \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^2](\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T - \mathbf{K}_k E[\sigma_x^2]\mathbf{K}_k \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T - \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k
\end{aligned}$$

This is not yet our familiar form of equation (4.11), but it will simplify as we get the expression for the Kalman gain. This we get by minimizing the mean squared error of our best state estimate, with respect to the Kalman gain $\mathbf{K}_k$ (which is still a variable). The trace of $\mathbf{P}_{k|k}$ is the sum of the mean squared errors, hence we take the matrix derivative of the trace of $\mathbf{P}_{k|k}$ w.r.t. $\mathbf{K}_k$ and set it equal to zero to find our optimal $\mathbf{K}_k$.

$$\begin{aligned}\mathrm{Tr}[\mathbf{P}_{k|k}] &= \mathrm{Tr}[\mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1} - \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T] \\ &\overset{1}{=} \mathrm{Tr}[\mathbf{P}_{k|k-1}] - 2\mathrm{Tr}[\mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}] - \mathrm{Tr}[\mathbf{K}_k(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)\mathbf{K}_k^T]\end{aligned}$$

Taking the derivative of this and setting it to zero for the optimal $\mathbf{K}_k$:

$$\begin{aligned}\frac{\mathrm{dTr}[\mathbf{P}_{k|k}]}{\mathrm{d}\mathbf{K}_k} &= -2(\mathbf{H}_k\mathbf{P}_{k|k-1})^T + 2\mathbf{K}_k(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k + \mathbf{R}_k) = 0 \\ \Rightarrow \mathbf{K}_k &= \mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k + \mathbf{R}_k)^{-1}\end{aligned}$$

hence we have arrived at the result of (4.12). The final step is to update $\mathbf{P}_{k|k}$ with this result for the optimal gain:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \quad \text{(optimal gain)}$$

which is identical to (4.11).

With these equations we are ready to apply the filter to our data. One example is plotted in figure 4.4, where we can directly see that the Kalman Filter improves our estimation of where the cart is at each time step. The corresponding program code can be found in appendix B.1.

[1] Here we have used Tr[AB]=Tr[$(AB)^T$]

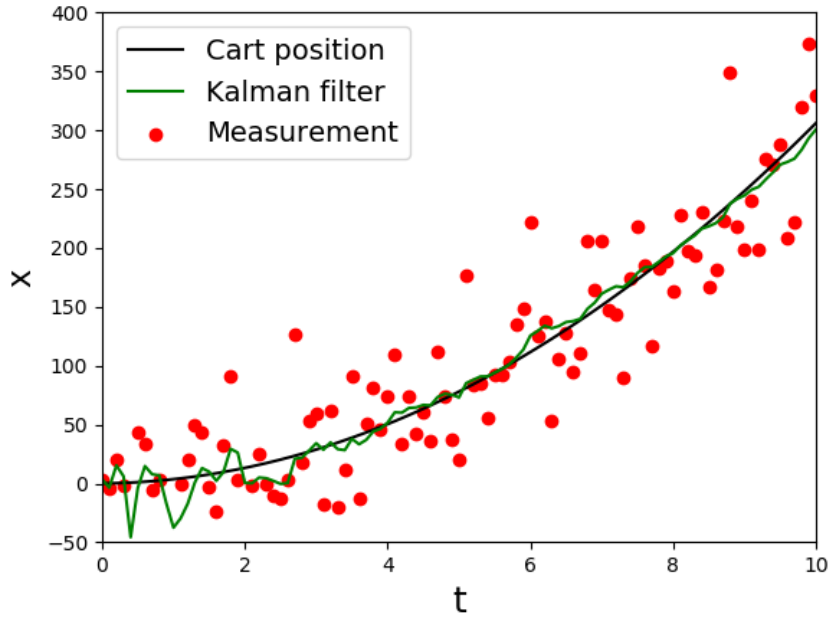

FIGURE 4.4: The Kalman Filter applied to our problem of cart tracking. As we can see the Kalman Filter improves our measurements as expected. As we take more data points the estimate improves as well. Again we have used values of $\sigma_x = 40$, a=6 and $dt = 0.1$.

4.1.2 Two identical carts

In extension to the case of tracking one cart, we have also looked at the case in which we want to track two identical carts (we can not distinguish which cart is which each time we make a measurement). This was done with upcoming challenges in mind in the imaging of lightning (whereby we would like to track two identical pulses at the same time). Unfortunately, there was not enough time to implement this model into the lightning research (yet). Therefore this section can be skipped without loss of continuity.

A model was made based on assigning weights to each observation, where the weights were determined in the following way. At each time step we have two predictions of where the two carts should be (a, b), and two measurements of where the two carts are (1,2). We are going to assign the predictions done by the Kalman Filter (a, b) of each of the two carts, to the two measurements of the carts (1,2), by weights. The measurements (1,2) always have, as stated in the problem, a constant Gaussian error distribution, given by their standard deviation of measurement σ_x, as shown in figure 4.5. The weights are then determined from the area under the Gaussian curves, which are integrated from the measurements to the state predictions. This technique is able to handle every possible ordering of where the predictions and measurements are.

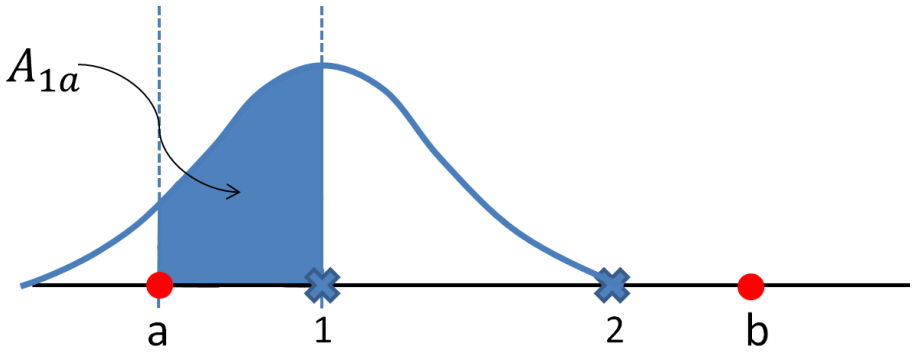

FIGURE 4.5: Two measurements 1,2 and two predictions a, b for where the carts should be. The weights are calculated from the areas A_{ik} under the Gaussian curve centered on the measurement with standard deviation σ_x, where the integration is done up to the the state predictions. This is shown here for the case i=1 and k=a, but the same is done for all i=1,2 and k=a, b.

We integrate all possibilities A_{1a}, A_{1b}, A_{2a}, A_{2b} where A_{1a} thus denotes the area under the curve between measurement 1 and prediction a. We then calculate the probability from measurement 1, that measurement 1 belongs to prediction a the following way:

$$P_{1a}^{(1)} = \frac{0.5 - A_{1a}}{0.5 - A_{1a} + 0.5 - A_{2a}} \tag{4.14}$$

where the superscript denotes that this takes into account only the information of measurement 1. This is of course equivalent to the probability that the prediction of cart b belongs to measurement 2 (based *only* on measurement 1), since every cart prediction has to be assigned to either one of the measurements. Hence automatically: $P_{1b}^{(1)} = P_{2a}^{(1)} = 1 - P_{1a}^{(1)}$ We do the similar calculation for measurement 2, hence we have all probabilities $P_{ik}^{(l)}$, where l=1,2, i=1,2 and k=a, b. The final weights given in this approach are then calculated the following way:

$$w_{ik} = \frac{P_{ik}^{(1)} + P_{ik}^{(2)}}{P_{ia}^{(1)} + P_{ib}^{(1)} + P_{ia}^{(2)} + P_{ib}^{(2)}} = \frac{P_{ik}^{(1)} + P_{ik}^{(2)}}{2}. \tag{4.15}$$

This means w_{1a} then gives the final weight for measurement 1 to belong to cart a, considering all information of measurements 1,2 and predictions a, b. The measurements are then assigned as a weighted average in the update step of the Kalman Filter (4.10) in the following way:

$$\hat{\mathbf{x}}_{k|k}^{a} = \hat{\mathbf{x}}_{k|k-1}^{a} + \mathbf{K}_k(w_{1a}z_k^1 + w_{2a}z_k^2 - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}^{a}) \tag{4.16}$$

$$\hat{\mathbf{x}}_{k|k}^{b} = \hat{\mathbf{x}}_{k|k-1}^{b} + \mathbf{K}_k(w_{1b}z_k^1 + w_{2b}z_k^2 - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}^{b}) \tag{4.17}$$

where $\hat{\mathbf{x}}_{k|k}^{a,b}$ is the best estimate for cart a, b, and $z_k^{1,2}$ are the measurements 1,2. We can simplify the equation by noting that for two carts, $w_{2a} = w_{1b} = 1 - w_{1a}$ and

$w_{2b} = w_{1a}$. If we define $w_{1a} \equiv W$, this results in

$$\hat{\mathbf{x}}^a_{k|k} = \hat{\mathbf{x}}^a_{k|k-1} + \mathbf{K}_k(Wz^1_k + (1-W)z^2_k - \mathbf{H}_k\hat{\mathbf{x}}^a_{k|k-1}) \tag{4.18}$$

$$\hat{\mathbf{x}}^b_{k|k} = \hat{\mathbf{x}}^b_{k|k-1} + \mathbf{K}_k((1-W)z^1_k + Wz^2_k - \mathbf{H}_k\hat{\mathbf{x}}^b_{k|k-1}). \tag{4.19}$$

This process is repeated at each time step. An example of where this approach is applied is shown in figure 4.6. Here two carts which have different accelerations are tracked simultaneously.

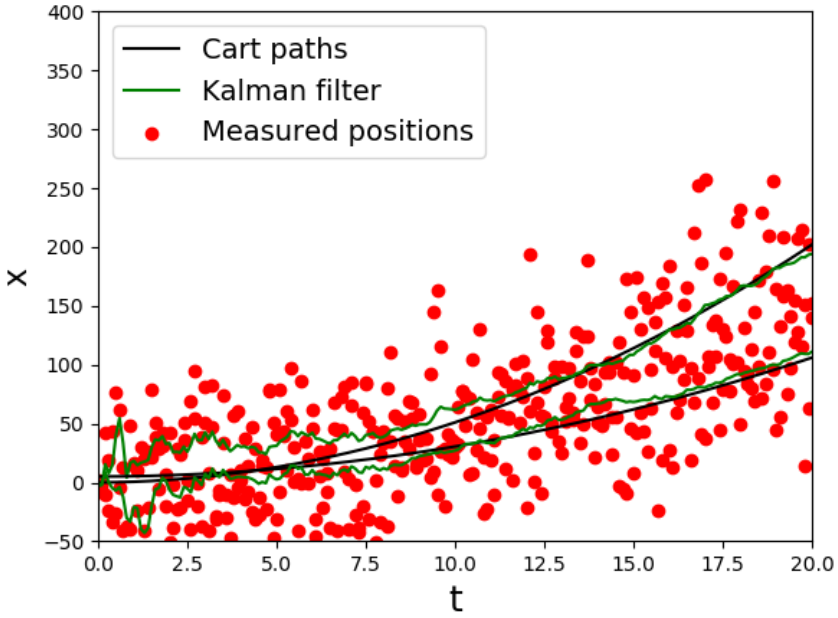

FIGURE 4.6: Two identical carts with different accelerations are tracked simultaneously. Again $\sigma_x = 40$ and $dt = 0.1$ are used. The carts have close starting points and small differences in acceleration which makes it harder have a clear identification of which measurement belongs to which cart. As one can see, our algorithm is able to track the carts even in this difficult situation.

This result is of interest for the case of imaging lightning, but still has to be adjusted slightly. The method used here depends on the standard deviation of the measurement (determining A_{ik}), but determining this in our case of lightning imaging (pulse tracking) turned out to be non-trivial, and still requires some work.

4.2 The non-linear filter

In the previous section the Kalman filter was discussed in the case of a linear model. Most systems in physics and engineering however, are described by non-linear dynamics. Also in our case of studying a lightning flash, we will encounter non-linearity. Therefore, in the next section, we will derive the "Extended Kalman Filter" (EKF), which can also be applied to non-linear systems. The idea remains

exactly the same, and we will effectively only change two equations.

As we extend the filter to non-linear systems, the equation describing the exact evolution of the state vector is now a non-linear function $\mathbf{f}$ of the previous state vector $\mathbf{x}_{k-1}$:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}). \tag{4.20}$$

In this extension however, the non-linear function (evolution according to the laws of physics) is also working on the (Gaussian) error covariance. Because a non-linear function multiplied with a Gaussian distribution does not yield a Gaussian distribution, the Gaussianity is not preserved. This was the basis for the recursiveness of our Kalman filter and hence something we would like to preserve. Therefore, we linearize the non-linear function by a first order Taylor expansion, and use the linearization to approximate the state prediction and the corresponding error propagation. This is only necessary for the state prediction part, since in our case we still have a linear measurement function (a derivation can also be obtained for non-linear measurement functions, see [23]).

State prediction

We first note that in general for our state prediction, the following (substituting (4.20)) holds:

$$\hat{\mathbf{x}}_{k|k-1} = E[\mathbf{x}_k|z_{k-1}] \tag{4.21}$$
$$= E[\mathbf{f}(\mathbf{x}_{k-1})|z_{k-1}]. \tag{4.22}$$

Since we do not know the true value of the state $\mathbf{x}_{k-1}$, we calculate the expectation value using the following Taylor expansion:

$$\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + \frac{d\mathbf{f}}{d\mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}) \tag{4.23}$$

where we have expanded in the point $\hat{\mathbf{x}}_{k-1|k-1}$, our best estimate for the previous state, and higher order terms are considered negligible. Our state prediction then becomes

$$\hat{\mathbf{x}}_{k|k-1} = E[\mathbf{f}(\mathbf{x}_{k-1})|z_{k-1}] \tag{4.24}$$
$$\approx E[\mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + \frac{d\mathbf{f}}{d\mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})|z_{k-1}]. \tag{4.25}$$

Since the expectation value of the difference between the state prediction and the true state is equal to zero, the second term of the Taylor expansion drops out ($E[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})|z_{k-1}] = 0$), and we arrive at the following equation for our state prediction:

$$\hat{\mathbf{x}}_{k|k-1} \approx E[\mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1})|z_{k-1}] \tag{4.26}$$
$$= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}). \tag{4.27}$$

This is a simple and intuitive result; we can just use the general non-linear function on our previous best estimate to calculate the state prediction. We do however make a small error by neglecting the higher order terms, but this is considered manageable for functions which are not highly non-linear since the contribution of the first-order term of the Taylor Expansion vanishes.

With this result we can also derive the predicted error covariance. The error covariance is defined to be the variance between the state prediction and the true state:

$$\mathbf{P}_{k|k-1} = E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^T] \tag{4.28}$$
$$= E[(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}))(\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}))^T]. \tag{4.29}$$

To express this in values known to us we use the same Taylor expansion as we did in (4.23) for $\mathbf{f}(\mathbf{x}_{k-1})$, such that the term in brackets becomes

$$\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) + \mathbf{J}_f(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}) - \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \tag{4.30}$$
$$= \mathbf{J}_f(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}). \tag{4.31}$$

Here we have used shorthand notation $\mathbf{J}_f$ for the Jacobian of $\mathbf{f}$: $\mathbf{J}_f = \frac{d\mathbf{f}}{d\mathbf{x}}\Big|_{\hat{\mathbf{x}}_{k-1|k-1}}$. Equation (4.29) then becomes

$$\mathbf{P}_{k|k-1} \approx \mathbf{J}_f E[(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1})^T]\mathbf{J}_f^T \tag{4.32}$$
$$= \mathbf{J}_f \mathbf{P}_{k-1|k-1}\mathbf{J}_f^T \tag{4.33}$$

which is similar to our result for the linear Kalman filter, only now we have replaced the evolution operators with Jacobians evaluated in the previous best estimate. Of course in the case that our function $\mathbf{f}(\mathbf{x}_{k-1})$ becomes linear, we obtain the same results as in chapter 4.1.

Data fusion

After we have obtained the measurement result z_k with the corresponding covariance $\mathbf{R}_k$, we can piece together again our best estimate $\hat{\mathbf{x}}_{k|k}$. The procedure from here is for our case identical to the one treated in chapter 4.1, and therefore will not be treated here again. There are however extensions to the Kalman filter which make use of non-linear measurement functions, but in our case $\mathbf{H}_k$, which maps the state vector to the measurement domain, is a linear function. Therefore we have the same result for the best estimate $\hat{\mathbf{x}}_{k|k}$, the Kalman gain $\mathbf{K}_k$ and the best estimate error covariance $\mathbf{P}_{k|k}$ as before:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}) \tag{4.34}$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{4.35}$$
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{4.36}$$

4.2.1 EKF simulation: spherical wave emitted from point source

In summary, we have derived the following equations for our non-linear Kalman filter, for the processes of state prediction and data fusion.

State prediction step:

$$\hat{\mathbf{x}}_{k|k-1} \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \tag{4.37}$$

$$\mathbf{P}_{k|k-1} \approx \mathbf{J}_f \mathbf{P}_{k-1|k-1} \mathbf{J}_f^T. \tag{4.38}$$

Data fusion step:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(z_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \tag{4.39}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{4.40}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}. \tag{4.41}$$

These equations will be implemented for the case which we're interested in, that is, the example of a point source emitting a spherical radio wave, which is measured at different locations in space and time by our antennas (figure 4.7). The source is located at x_0, y_0, z_0 in space and emits the radio pulse at time t_0. These parameters are unknown to us, and we want to estimate them using the Extended Kalman Filter. In theory to obtain the location of the source, we need at least four antennas who have recorded the arrival time of the pulse with infinite precision. In reality we need a lot more to come to a precise location estimate, since every pulse has some width and uncertainty to it. Therefore the antennas record the time of arrival of the signal with some Gaussian noise. With those measurements and our model for the wave propagation we can apply the EKF and estimate the source location.

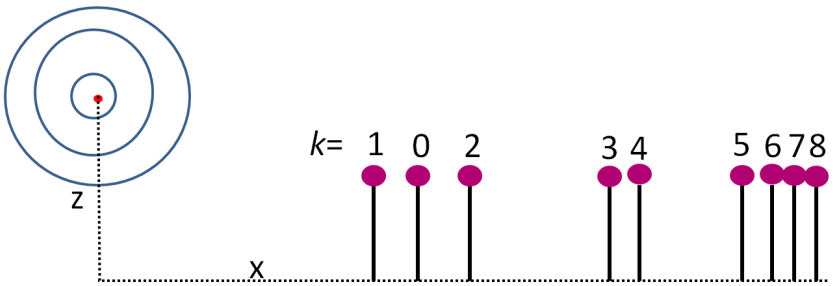

FIGURE 4.7: A visual (2D) representation of the model used in our simulation. A point source emits a spherical wave which is observed at different locations and times by different antennas. Take into account that instead of iterating over the time intervals, the EKF now iterates over antenna number k! Here we have chosen a specific antenna to be our first antenna (the zero antenna), and order antennas by means of their distance to our first antenna.

We start with an initial guess of the state vector, where the state vector now takes

on the form

$$\mathbf{x}_k = \begin{pmatrix} t_k \\ x_0 \\ y_0 \\ z_0 \end{pmatrix}. \tag{4.42}$$

where t_k is the time of arrival of the signal in antenna k. For our first antenna this is taken to be the moment in time at which the peak of the pulse (which we have decided to map) is located. The coordinates x_0, y_0, z_0 are initially estimated by an independent lightning detection system, and is a crude estimate. In principle if the location of the source is known, one can of course exactly calculate t_k from x_0, y_0, z_0, the coordinates of antenna k, and the speed of light in the medium. To us, only the antenna coordinates and the time of arrival in antenna k is known. Therefore we estimate x_0, y_0, z_0 and at each step hope to improve the accuracy by means of the EKF.

Our EKF demands that we model our system such that it is dependent only on our previous state vector $\mathbf{x}_{k-1}$. This means that for our first entry of the state vector, the following relationship holds:

$$t_k = t_{k-1} + \frac{|\vec{r}_k| - |\vec{r}_{k-1}|}{c_{air}} \tag{4.43}$$

where

$$|\vec{r}_k| = \sqrt{(x_k - x_0)^2 + (y_k - y_0)^2 + (z_k - z_0)^2}. \tag{4.44}$$

The coordinates x_0, y_0, z_0 are used from the $k-1$ estimate (k is antenna number). They only get updated after data fusion (equation (4.39)). This means we can write the following non-linear function for our state prediction:

$$\mathbf{x}_k = \begin{pmatrix} t_k \\ x_{0,k} \\ y_{0,k} \\ z_{0,k} \end{pmatrix} = \mathbf{f}(\mathbf{x}_{k-1}) \tag{4.45}$$

$$= \begin{pmatrix} t_{k-1} + \frac{\sqrt{(\vec{x}_k - \vec{x}_{0,k-1})^2} - \sqrt{(\vec{x}_{k-1} - \vec{x}_{0,k-1})^2}}{c_{air}} \\ x_{0,k-1} \\ y_{0,k-1} \\ z_{0,k-1} \end{pmatrix}. \tag{4.46}$$

If we also want to calculate our predicted covariance, we need the Jacobian $\mathbf{J}_f$:

$$\mathbf{J}_f = \frac{d\mathbf{f}}{d\mathbf{x}_{k-1}} = \begin{pmatrix} 1 & \frac{\partial f_t}{\partial x_{0,k-1}} & \frac{\partial f_t}{\partial y_{0,k-1}} & \frac{\partial f_t}{\partial z_{0,k-1}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.47}$$

where

$$\frac{\partial f_t}{\partial x_i} = \frac{(x_i)_{0,k-1} - (x_i)_k}{\sqrt{(\vec{x}_k - \vec{x}_{0,k-1})^2}} - \frac{(x_i)_{0,k-1} - (x_i)_{k-1}}{\sqrt{(\vec{x}_k - \vec{x}_{0,k-1})^2}} \tag{4.48}$$

and $x_i = x, y, z$.

This is enough to apply the EKF. Following the steps from equation (4.37) to (4.41), we can get the following result (figure 4.8) for a simulated situation. Corresponding code can be found in appendix B.2.

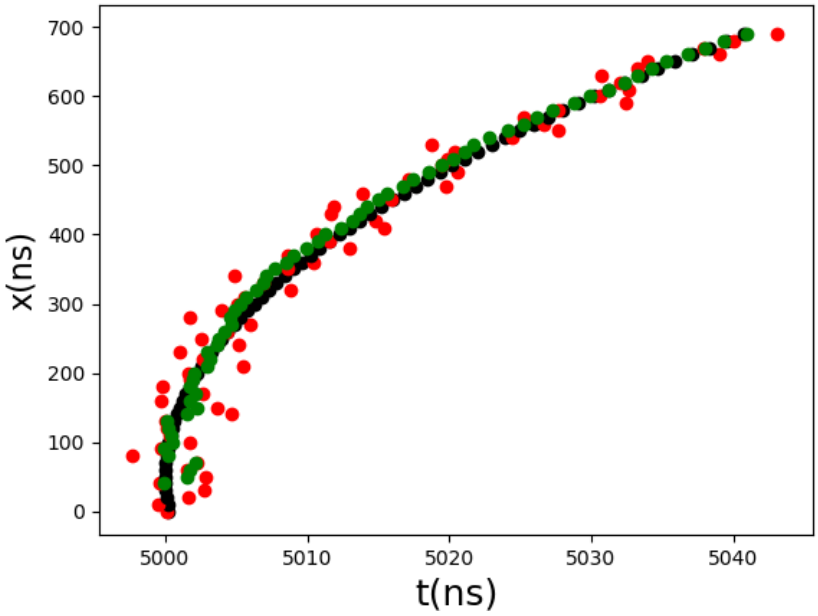

FIGURE 4.8: The extended Kalman filter applied to a radio source emitting a spherical wave, observed by a 1D array of antennas. Again the actual times of arrival are plotted in black, actual observations (with noise) in red and the Kalman filter in green. Units s.t. c=1 are used.

4.3 Methods used with LOFAR data

With the results from section 4.2.1, we are now ready to apply our algorithm to real data. The data is provided by the LOFAR telescope, and processed by the methods mentioned in section 3.1. We already have the body of our filter complete, as provided in the previous section, but we are going to use a different strategy than before. We are going to make use of the fact that the Kalman Filter's state prediction gives an estimate of where the pulse we are trying to map is located in the next antenna. This estimate will be shown to be very accurate, which means the window in which we search for the pulse in the next antenna with this estimate is considerably smaller than the window corresponding to the analysis of the maximum travel time to the next antenna by the pulse, based on geometrical

considerations and the speed of light in the medium. As a result we are less probable of making an error, can map a higher density spectrum of pulses and also the efficiency of our algorithm increases.

We start by loading and processing the data of our first antenna, for the specified block. In this block we then choose a specific event/pulse which we want to map. In this case this was chosen to be the pulse in the block which had the highest peak. The peak is then "*windowed*", which means a specified number of data points are taken from intervals directly above and below the peak. We then mark the maximum of the peak as the time of arrival of the pulse in the first antenna. This location in time for the first pulse is entered into the initial state vector $(\mathbf{x}_k = (t_k, x_{0,k}, y_{0,k}, z_{0,k}))$, where the initial guess of the location x_0, y_0, z_0 is given by the general location of the flash, found by a different lightning detection system (which is a crude estimate).

Then the next thing to be determined is the arrival time of the pulse in the next antenna on the list. Recall that the antennas are ordered by distance from the first antenna, such that the next antenna is the antenna which is closest to the previous one. For determining the arrival time of the signal in the next antenna we are going to use the method of "*cross-correlation*"(CC), which, as stated nicely by Wikipedia, "*is a measure of similarity between two series as a function of their replacement with respect to each other*" [24]. It is represented mathematically as

$$h(\tau) = f \otimes g = \int_{-\infty}^{+\infty} \bar{f}(t) g(t+\tau) dt \tag{4.49}$$

where $\bar{f}$ denotes the complex conjugate of f. From the function $h(\tau)$, the time delay of the functions f, g w.r.t. each other can be deduced. A simple example is given in figure 4.9, where we computed an example of the cross correlation of two similar functions, whereby one is displaced in time w.r.t. the other, and also the amplitudes are varied slightly.

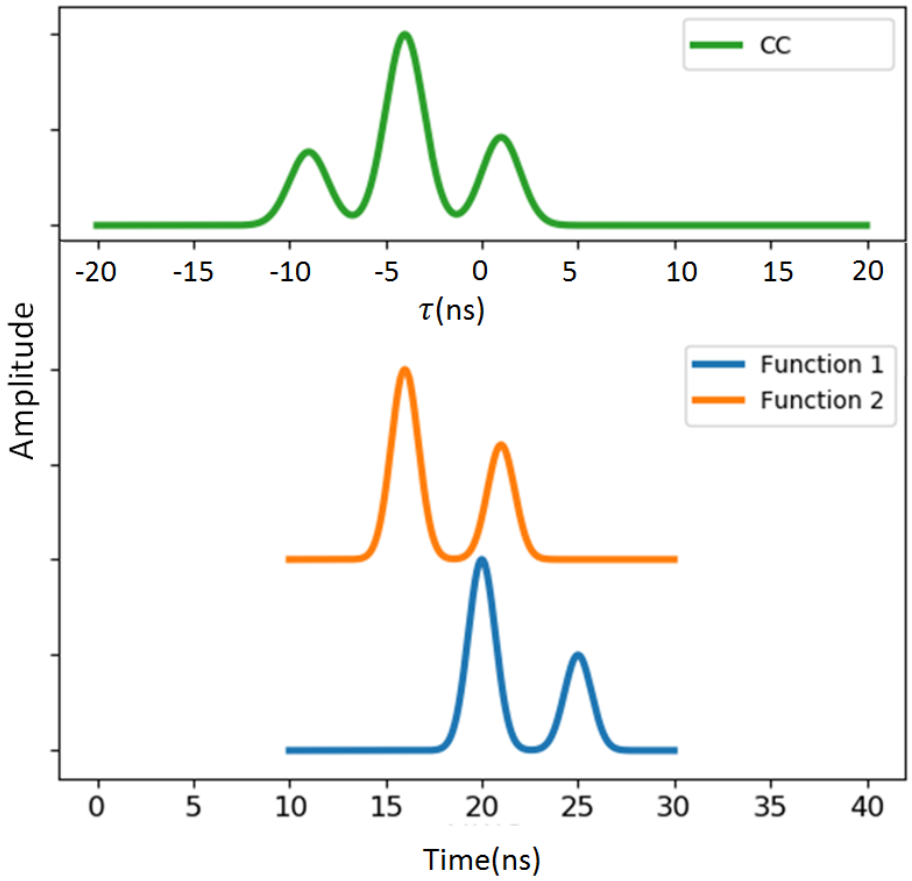

FIGURE 4.9: Two similar functions (blue,yellow) with their cross-correlation (green) plotted above each other. As we "slide" the blue function either in left or right direction over the yellow one, we can see peaks in the green function corresponding to the peaks of the two functions being multiplied there. The central peak of the CC is at the time at which the functions are maximally overlapping with each other. From this, the time delay of the signals w.r.t. each other can be deduced.

As one can see from figure 4.9, the maximum value of the cross correlation is the (negative) time delay of signal 2 w.r.t. signal 1. The two peaks besides the central one in the CC are from the small peaks overlapping with the larger one in the other signal. When the large and small blue peaks are simultaneously aligned with the large and small yellow peaks the cross correlation achieves its maximum, from where we extract the time delay between the signals.

The arrival time of the pulse in the next antenna on the list can then be obtained in the following way. Suppose the blue function in figure 4.9 corresponds to the pulse we are trying to map, in the first antenna. The arrival time of the signal of this pulse in the first antenna, together with our initial (crude) estimate of the location of the event, determines the first state estimate of the Kalman Filter. With this we can perform the state prediction step of the filter, which gives predicted arrival time t_{pred} of the blue pulse in the next antenna on the list. As mentioned before, we can take data points around this value, whereby see if we can recognize the pulse in the neighbourhood of this prediction. This window in which we

search is the yellow pulse plotted in figure 4.9. This window is centered around t_{pred}, but since this predicted value by the Kalman Filter is almost never exactly aligned with the actual value of the arrival time of the pulse, we have to obtain the delay (which can be negative) w.r.t. this prediction. Here we use the method of cross-correlation on our two windowed pulses, which is represented by the green curve. The location in time of the peak of the cross-correlation (around t=-4 ns) is taken to be the time delay w.r.t. the Kalman Filter prediction. This we will denote with Δt_{CC}, and is negative in figure 4.9. It is then a trivial calculation to obtain the location in real time of the pulse in the desired antenna k:

$$t_k = t_{\text{pred}} + \Delta t_{\text{CC}}. \tag{4.50}$$

This is how the measured arrival time of our pulse in antenna k is determined in our algorithm. A pulse which has been recorded by LOFAR and to which this procedure is applied to is plotted in figure 4.10.

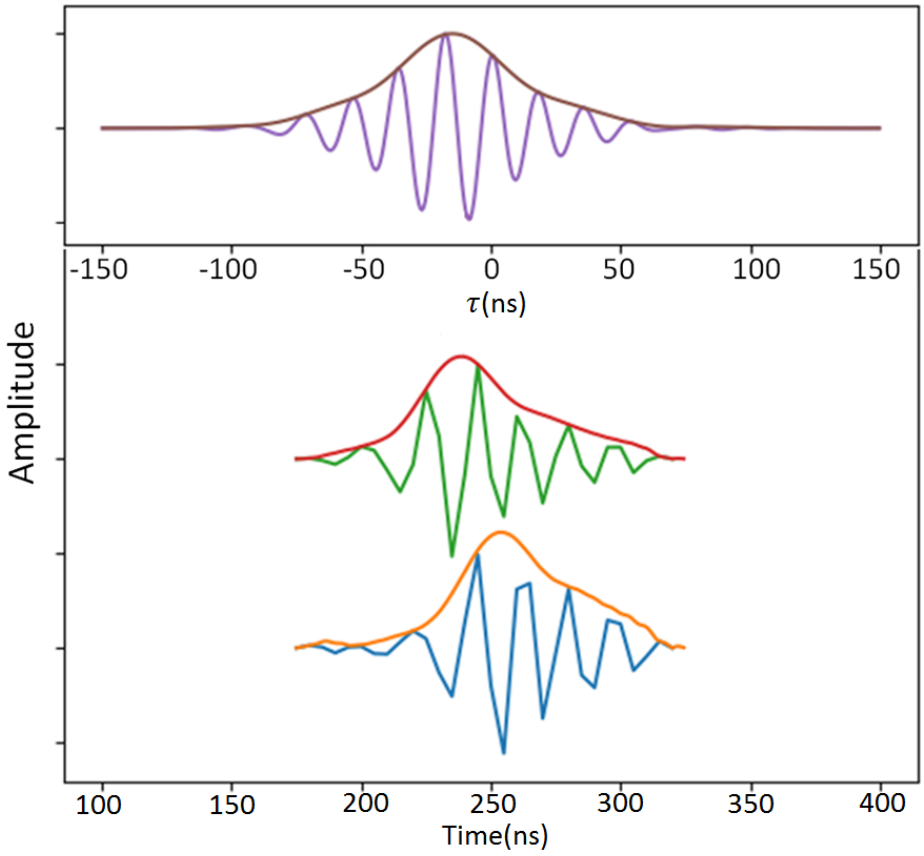

FIGURE 4.10: An example of a recorded pulse which has been taken out of the spectrum from the first antenna (blue/yellow), and cross correlated (purple/brown) with the window taken out of the spectrum of the second antenna (green/red), which is centered around the Kalman Filter estimate. As one can see, the prediction is accurate to about 10 ns, even though this was done on the initial crude estimate of the source location. Also, the shapes are very similar even though two spatially separate antennas recorded them. This happens also for more remote antennas which can be around 80 km from the initial one. The shapes are usually not exact Gaussians but have some deformations to it. If these deformations get too severe, the cross-correlation becomes more unreliable.

As one can see from figure 4.10, the time of arrival predicted by the Kalman Filter differs from the actual location by only about 10 ns. This was done based on the crude estimate given by the lightning detection system, but still proves to be accurate. As more data from antennas is collected, this estimate tends to become even more accurate. If we compare this to the width of the pulses, which are usually around 50-80 ns, we see that our estimate predicted by the Kalman Filter is accurate to within the width of a pulse, even for a crude estimation of the source location. This means that pulses can be mapped individually even for dense spectra. If we compare this with existing methods, the inter-pulse separation needs to be 150 ns (at best) for the method to be able to image them.

One problem which does remain to be solved however, is the case in which the shapes of the pulses are not as "clean" as the example shown in figure 4.10. What

is meant by this is that there are spectra for which the windowed spectrum actually contains two pulses which are very close or even overlapping (see figure 4.11 for an example). Here the first pulse on the bottom is overlapping with another pulse and the pulse above that even has three pulses in this small window. These pulses can change order when observed in another antenna, which leads to the problem of identifying which pulse belongs to which (sometimes the amplitudes also vary as we go over different antennas' spectra, which complicate things even more). This is not taken into account in this work, and is left to be investigated in future research.

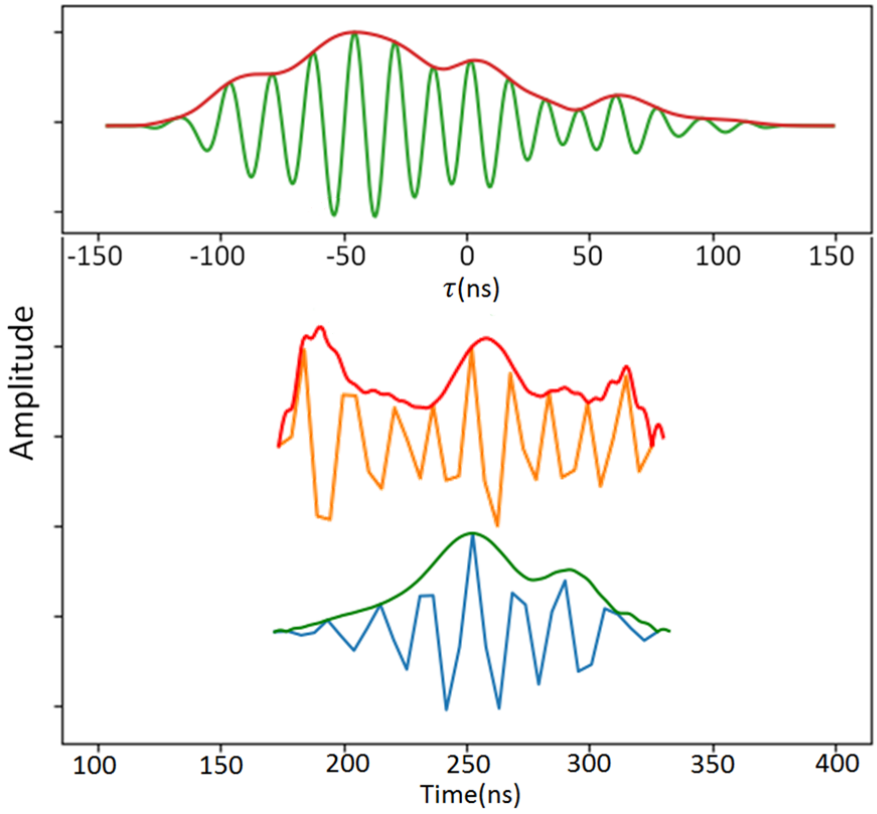

FIGURE 4.11: Two windows of spectrum which were found by the same method earlier, only now from part of the spectrum where the pulse density reaches its maximum. The cross-correlation is also plotted, but determining the delay by means of the peak of the cross-correlation shows to be unreliable here. These spectra are also not yet handled in this work, but may be looked at in future work. The wiggling of the peaks on the left and right of the red curve are an artifact of the Hann-window with which the spectrum was multiplied with, and can be dealt with.

For the pulse shown at the bottom of figure 4.11, we can see that there is one additional "bump" next to our peak in the center. This second bump, usually around half the amplitude of the one we were interested in, was in most cases (not figure 4.11) observed in all antennas to stay relatively at the same place w.r.t. the large peak, in all antennas. If this distance between bump and peak did in- or decrease, it resulted in the timing of the peak of the cross-correlation not to be exactly aligned with the peak which we were interested in, and affected the measured arrival time. In these cases, larger deviations in RMS values were found, but they

were still used in this research if the shapes did not vary significantly. RMS values were calculated to check how well the final location fitted the time of arrival of the pulse in all antennas. The RMS was calculated the following way:

$$RMS = \sqrt{\sum_{k} \frac{(t_{k,modeled} - t_{k,observed})^2}{N}} \tag{4.51}$$

where the sum is over antenna number and N is the total number of antennas. The modeled arrival time is the expected time of arrival of the pulse based on the final location in space and time estimated by the algorithm. For clean pulses this value was close to 1-2 nanoseconds. Pulses which had deviating structures (e.g. the peak with the extra bump in figure 4.11) were found to have an RMS value closer to 7-10 nanoseconds, whereby intermediate pulses showed intermediate values.

Chapter 5
Results

Several pulses in different blocks were analyzed whereby the Kalman Filter was also compared with a non-linear fitter. The non-linear fitter used was the Levenberg-Marquardt (LM) algorithm, which is a least-squares curve fitter. They were compared by means of an RMS test and final location estimate. The observed arrival times of the pulse determined by the cross-correlation method (equation (4.50)) was also used for the LM fitting.

Differences in RMS of the Kalman Filter w.r.t. the non-linear fitter varied from 0.01 ns to 0.1 ns. The differences in final location of the event varied from around one meter to several meters. When inspecting these differences closer, we see that the high end of these differences was observed when there were anomalies in the pulse shapes, at least for some of the antennas. Nevertheless, for clean pulses with no overlap the filter was shown to be close to optimality.

Probably of most value to current research, is the surprising accuracy of the predicted value done by the Kalman Filter. In the worst case where the location of the source is yet a crude estimate, it proves to predict the (time) location of the pulse accurate to a few tens of nanoseconds (worst case). Aside from that, it was shown to be accurate to less than 10 nanoseconds in about 98% of the antennas investigated, which is a great improvement w.r.t. current imaging methods.

Chapter 6
Conclusions

In this thesis we investigated if a Kalman Filter could help to improve current lightning imaging. We have shown that the predictive step in the Kalman Filter procedure surpasses current imaging techniques in accuracy. It promises to image high-density pulse emissions from lightning which have never been imaged before in this detail. Although there is still work to be done, for example on the cases of deviating pulse structures, this is an invigorating result, hopefully leading to new discoveries in the field.

The highest density spectra with a lot of pulses in a short time-window remains a challenge. Also this research looks out on that. With the methods from 4.1.2 it is potentially possible to double the amount of pulses which can be tracked simultaneously w.r.t. this research. There was however not enough time to include this in this paper, but hopefully this will also get a place in a report in the future.

Chapter 7
Acknowledgements

As for all research, this research was not done by myself alone. I am very grateful to my supervisor Olaf Scholten who was always available whenever I needed him, even though his schedules were busy. Discussions together were of crucial importance to this work. I would also like to thank my other colleague Brian Hare, who has also helped me out in many ways. Also without his help this work would not have been possible. Furthermore I would like to thank Johan Messchendorp for being the second assessor of my report. Finally, a word of thanks goes out to my friends, family and beautiful girlfriend for pulling my head out of the clouds sometimes.

Appendix A
Derivations

A.1 State vector evolution

We consider evolution of each of the state vector elements of the constantly accelerating cart, at each time step. We want our result only in terms of the previous state vector elements $s_{k-1}, v_{k-1}, a_{k-1}$ and the constant dt. The following holds for our next time step acceleration:

$$a_k = a_{k-1}$$

and will be denoted simply by a since acceleration is constant. We can then write for the velocity

$$\begin{aligned} v_k &= \underbrace{v_0 + a \cdot t_k}_{v_k} + a \cdot dt \\ &= v_{k-1} + a \cdot dt \end{aligned}$$

where v_0 is the starting velocity and t_k is the elapsed time since departure. Finally, the equation governing the evolution of position can be written as follows:

$$\begin{aligned} s_k &= s_{k-1} + (\underbrace{v_0 + a \cdot t_{k-1}}_{v_{k-1}})dt + \frac{a \cdot dt^2}{2} \\ &= s_{k-1} + v_{k-1} \cdot dt + \frac{a \cdot dt^2}{2} \end{aligned}$$

with which equation 4.5 has been proven.

Appendix B
Kalman Filter Scripts

B.1 Linear Kalman Filter applied to one or two carts

```
#linear kalman filter for one or two carts
# \ means line is broken and continued on the line below
import numpy as np
import matplotlib.pylab as plt

#duration of cart acceleration
duration=20
#update time
dt=0.1
#number of carts
ncarts=2

#action matrices based on physics; x=(position;velocity;acceleration)
#evolution of state vector based on physical model
A=[[1, dt, dt**2/2],[0, 1, dt],[0, 0, 1,]]
C=[[1, 0, 0]]#mapping state vector to measurement

u=np.zeros(ncarts)#carts accelerations vector
q=np.zeros(ncarts)#carts starting positions vector
Q=np.zeros((3,ncarts))#carts state vectors
#main variables
q[0]=0 #cart1 starting point
q[1]=5 #cart2 starting point
u[0]=1.5 #cart1 acc
u[1]=0.5 #cart2 acc

#Generate initial state vector for all carts
for gg in range(ncarts):
    Q[0,gg]=q[gg]
    Q[2,gg]=u[gg]
Q=np.mat(Q)#initial cart state vector
delta_x=40#stdv measurement

#actual locations of the carts
Q_loc=np.zeros((ncarts,int(duration/dt+1)))
#measured locations
Q_meas=np.zeros((ncarts,int(duration/dt+1)))
#measurement noise at each step (independent for all carts)
measnoise=np.zeros(ncarts)
measment=np.zeros(ncarts)
```

```
#generate cart path and what is measured
for ii in range(0,int(duration/dt+1)):
    for kk in range(ncarts):
        #cart paths
        Q[:,kk]=np.mat(A)*np.mat(Q[:,kk])
        #measurements
        measnoise[kk]=delta_x*np.random.randn()
        measment[kk]=np.mat(C)*np.mat(Q[:,kk])+measnoise[kk]
    #make plot vectors
    Q_loc[:,ii]=Q[0,:]#actual cart paths
    Q_meas[:,ii]=measment#actual measured paths

#KALMAN FILTER
#Re-initialize positions etc
#initial state pred covariance
#(starting position known, uncertain acceleration)
P=[[0, 0, 0],[0, 0, 0],[0, 0, 10**10]]
P=np.mat(P)
R=delta_x**2#measurement variance
#array for kalman estimated positions
Q_Kal=np.zeros((ncarts,int(duration/dt+1)))
Q_est=np.zeros((3,ncarts))#array for state vectors estimate
#Initial state
for yy in range(ncarts):
    Q_est[0,yy]=q[yy]
Q_est=np.mat(Q_est)
mu=np.zeros(ncarts)
sigma=np.zeros(ncarts)
Integr=np.zeros((ncarts,2))
Probs=np.zeros(ncarts)

from scipy.integrate import quad
def gaussian(x,mu,sigma):
    return (1/(np.sqrt(2*np.pi)*sigma))\
            *np.exp(-(x-mu)**2/(2*sigma**2))
#Do the Kalman filter with the weights
for tt in range(0,int(duration/dt+1)):
    #state prediction
    for jj in range(ncarts):
        Q_est[:,jj]=np.mat(A)*np.mat(Q_est[:,jj])
    P=np.mat(A)*P*np.mat(A).T#prediction covariance
    #Kalman gain
    K=np.mat(P)*np.mat(C).T/(np.mat(C)*np.mat(P)*np.mat(C).T+R)
    ##Assign state estimates weights
    if ncarts==2:
        for bb in range(ncarts):
            for qq in range(ncarts):
                mu[qq]=Q_est[0,qq]
                sigma[qq]=delta_x
                I=quad(gaussian,mu[qq],Q_meas[bb,tt],args\
```

```
                    =(mu[qq],sigma[qq]))
                Integr[qq,:]=np.asarray(I)
            Area=np.absolute(Integr)
            Probs[bb]=(0.5-Area[0,0])/((0.5-Area[0,0])+\
                        (0.5-Area[1,0]))
        #weight that measurement 1 belongs to cart 1
        W=(Probs[0]+1-Probs[1])/2
        #Kalman update
        Q_est[:,0]=Q_est[:,0]+np.mat(K)*(W*Q_meas[0,tt]+\
                    (1-W)*Q_meas[1,tt]-np.mat(C)*np.mat(Q_est[:,0]))
        Q_est[:,1]=Q_est[:,1]+np.mat(K)*((1-W)*Q_meas[0,tt]+\
                    W*Q_meas[1,tt]-np.mat(C)*np.mat(Q_est[:,1]))
    else:
        Q_est[:,0]=Q_est[:,0]+np.mat(K)*(Q_meas[0,tt]-np.mat(C)\
                    *np.mat(Q_est[:,0]))
    #update covariance estimate
    P=(np.identity(3)-np.mat(K)*np.mat(C))*np.mat(P)
    #store for plot
    Q_Kal[:,tt]=Q_est[0,:]

#plot all data
x=np.arange(0,duration+dt,dt)
plt.plot(x,Q_loc[0,:].T,'-k',label='Cart paths')
plt.scatter(x,Q_meas[0,:].T,c='r',label='Measured positions')
plt.plot(x,Q_Kal[0,:].T,'-g',label='Kalman filter')
for uu in range(1,ncarts):
    plt.plot(x,Q_loc[uu,:].T,'-k')
    plt.scatter(x,Q_meas[uu,:].T,c='r')
    plt.plot(x,Q_Kal[uu,:].T,'-g')
plt.legend(loc='upper left',fontsize=14)
plt.xlabel('t',fontsize=18)
plt.ylabel('x',fontsize=16)
plt.axis([0,duration,-50,400])
plt.show()
```

B.2 EKF simulation: spherical wave emitted from point source

```
#nonlinear kalman filter  for multiple radio sources
import math as m
import numpy as np
import matplotlib.pylab as plt

#Define simulation values
nantennas=80#number of antennas
nsources=1#number of sources
x1=50 #position first antenna (define coordinate system's origin)
delta=1#space between antennas
delta_time_antenna=2 #stdv of measured time by antennas
#Sources location and emission time
```

```
t0=np.zeros(nsources)
x0=np.zeros(nsources)
y0=np.zeros(nsources)
z0=np.zeros(nsources)
t0[0]=0
#t0[1]=0

x0[0]=100
#x0[1]=250

y0[0]=50
#y0[1]=50

z0[0]=0

##main variables##
#initial source state vector
C=[[1, 0, 0, 0]]#Measure vector; S=(time;x;y;z)
S=np.zeros((4,nsources),dtype=np.double)
for mm in range(nsources):
    S[0,mm]=t0[mm]
    S[1,mm]=x0[mm]
    S[2,mm]=y0[mm]
    S[3,mm]=z0[mm]

#Initialize variables
    #matrix containing antenna positions
x=np.arange(x1,x1+nantennas*delta,delta)
y=np.zeros(nantennas)
z=np.zeros(nantennas)
#in case some antennas are not at y=0
#for jj in range(3):
#    y[jj]=jj+1

#generate real path and antenna observations
time_actual=np.zeros((nsources,nantennas))
time_observed=np.zeros((nsources,nantennas))
measnoise=np.zeros(nsources)
measmnt=np.zeros(nsources)
for ii in range(0,nantennas):
    #Wave path
    for nn in range(nsources):
        S[0,nn]=t0[nn]+m.sqrt((x[ii]-x0[nn])**2+\
                (y[ii]-y0[nn])**2+(z[ii]-z0[nn])**2)
        #What antennas measure
        measnoise[nn]=delta_time_antenna*np.random.randn()
        measmnt[nn]=np.mat(C)*np.mat(S[:,nn]).T+measnoise[nn]
    #make plot vectors
    time_actual[:,ii]=S[0,:]#Actual arrival time in antennas
    time_observed[:,ii]=measmnt#what antennas measure
```

```
    for kk in range(nsources):
        plt.scatter(time_actual[kk,ii],x[ii], c='k')
        plt.scatter(time_observed[kk,ii],x[ii], c='r')

#INitials for filter
#first guess
S_est=np.zeros((4,nsources),dtype=np.double)#Inital estimate
for pp in range(nsources):
    S_est[0,pp]=time_observed[pp,3]
    S_est[1,pp]=x0[pp]-40
    S_est[2,pp]=y0[pp]+50
    S_est[3,pp]=0#
S_est=np.mat(S_est)
#Kalman filter
#Re-initialize positions, covariance etc
V=np.array(([[500, 0, 0, 0],[0, 500, 0, 0],\
           [0, 0, 500, 0],[0, 0, 0, 500]]))
P=np.zeros((4*nsources,4*nsources))
for qq in range(nsources):
#initial state pred covariance
    P[4*qq:4*qq+4,4*qq:4*qq+4]=delta_time_antenna**2*np.mat(V)
R=delta_time_antenna**2#measurement variance
S_Kal=np.zeros((nsources,nantennas))
A=np.zeros((nsources,nantennas))
B=np.zeros((nsources,nantennas))
D=np.zeros((nsources,nantennas))
G=np.zeros((4*nsources,4*nsources))
K=np.zeros(4*nsources)
#weights init
mu=np.zeros(nsources)
sigma=np.zeros(nsources)
Integr=np.zeros((nsources,2))
Probs=np.zeros(nsources)
from scipy.integrate import quad
def gaussian(x,mu,sigma):
    return (1/(np.sqrt(2*np.pi)*sigma))\
           *np.exp(-(x-mu)**2/(2*sigma**2))
for tt in range(2,nantennas):
    for ll in range(nsources):
        #State prediction
        S_est[0,ll]=S_est[0,ll]+m.sqrt((x[tt]-S_est[1,ll])**2+\
        (y[tt]-S_est[2,ll])**2+(z[tt]-S_est[3,ll])**2)-\
        m.sqrt((x[tt-1]-S_est[1,ll])**2+\
        (y[tt-1]-S_est[2,ll])**2+(z[tt-1]-S_est[3,ll])**2)
    if nsources!=1:
        for bb in range(nsources):
            for zz in range(nsources):
                mu[zz]=S_est[0,zz]
                sigma[zz]=delta_time_antenna#
```

```
                I=quad(gaussian,mu[zz],time_observed[bb,tt],args=\
                    (mu[zz],sigma[zz]))
                Integr[zz,:]=np.asarray(I)
                Area=np.absolute(Integr)
                Probs[bb]=(0.5-Area[0,0])/((0.5-Area[0,0])+\
                            (0.5-Area[1,0]))
        #weight that measurement 1 belongs to quail 1
        W=Probs[0]/(Probs[0]+Probs[1])
    else:
        W=1
    for hh in range(nsources):
        #Construct error propagation matrix for prediction of
        #error covariance (using partial derivatives)
        #df/dx_i:
        A[hh,tt]=-(S_est[1,hh]-x[tt-1])/m.sqrt((x[tt-1]-S_est[1,hh])**2+\
        (y[tt-1]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)+\
        (S_est[1,hh]-x[tt])/m.sqrt((x[tt]-S_est[1,hh])**2+\
        (y[tt]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)
        B[hh,tt]=-(S_est[2,hh]-y[tt-1])/m.sqrt((x[tt-1]-S_est[1,hh])**2+\
        (y[tt-1]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)+\
        (S_est[2,hh]-y[tt])/m.sqrt((x[tt]-S_est[1,hh])**2+\
        (y[tt]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)
        D[hh,tt]=-(S_est[3,hh]-z[tt-1])/m.sqrt((x[tt-1]-S_est[1,hh])**2+\
        (y[tt-1]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)+\
        (S_est[3,hh]-z[tt])/m.sqrt((x[tt]-S_est[1,hh])**2+\
        (y[tt]-S_est[2,hh])**2+(z[tt-1]-S_est[3,hh])**2)
        #Jacobian:
        G[4*hh:4*hh+4,4*hh:4*hh+4]=np.array([[1, A[hh,tt], B[hh,tt], \
        D[hh,tt]],[0, 1, 0, 0],[0, 0, 1, 0],[0, 0, 0, 1]])
        #Predicted error covariance
        P[4*hh:4*hh+4,3*hh:4*hh+4]=np.mat(G[4*hh:4*hh+4,4*hh:4*hh+4])*\
        np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])*\
        np.mat(G[4*hh:4*hh+4,4*hh:4*hh+4]).T
        #Kalman gain
        K[4*hh:4*hh+4]=(np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])*np.mat(C).T/(\
        np.mat(C)*np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])*np.mat(C).T+R)).T
        #Update state estimate and error covariance!
        P[4*hh:4*hh+4,4*hh:4*hh+4]=(np.identity(4)-np.mat(K[4*hh:4*hh+4]).T*\
        np.mat(C))*np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])
    #weights
    if W!=1:
        S_est[:,0]=S_est[:,0]+np.mat(K[3*0:3*0+3]).T*\
        (W*time_observed[0,tt]+(1-W)*time_observed[1,tt]-np.mat(C)*\
        np.mat(S_est[:,0]))
        S_est[:,1]=S_est[:,1]+np.mat(K[3*1:3*1+3]).T*\
        ((1-W)*time_observed[0,tt]+W*time_observed[1,tt]-np.mat(C)*\
        np.mat(S_est[:,1]))
    else:
        S_est[:,0]=S_est[:,0]+np.mat(K[4*0:4*0+4]).T*\
        (time_observed[0,tt]-np.mat(C)*np.mat(S_est[:,0]))
```

```
    #Store for plot
    S_Kal[:,tt]=S_est[0,:]
    for ww in range(nsources):
        plt.scatter(S_Kal[ww,tt],x[tt], c='g')#plot points


#plot data
plt.xlabel('t', fontsize=18)
plt.ylabel('x', fontsize=16)
plt.show()
print("S_est=", S_est)
print("x0,y0,z0=", x0, y0, z0)
```

B.3 EKF to LOFAR data

```
#!/usr/bin/env python3

#### just plots a block of data on all antennas
#### usefull for health checks
import pdb
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal

from LoLIM.utilities import processed_data_dir, v_air
from LoLIM.IO.raw_tbb_IO import MultiFile_Dal1, filePaths_by_stationName
from LoLIM.signal_processing import remove_saturation, num_double_zeros,\
    half_hann_window, upsample_and_correlate
from LoLIM.findRFI import window_and_filter
from LoLIM.read_pulse_data import  read_station_delays, \
    read_antenna_pol_flips, read_bad_antennas, read_antenna_delays


## these lines are anachronistic and should be fixed at some point
from LoLIM import utilities
utilities.default_raw_data_loc = "/exp_app2/appexp1/public/raw_data"
utilities.default_processed_data_loc = "/home/student5/APel/output"


if __name__ == "__main__":
    timeID = "D20180813T153001.413Z"
    point = int( 3900*(2**16) )
    block_size = 2**16

    X_source=0#initial estimate for filter
    Y_source=-52000
    Z_source=5000

    X_cc_est=7338#Estimate after running filter (used for RMS,est dev)
```

```
Y_cc_est=-45071.2
Z_cc_est=6426.1

polarization_flips = "polarization_flips.txt"
bad_antennas = "bad_antennas.txt"
additional_antenna_delays = "ant_delays.txt"
station_delays="station_delays.txt"

positive_saturation = 2046
negative_saturation = -2047
saturation_post_removal_length = 50
saturation_half_hann_length = 50

processed_data_folder = processed_data_dir(timeID)
polarization_flips = read_antenna_pol_flips( processed_data_folder +\
                    '/' + polarization_flips )
bad_antennas = read_bad_antennas( processed_data_folder +\
                '/' + bad_antennas )
additional_antenna_delays=read_antenna_delays(processed_data_folder+\
                         '/' + additional_antenna_delays )
station_delays=read_station_delays( processed_data_folder + '/' +\
                station_delays)
raw_fpaths = filePaths_by_stationName(timeID)

nsources=1#number of sources
delta_time_antenna=4.0E-9 #stdv of measured time by antennas
P0=np.array(([[(2.0E-9)**2, 0, 0, 0],[0, 10000**2/(v_air**2), 0, 0],\
[0, 0, 10000**2/(v_air**2), 0],[0, 0, 0, 10000**2/(v_air**2)]]))
P=np.zeros((4*nsources,4*nsources))
for qq in range(nsources):
    P[4*qq:4*qq+4,4*qq:4*qq+4]=np.mat(P0)#initial state pred covariance
R=delta_time_antenna**2#measurement variance
G=np.zeros((4*nsources,4*nsources))
K=np.zeros(4*nsources)
C=[[1, 0, 0, 0]]#Measure vector; S=(time;x;y;z)
S_est=np.zeros((4,nsources),dtype=np.double)

H = 0
resamp_rate=8
first=100
ant_numb=0
time_peaks_cc=np.zeros(150,dtype=np.double)
location_peaks_cc=np.zeros((3,118),dtype=np.double)
dev_sq=np.zeros(150)
first_ant_spect=0

window_datapoints=15
windowed_odd_HE_pair=np.zeros(2*window_datapoints)
windowed_odd_HE_first=np.zeros(2*window_datapoints)
for statname in station_delays.keys():
```

```
station = statname
TBB_data = MultiFile_Dal1( raw_fpaths[station],\
force_metadata_ant_pos=True, polarization_flips=polarization_flips,\
bad_antennas=bad_antennas, \
additional_ant_delays=additional_antenna_delays )
RFI_filter = window_and_filter(timeID=timeID, sname=station)

data = np.empty(block_size, dtype=np.double)

ant_pos=TBB_data.get_LOFAR_centered_positions()
ant_names = TBB_data.get_antenna_names()
num_antenna_pairs = int( len( ant_names )/2 )
ant_del=TBB_data.get_timing_callibration_delays()
station_del=station_delays[station]
nom_sample_numb=TBB_data.get_nominal_sample_number()
print(statname,H)

peak_indices_odd=np.zeros((2,num_antenna_pairs))
peak_values_odd=np.zeros((2,num_antenna_pairs))
for pair in range(num_antenna_pairs):
    if first==100:
        flash_travel_delay=np.sqrt((ant_pos[pair*2,0]-X_source)**2+\
        (ant_pos[pair*2,1]-Y_source)**2+(ant_pos[pair*2,2]-\
        Z_source)**2)/v_air
        TD_even=(station_del+ant_del[pair*2]+flash_travel_delay)\
        /5.0E-9-nom_sample_numb
        Res_even=(TD_even-int(TD_even))*5.0E-9
        TD_odd=(station_del+ant_del[pair*2+1]+flash_travel_delay)\
        /5.0E-9-nom_sample_numb
        cal_delay_odd=station_del+ant_del[pair*2+1]-\
        nom_sample_numb*5.0E-9
        Res_odd=(TD_odd-int(TD_odd))*5.0E-9

        data[:] = TBB_data.get_data(int(point+TD_even),\
        block_size, antenna_index=pair*2)
        numb_double_zeros_even=num_double_zeros(data)
        if numb_double_zeros_even<100:
            remove_saturation(data, positive_saturation, \
            negative_saturation, saturation_post_removal_length,\
            saturation_half_hann_length)
            filtered_data = RFI_filter.filter( data )
            even_HE = np.abs(filtered_data)
            even_HE_re=scipy.signal.resample(even_HE,\
            resamp_rate*len(even_HE))
        else:
            even_HE_re=0

        data[:] = TBB_data.get_data(int(point+TD_odd), \
        block_size, antenna_index=pair*2+1)
```

```
numb_double_zeros_odd=num_double_zeros(data)

if numb_double_zeros_odd<100:
    remove_saturation(data, positive_saturation, \
    negative_saturation, saturation_post_removal_length, \
    saturation_half_hann_length)
    filtered_data = RFI_filter.filter( data )
    odd_HE = filtered_data
    odd_HE_abs = np.abs(filtered_data)
    odd_HE_re=scipy.signal.resample(odd_HE_abs,\
    resamp_rate*len(odd_HE_abs))

    Delta=odd_HE_re[1:]-odd_HE_re[:-1]
    ispeak=np.logical_and(Delta[:-1]>0,Delta[1:]<0)
    peak_index=np.where(ispeak)[0]+1
    peak_amplitude=odd_HE_re[peak_index]
    sorter=np.argsort(-peak_amplitude)
    peak_values_odd[:,pair]=peak_amplitude[sorter[:2]]
    peak_indices_odd[:,pair]=peak_index[sorter[:2]]
    first_ant_pos=ant_pos[pair*2,:]
    first_ant_peak=peak_indices_odd[0,pair]
    res_first_peak=first_ant_peak/resamp_rate*5E-9-\
    int(first_ant_peak/resamp_rate)*5E-9
    time_peaks_cc[ant_numb]=(first_ant_peak/resamp_rate+\
    int(point+TD_odd))*5.0E-9-cal_delay_odd

    for iii in range(3):
        location_peaks_cc[iii,ant_numb]=\
        ant_pos[pair*2,iii]/v_air
    cal_del_first_odd=cal_delay_odd
    TD_first_odd=TD_odd

    peak = np.max(odd_HE)
    odd_HE/=peak

    window_datapoints_first=15
    windowed_odd_HE_first=\
    odd_HE[int(first_ant_peak/resamp_rate)-\
    window_datapoints_first:\
    int(first_ant_peak/resamp_rate)+window_datapoints_first]
    odd_hann_window_first=half_hann_window(len(\
    windowed_odd_HE_first),0.1)
    windowed_odd_HE_first=windowed_odd_HE_first*\
    odd_hann_window_first

    #Initialize filter
    #first guess
    for pp in range(nsources):
        S_est[0,pp]=time_peaks_cc[0]
        S_est[1,pp]=X_source/v_air
```

```
                S_est[2,pp]=Y_source/v_air
                S_est[3,pp]=Z_source/v_air
            S_est=np.mat(S_est)
            ant_numb=ant_numb+1
        else:
            odd_HE=0


    if first!=100:
        #skip antennas/stations which give problems
        if station=="RS503" and pair==0:
            continue
        #if station=="RS406" and pair==0:
        #    continue
        #if station=="RS208" and pair==2:
        #    continue
        if station=="RS508":
            continue
        if station=="RS409":
            continue
        #if station=="RS306":
        #    continue

        #check if numb_double_zeros>100 before\
        #updating state vector (not wanted if antenna is not active)
        cal_delay_odd=station_del+ant_del[pair*2+1]-\
        nom_sample_numb*5.0E-9
        est_bl_op=((S_est[0,0]+(np.sqrt((v_air*\
        (location_peaks_cc[0,ant_numb]-S_est[1,0]))**2+\
        (v_air*(location_peaks_cc[1,ant_numb]-S_est[2,0]))**2+\
        (v_air*(location_peaks_cc[2,ant_numb]-S_est[3,0]))**2)\
        -np.sqrt(\
        (v_air*(location_peaks_cc[0,ant_numb-1]-S_est[1,0]))**2+\
        (v_air*(location_peaks_cc[1,ant_numb-1]-S_est[2,0]))**2+\
        (v_air*(location_peaks_cc[2,ant_numb-1]-S_est[3,0]))**2))\
        /v_air)+cal_delay_odd)/5E-9-block_size/2
        data[:] = TBB_data.get_data(int(est_bl_op),\
        block_size, antenna_index=pair*2+1)
        numb_double_zeros_odd=num_double_zeros(data)
        if numb_double_zeros_odd>100:
            continue
        #store antenna locations
        for jjj in range(3):
                location_peaks_cc[jjj,ant_numb]=ant_pos[pair*2,jjj]\
                /v_air

        for ll in range(nsources):
            #State prediction
            S_est[0,ll]=S_est[0,ll]+(np.sqrt((v_air*\
            (location_peaks_cc[0,ant_numb]-S_est[1,ll]))**2+\
```

```
        (v_air*(location_peaks_cc[1,ant_numb]-S_est[2,ll]))**2+\
        (v_air*(location_peaks_cc[2,ant_numb]-S_est[3,ll]))**2)\
        -np.sqrt((v_air*\
        (location_peaks_cc[0,ant_numb-1]-S_est[1,ll]))**2+\
        (v_air*(location_peaks_cc[1,ant_numb-1]-S_est[2,ll]))**2+\
        (v_air*(location_peaks_cc[2,ant_numb-1]-S_est[3,ll]))**2))\
        /v_air
    for hh in range(nsources):
        #Error propagation
        #df/dx_i:
        A=-(S_est[1,hh]-location_peaks_cc[0,ant_numb-1])/np.sqrt(\
        (location_peaks_cc[0,ant_numb-1]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb-1]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb-1]-S_est[3,hh])**2)\
        +(S_est[1,hh]-location_peaks_cc[0,ant_numb])\
        /np.sqrt((location_peaks_cc[0,ant_numb]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb]-S_est[3,hh])**2)
        B=-(S_est[2,hh]-location_peaks_cc[1,ant_numb-1])/np.sqrt(\
        (location_peaks_cc[0,ant_numb-1]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb-1]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb-1]-S_est[3,hh])**2)\
        +(S_est[2,hh]-location_peaks_cc[1,ant_numb])\
        /np.sqrt((location_peaks_cc[0,ant_numb]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb]-S_est[3,hh])**2)
        D=-(S_est[3,hh]-location_peaks_cc[2,ant_numb-1])/np.sqrt(\
        (location_peaks_cc[0,ant_numb-1]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb-1]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb-1]-S_est[3,hh])**2)\
        +(S_est[3,hh]-location_peaks_cc[2,ant_numb])\
        /np.sqrt((location_peaks_cc[0,ant_numb]-S_est[1,hh])**2+\
        (location_peaks_cc[1,ant_numb]-S_est[2,hh])**2+\
        (location_peaks_cc[2,ant_numb]-S_est[3,hh])**2)
        #Jacobian
        G[4*hh:4*hh+4,4*hh:4*hh+4]=np.array([[1, A, B, D],\
        [0, 1, 0, 0],[0, 0, 1, 0],[0, 0, 0, 1]])
        #Predicted error covariance
        P[4*hh:4*hh+4,3*hh:4*hh+4]=np.mat(\
        G[4*hh:4*hh+4,4*hh:4*hh+4])*np.mat(\
        P[4*hh:4*hh+4,4*hh:4*hh+4])*\
        np.mat(G[4*hh:4*hh+4,4*hh:4*hh+4]).T

    cal_delay_odd=station_del+ant_del[pair*2+1]-\
    nom_sample_numb*5.0E-9
    block_open_est=(S_est[0,0]+cal_delay_odd)/5E-9-block_size/2
    res_est_block_open=block_open_est*5E-9-\
    int(block_open_est)*5E-9
    data[:] = TBB_data.get_data(int(block_open_est),\
    block_size, antenna_index=pair*2+1)
```

```
numb_double_zeros_odd=num_double_zeros(data)
#arrival time based on final location found in previous run
model_arr_time=time_peaks_cc[0]+np.sqrt(\
(ant_pos[pair*2,0]-X_cc_est)**2+\
(ant_pos[pair*2,1]-Y_cc_est)**2+\
(ant_pos[pair*2,2]-Z_cc_est)**2)/v_air-\
np.sqrt((v_air*location_peaks_cc[0,0]-X_cc_est)**2+\
(v_air*location_peaks_cc[1,0]-Y_cc_est)**2+\
(v_air*location_peaks_cc[2,0]-Z_cc_est)**2)/v_air

if numb_double_zeros_odd<100:
    remove_saturation(data, positive_saturation,\
    negative_saturation, saturation_post_removal_length,\
    saturation_half_hann_length)
    filtered_data = RFI_filter.filter( data )
    odd_HE = filtered_data
    #hanwindows
    windowed_odd_HE_pair=odd_HE[int(block_size/2)-\
    window_datapoints:int(block_size/2)+window_datapoints]
    odd_hann_window=half_hann_window(\
    len(windowed_odd_HE_pair),0.1)
    windowed_odd_HE_pair=windowed_odd_HE_pair*\
    odd_hann_window

    peak = np.max(windowed_odd_HE_pair)
    windowed_odd_HE_pair/=peak

    #cross-correlations
    x_cc=np.arange(0,2*len(windowed_odd_HE_pair)*\
    resamp_rate)/resamp_rate
    x_cc*=5.0E-9
    correlator=upsample_and_correlate(len(\
    windowed_odd_HE_pair),resamp_rate)
    cc=np.abs(correlator.run(windowed_odd_HE_pair,\
    windowed_odd_HE_first))
    cc_comp=correlator.run(windowed_odd_HE_pair,\
    windowed_odd_HE_first)
                ##find peak of cc
    Delta=cc[1:]-cc[:-1]
    ispeak=np.logical_and(Delta[:-1]>0,Delta[1:]<0)
    peak_index=np.where(ispeak)[0]+1
    peak_amplitude=cc[peak_index]
    sorter=np.argsort(-peak_amplitude)
    peak_values_odd[:,pair]=peak_amplitude[sorter[:2]]
    peak_indices_odd[:,pair]=peak_index[sorter[:2]]

    if peak_indices_odd[0,pair]<int(len(x_cc)/2):
        delta_time_cc=x_cc[int(peak_indices_odd[0,pair])]
    else:
        delta_time_cc=-(x_cc[len(x_cc)-1]-\
```

```
    x_cc[int(peak_indices_odd[0,pair])])
if cc[1]-cc[0]<0 and (cc[len(cc)-1]-cc[len(cc)-2])>0:
    if cc[0]>cc[int(peak_indices_odd[0,pair])]:
        delta_time_cc=0
if delta_time_cc>1E-8:
    print(delta_time_cc)

time_peaks_cc[ant_numb]=S_est[0,0]+delta_time_cc+\
res_first_peak-res_est_block_open
dev_sq[ant_numb]=(model_arr_time-\
time_peaks_cc[ant_numb])**2

                    #rotations for plotting cc
windowed_odd_re_first=scipy.signal.resample(\
windowed_odd_HE_first,resamp_rate*\
len(windowed_odd_HE_first))
windowed_odd_re_pair=scipy.signal.resample(\
windowed_odd_HE_pair,resamp_rate*\
len(windowed_odd_HE_pair))

x_cc_rot=np.arange(len(cc))/resamp_rate
x_cc_rot-=(len(x_cc)/(2*resamp_rate)-1/(2*resamp_rate))
cc_comp_rot=np.zeros(len(cc))
cc_comp_rot[0:int(len(cc_comp)/2)]=cc_comp[\
int(len(cc_comp)/2):len(cc_comp)]
cc_comp_rot[int(len(cc_comp)/2):len(cc_comp)]=\
cc_comp[0:int(len(cc_comp)/2)]
cc_rot=np.zeros(len(cc))
cc_rot[0:int(len(cc)/2)]=cc[int(len(cc)/2):len(cc)]
cc_rot[int(len(cc)/2):len(cc)]=cc[0:int(len(cc)/2)]

x_window_first=np.arange(len(windowed_odd_HE_first))*1.0
x_window_first_re=np.arange(resamp_rate*len(\
windowed_odd_HE_first))/(1.0*resamp_rate)
x_window_first-=(len(x_window_first)/2)
x_window_first_re-=(len(x_window_first)/2)

if ant_numb>130:
    #in case one wants to know\
    #what the windowed pulses look like
    print(time_peaks_cc[ant_numb], pair)
    plt.plot(x_window_first,windowed_odd_HE_first)
    plt.plot(x_window_first_re,\
    np.abs(windowed_odd_re_first),"g")
    plt.plot(x_window_first,windowed_odd_HE_pair+2)
    plt.plot(x_window_first_re,\
    np.abs(windowed_odd_re_pair)+2,"r")

    plt.plot(x_cc_rot,cc_comp_rot/np.max(cc_comp_rot)+4)
    plt.plot(x_cc_rot,cc_rot/np.max(cc_rot)+4)
```

```
                    #plt.axvline(x=0,color="r")
                    #plt.axvline(x=delta_time_cc/5E-9)
                    plt.xticks(fontsize=12)
                    plt.yticks(fontsize=12)
                    plt.xlabel("Datapoints",fontsize=14)
                    plt.show()
                #Kalman gain and combination procedure
                    for hh in range(nsources):
                        #Kalman gain
                        K[4*hh:4*hh+4]=(np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])*\
                        np.mat(C).T/(np.mat(C)*\
                        np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])*np.mat(C).T+R)).T
                        #Update state estimate and error covariance!
                        P[4*hh:4*hh+4,4*hh:4*hh+4]=(np.identity(4)-\
                        np.mat(K[4*hh:4*hh+4]).T*np.mat(C))*\
                        np.mat(P[4*hh:4*hh+4,4*hh:4*hh+4])
                        S_est[:,0]=S_est[:,0]+np.mat(K[4*0:4*0+4]).T*\
                        (time_peaks_cc[ant_numb]-np.mat(C)*\
                        np.mat(S_est[:,0]))[0,0]
                    ant_numb=ant_numb+1

            if numb_double_zeros_odd<100 and first==100:
                first=pair
                print(time_peaks_cc[ant_numb-1],ant_numb-1)

        H += 1
        #if station=="CS501":
        #    break
    print("x_estimated=", v_air*S_est[1,0].T,"while initial input=", X_source)
    print("y_estimated=", v_air*S_est[2,0].T,"while initial input=", Y_source)
    print("z_estimated=", v_air*S_est[3,0].T,"while initial input=", Z_source)
    idx = np.where(time_peaks_cc == 0)#remove zero columns
    time_peaks_cc = np.delete(time_peaks_cc, idx, axis=0)#remove zero columns
    RMS_dev=np.sqrt(sum(dev_sq)/(len(time_peaks_cc)-1))
    print("RMS=",RMS_dev)

    idxx = np.argwhere(np.all(location_peaks_cc[..., :] == 0, axis=0))
    #remove zero columns
    location_peaks_cc = np.delete(location_peaks_cc, idxx, axis=1)
    y=location_peaks_cc[1,:], z=location_peaks_cc[2,:])

    #non-linear fitter to data to compare
    time_observed=time_peaks_cc[:]
    xx=v_air*location_peaks_cc[0,:]
    yy=v_air*location_peaks_cc[1,:]
    zz=v_air*location_peaks_cc[2,:]
    t0=time_observed[0]-np.sqrt((xx[0]-X_source)**2+\
    (yy[0]-Y_source)**2+(zz[0]-Z_source)**2)/v_air
    ##lmfit:
    from lmfit import Model
```

```
def time_obs(D, t0,x0,y0,z0):
    return t0+np.sqrt((xx-x0)**2+(yy-y0)**2+(zz-z0)**2)/v_air
gmodel = Model(time_obs)
result = gmodel.fit(time_observed, D=3,  t0=t0,\
x0=X_source, y0=Y_source, z0=Z_source)

print(result.fit_report())
deviation=time_obs(3,result.values['t0'],result.values['x0'],\
result.values['y0'],result.values['z0'])-time_observed
for mm in range(len(deviation)):
    deviation[mm]=deviation[mm]**2
RMS_LM=np.sqrt(np.sum(deviation)/len(deviation))
print("RMS_dev_LMfit =", RMS_LM)
```

Bibliography

[1] John E. Oliver. *Encyclopedia of World Climatology*. Springer, 2005. ISBN 97814020326465.

[2] Wikipedia. Kite experiment, 2019. URL https://en.wikipedia.org/wiki/Kite_experiment#cite_note-12.

[3] Tom Tucker. *Bolt of fate: Benjamin Franklin And His Electric Kite Hoax*. PublicAffairs, 2003. ISBN 97807567933265.

[4] Vladimir A. Rakov and Martin A. Uman. *Lightning: Physics and Effects*. Cambridge University Press, 2006. ISBN 9780521035415.

[5] Martin A. Uman. Everything you always wanted to know about lightning but were afraid to ask. *The Saturday Review*, 32:36–41, 1972.

[6] C.V. Boys. Progressive lightning. *Nature*, 118:749–750, 1926.

[7] B. F. J. Schonland. The Lightning Discharge. *Handbuch der Physik*, 22:576–628, 1956.

[8] D. Ward University of Arizona. Lecture notes on atmospheric sciences, 2018. URL http://www.atmo.arizona.edu/students/courselinks/spring08/atmo336s1/courses/spring18/atmo170a1s2/lecture_notes/apr20.html.

[9] B.M. Hare O. Scholten A. Bonardi S. Buitink A. Corstanje et al. Lofar lightning imaging: Mapping lightning with nanosecond precision. *Journal of geophysical research:Atmospheres*, 123:2861–2876, 2018.

[10] M.A. Uman and J.R. Dwyer. The physics of lightning. *Physics reports*, 534: 147–241, 2014.

[11] Encyclopaedia Britannica. Thunderstorm electrification, 2019. URL https://www.britannica.com/science/thunderstorm/Thunderstorm-electrification.

[12] R. Solomon V. Schroeder and M.B. Baker. Lightning initiation-conventional and runaway-breakdown hypotheses. *Q. J. R. Meteorol. Soc.*, 127:2683–2704, 2000.

[13] A.V. Gurevich G.M. Milikh and R. Russel-Dupre. Runaway electron mechanism of air breakdown and preconditioning during a thunderstorm. *Physics Letters A*, 165:436–438, 1992.

[14] J.R. Dwyer. The initiation of lightning by runaway air breakdown. *Geophysical Research Letters*, 32:L208008, 2005.

[15] B.F.J. Schonland. The lightning discharge. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 143 (850), 1934.

[16] J. Kasparian R. Ackermann Y.B. Andre G. Mechain G. Mejean et al. Progress towards lightning control using lasers. *Journal of the European Optical Society*, 3(08035), 2008.

[17] T.N.G. Trinh O. Scholten. A. Bonardi S. Buitink A. Corstanje et al. Thunderstorm electric fields probed by extensive air showers through their polarized radio emission. *Physical review D*, 95(8), 2017.

[18] B.M. Hare O. Scholten J. Dwyer T.N.G. Trinh S. Buitink et al. Needle-like structures discovered on positively charged lightning branches. *Nature*, 568: 360–363, 2019.

[19] Astron. Lofar, 2019. URL http://www.lofar.org/.

[20] P. Schellart A. Nelles S. Buitink A. Corstanje J. E. Enriquez et al. Detecting cosmic rays with the lofar radio telescope. *AA*, 560(A98), 2013.

[21] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[22] Ramsey Faragher. Understanding the basis of the kalman filter via a simple and intuitive derivation. *IEEE Signal Processing Magazine*, 29(5):128–132, 2012.

[23] Gabriel Terejanu. Extended kalman filter tutorial. *University at Buffalo*, 2008.

[24] Wikipedia. Cross-correlation, 2019. URL https://en.wikipedia.org/wiki/Cross-correlation.