Title
An Algorithm for High-Resolution Multipath Mitigation in a Channel with Known Constraints

Author
Jishi, Ali

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

An Algorithm for High-Resolution Multipath Mitigation in a Channel with Known Constraints

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Electrical and Computer Engineering

by

Ali Radi Jishi

2019

ABSTRACT OF THE THESIS

An Algorithm for High-Resolution
Multipath Mitigation in a Channel
with Known Constraints

by

Ali Radi Jishi

Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2019
Professor Yuanxun Wang, Chair

Multipath is the dominant source of error in indoor positioning systems, because it can significantly distort the shape of the correlation function used for time-delay estimation. With a narrowband signal, the range resolution is often insufficient to decompose the overlapped received signals. However, if the signal has constraints, we can theoretically estimate the channel response more accurately. We assume the constraint that there is a known amount of nonzero values in the channel response, each of which represents a delayed version of the training signal in the receiver. We focused our work on the principles of the least mean squares equalizer, which tries to estimate the channel response, because it is fast and can be implemented in parallel and in real-time. Our modified algorithm uses the general principle of the least mean squares equalizer

and sets the constraints. The algorithm is designed to be implemented in parallel and in real-time.

The thesis of Ali Radi Jishi is approved.

Danijela Cabric

Yahya Rahmat-Samii

Yuanxun Wang, Committee Chair

University of California, Los Angeles

2019

Table of Contents

List of Figures

List of Tables

Introduction

Ranging is at the heart of many engineering systems and problems. One example is localization, which is a very ubiquitous problem with a wide range of applications, including the Global Positioning System (GPS). GPS estimates location using trilateration, which requires estimates of the range between a satellite and the user. Trilateration systems estimate the location based on the time-of-arrival or the time-difference-of-arrival of known signals coming from different sources with a known position.

GPS is reliable in many environments, but is unable to provide an accurate or meaningful location when used indoors, especially in large buildings. This is in part because of the large attenuation in large buildings, but also because of multipath propagation [2],[4]. Multipath propagation is the process by which a signal traveling from a transmitter to a receiver reflects off objects and arrives at the receiver in a path different from the direct path and with a delay, interfering with the shortest signal. Multipath is the main source of ranging error in indoor localization systems in general because indoor environments are characterized by closely spaced walls and ceilings that reflect the signal, leading to attenuation of the signal power if there is a wall between the transmitter and the receiver and resulting in potentially powerful reflections interfering with each other and with the shortest signal [1].

The effects on signal propagation through a multipath channel can be characterized by equation 1 below [1]. In this equation, n is a discrete time variable, s is the signal being sent, and r is the signal that is received. In the first term on the right hand side, we see that there are two coefficients that capture the act of propagation of the direct signal or the reflections: the term a_i represents the attenuation of that particular signal between the transmitter and the receiver, and the term b_i represents the relative delay associated with the propagational element i. The second

term with the exp(j*Φ_i) refers to a reflection that results in a change in the carrier frequency. In a real system, this would most likely result from a Doppler shift due to a moving reflector. In our design, we will assume that none of the reflections result in a frequency shift, thereby eliminating that term from consideration, and with that the coefficients of the amplitude and delay associated with that term.

$$r(n) = \Sigma_i(a_i * s[n - b_i] + c_i * s[n - d_i] * \exp(j * \Phi_i)) \qquad (1)$$

In general, neglecting changes in the speed of the carrier due to a dielectric, the signal with the least delay is the line-of-sight signal (LOS) because it represents the signal that propagated from the transmitter to the receiver without any reflection. The other amplitudes and delays are multipath, or non-line-of-sight (NLOS).

The relevant amplitude and delay information can be combined into one vector, where the indices linearly represent the relative delays and the value of the vector at each index represents the net magnitude of the received signals with that delay. An important advantage of this form is that the received signal can be written as the convolution of the sent signal with this vector. This vector is therefore the channel response. In this form, the first nonzero term is the LOS signal, and the nonzero terms after that are the NLOS signals. Since each LOS and NLOS signal represents a nonzero element in the channel response with an index corresponding to that delay, in our discussion, we will refer to LOS or NLOS signals as pulses or Kronecker deltas in the channel response.

The problem under consideration is that someone is in a large building and people outside want to be able to track his or her location based on a signal he or she emits from a device. The system under consideration involves a set of base-stations that are located outside a building, and

the base-stations can synchronize and communicate. In this context, localization can be achieved by a time-difference-of-arrival approach, simplifying the problem of interest into a ranging problem, where we try to find the delays of the LOS signals between the transmitter and the base-station relative to one-another [2]. Since the source is indoors, the resolution is limited by ranging errors due to multipath.

In addition to the ranging errors caused by multipath, if this is to be designed to be implemented in a real-world system, a major limitation is the bandwidth. For a signal with a 3 dB bandwidth of B and with C as the speed of light, the range-resolution is C/B in distance or 1/B in time. This means that the smaller the band-width, the harder it is to resolve the range. This is a problem in real-world problems because we have to consider FCC regulations as well as the propagational properties of electromagnetic waves. Firstly, when we consider that multipath propagation from the inside of a building leads to a huge decline in total power that arrives at the receiver, we realize that we have to use one of the frequencies allocated for industrial, scientific, and medical (ISM) applications. Secondly, if we expect the LOS signal to propagate through walls and arrive at the receiver with a reasonable power, then the carrier frequency must be relatively low. In table 1 below, we see a list of the international ISM frequencies relevant to the United States, the total band-widths [31], and the range resolution.

Center Frequency	Bandwidth	Minimum Range for Resolution (m)
6.78 MHz	30 KHz	10000
13.56 MHz	14 KHz	21428
27.12 MHz	326 KHz	920.2
40.68 MHz	40 KHz	7500
915 MHz	26 MHz	11.54
2.45 GHz	100 MHz	3
5.8 GHz	150 MHz	2
24.125 GHz	250 MHz	1.2
61.25 GHz	500 MHz	0.6
122.5 GHz	1 GHz	0.3
245 GHz	2 GHz	0.15

Table 1: The United States ISM bands and their range resolutions

Therefore, we surmise that in order to get maximum band-width and propagation, we should use the 2.45 GHz band or the 915 MHz band. With these band-widths, the maximum theoretical resolutions are 3 m and 11.54 m, respectively. When we consider that for the 2.45 GHz band, the LOS power received would be attenuated and when we consider that there would be a lot of interference from other systems, such as WiFi, a realistic localization system requires a much better resolution. Therefore, the problem is further defined as trying to map the channel response between the transmitter and the receiver when the components of the channel response have a separation smaller than the minimum resolution, and to do so in the best way possible.

A signal of interest is the ATSC DTV signal. The ATSC standard uses an 8-VSB signal with a bandwidth of 6 MHz with a carrier in the UHF band in the 600 MHz range. ATSC signals contain training signals used for synchronization and for other purposes. One important training signal is the PN511 code, which consists of 511 symbols that repeat every 24.2 milliseconds.

Although digital television signals cannot be emitted at these frequencies, they prove useful in simulations [13].

Having defined the problem, further steps are taken to allow us to analyze and experiment with different algorithms and techniques. Firstly, the band-limited signals are over-sampled. This can come from physical oversampling of the signal (after extraneous frequencies have been filtered) or by sampling at the Nyquist rate and then performing a Fourier interpolation to a smaller step size. Both of these result in the frequency components of the signal being that of the original signal below their frequency limits and zero above the limit. These processes result in smaller time scales, which are needed to handle a higher resolution. In other words, a higher frequency scale becomes present in the problem, meaning that one can describe sharper/higher frequency pulses with these time scales, which is necessary in describing separations below the minimum resolution [3],[19].

Secondly, in order to efficiently experiment with techniques, we simulate propagation and receiver effects by defining a channel response vector and a training signal, convolving an upsampled training signal with the channel response, and downsampling and then interpolating (to ensure that it follows the same process a real system would, and to ensure that all higher-frequency components are zero). This gives us the received signal. To test a technique, we come up with a set of channel responses and their respective received signals and use whatever technique we are examining to find the channel response it predicts, and then compare that with the known ground truth. In this process, we define the ground truth channel response as a series of Kronecker delta functions, which defines the delay and amplitude of each LOS and NLOS signal arriving at the receiver. Note that this technique implicitly assumes that the delay is precisely as defined in the ground truth channel response vector, so the channel response vector

needs to have a small time scale to be precise and to have enough resolution to model closely spaced multipath environments. Furthermore, it is important to simulate cases in which many of the later delta functions are just as strong as or stronger than the first one if we wish for this to be useful for indoor environments.

Finally, in order to realistically be able to solve the problem at hand, it is vital to assume certain constraints in the channel. The reason is that Nyquist's limit essentially sets the minimum sampling frequency required to capture all of the information in a signal known to be constrained only by frequency, but does not strictly apply to signals known to have further constraints. This idea forms the basis of super-resolution algorithms, such as compressive sensing [7],[8], which will be discussed later on. In our work, we assume that we have knowledge of the training signal and the amount of Kronecker delta functions in the channel response; or in other words, the channel response is constrained to having only the predetermined amount of Kronecker deltas. With these three points in mind, we have the tools required to analyze and tackle this problem.

Review of Literature

Correlation

The correlation is a common method for separating or extracting a component of a signal. Correlation is the process of linearly finding the presence of one signal in another by taking the inner products of the two vectors. Furthermore, we assign a delay variable to one of the vectors and take the inner product at each delay to find the presence of a delayed version of one signal in another as a function of the delay variable. Separating multipath components of a signal works by knowing the training signal and correlating that with the received signal, theoretically locating the presence of delayed versions of that training signal. Some of its key advantages are that it can be done in real-time/online and that it can be implemented with relative simplicity [14].

Ideally, for this to be precise, the correlation pattern would be a series of Kronecker delta functions because it indicates that if there is a delay, it appears in the correlation delay pattern at only an infinitely single time. However, since the inner product used in finding correlation is linear, it results in no change to the bandwidth. In effect, rather than a series of delta functions appearing in the output of the correlation operation, the limited bandwidth results in filtered delta functions. This results in a series of sinc functions. If the delays are more closely spaced than the Fourier resolution, the peaks would be within the sinc full width at half maximum (FWHM), so they interfere and are not resolved. Since the usefulness of this method depends on being able to resolve sinc functions, this method works when the pulses in the channel response are spaced farther than the Nyquist resolution but does not work when they are spaced more closely, making this method inadequate in a problem with an indoor environment.

Figure 1 shows a block diagram of the steps of the correlation method when applied to a real time signal. The delay of a signal component relative to a clock with the same frequency as

the signal repetition rate (this can simply be an ATSC signal, since the PN511 codes repeat at the same frequency) is estimated by the position of the peaks relative to that clock [14].

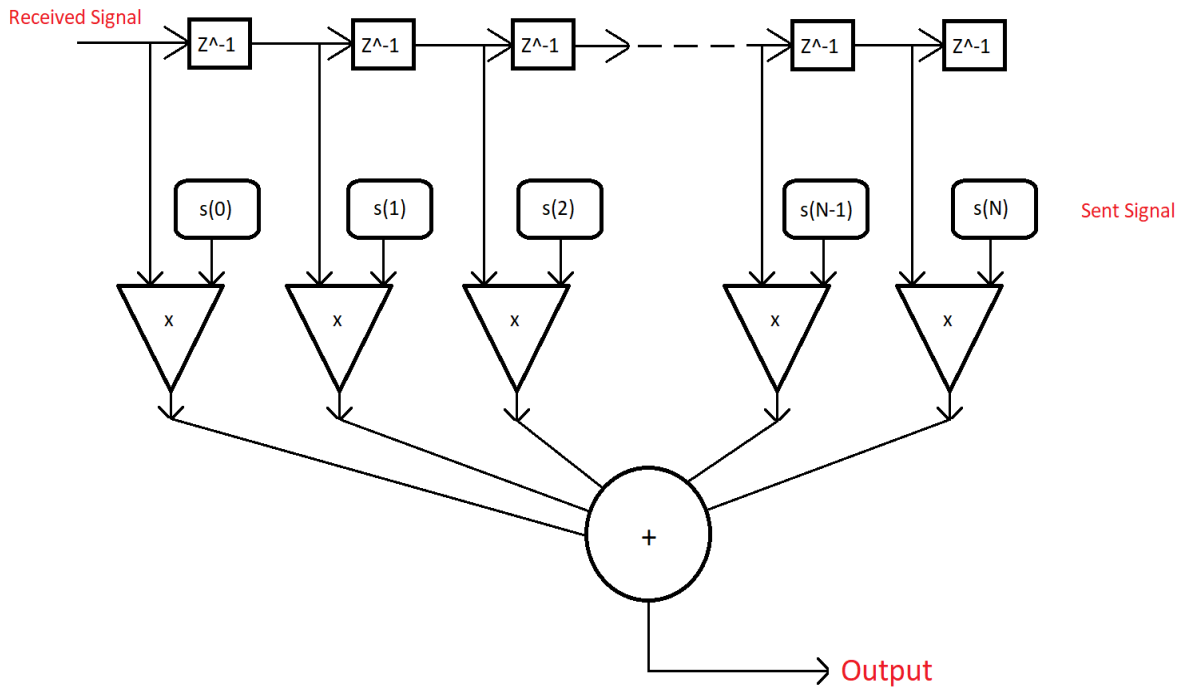

Figure 1: A block diagram of the process used to find the cross-correlation in real-time.

To demonstrate the bandwidth limitation of the correlation method, we present the results of this method when applied to an ATSC DTV signal with multipath components that cannot be resolved in figures 2 and 3 below. The signals are interpolated 10-to-1 above the Nyquist sampling frequency, and the remainder of the signal (containing simulated data) is included in the correlation. This is done for both the Gaussian signal and for the ATSC DTV signal in figure 2 and figure 3, respectively. The figures show a close view of a peak, highlighting the correlation method's inability to resolve the different signal components.

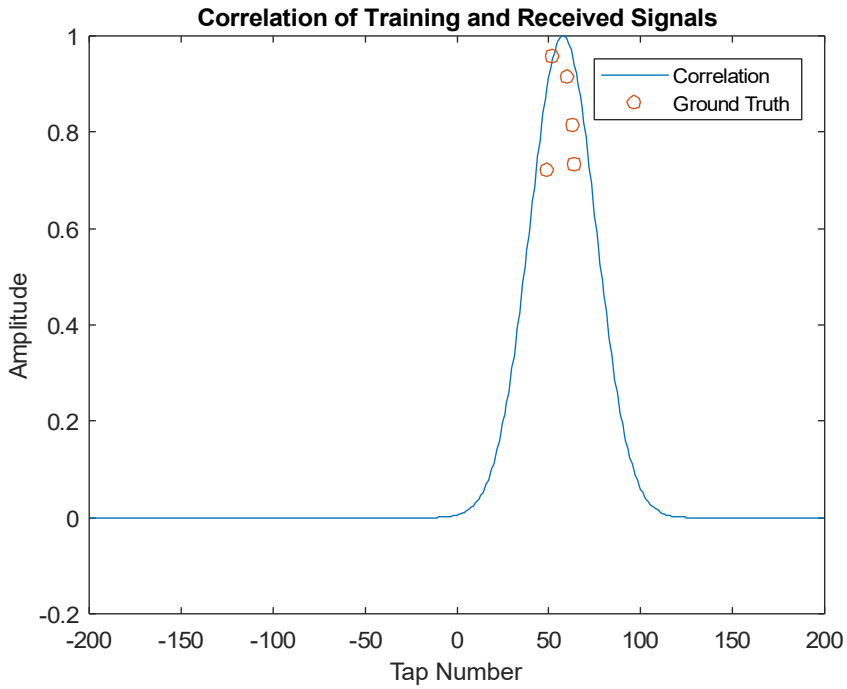

Figure 2: An attempt of the correlation method to resolve closely-spaced multipath when the signal is a Gaussian pulse.

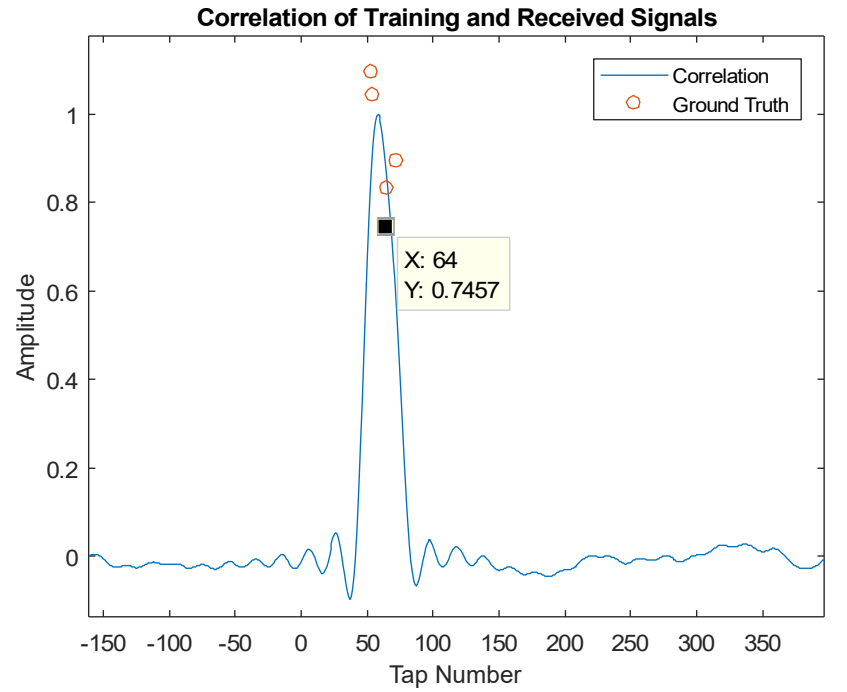

Figure 3: An attempt of the correlation method to resolve closely-spaced multipath when the signal is an ATSC PN511 code.

Least Mean Squares

A way one can approach a ranging problem is through adaptive filters, most notably, the least mean squares filter (LMS). The LMS is an algorithm that attempts to mimic an unknown system by updating and converging the weighting coefficients of a filter such that the mean squared error between the output and the received signal is minimized. It was developed by Professor Bernard Widrow and his student, Ted Hoff, at Stanford in 1960. The least mean squares filter is a stochastic gradient descent method; in other words, the coefficients update based on the values at a single time [9].

The least mean squares filter is an adaptive filter, in the sense that it is a linear FIR filter with adaptive weights, and its weights are adapted in order to minimize the mean square error between the output of the FIR filter for a given input and a desired output. In this regard, this problem is similar to a Wiener filter problem, although they are solved differently. The LMS filter is defined particularly by its update mechanism [9].

Wiener filters are solved by setting up a linear matrix equation relating the two signals such that performing a matrix pseudo-inversion yields the vector of weights that minimizes the expected value of the mean square error. This is done by taking the gradient of the expected value of the mean square error and setting it to zero, giving the system of equations [9].

To set up the Wiener filter, we begin by viewing the expected value of the mean square error, as seen in equation 2 below, and expanding it into three smaller expectation values. We want to take the gradient of this and set it equal to 0. We show this by finding the derivative with respect to each index/element of the filter tap weight vector, as represented by the arbitrary element *i* in equation 3 below. By defining the correlations in equation 4 and equation 5, we can write the derivative as in equation 6, and set it equal to zero to yield the minima. Setting it to

zero gives the relation shown in equation 7, which can be written in matrix form. Solving this system of equations using a pseudo-inversion method yields the optimal set of weights. In the derivation, N is the size of the filter, d is the received signal, and x is the sent/training signal [9].

As a result of the matrix pseudo-inversion, solving the Wiener filter has a relatively high complexity [15].

$$E[e^2(n)] = E[(y(n) - d(n))^2] = E[y^2(n)] + E[d^2(n)] - 2 * E[y(n) * d(n)]$$
$$= E[(\Sigma_{i=0}^{N}(a_i * x(n-i)))^2] + E[d^2(n)] - 2 * E[\Sigma_{i=0}^{N}(a_i * x(n-i) * d(n))] \quad (2)$$

$$\frac{\partial}{\partial a_i} E[e^2(n)] = 2 * E[\Sigma_{j=0}^{N}(a_j * x(n-j)) * x(n-i)] - 2 * E[x(n-i) * d(n)]$$
$$= 2\Sigma_{j=0}^{N}(E[x(n-j) * x(n-i)] * a_j) - 2 * E[x(n-i) * d(n)] \quad (3)$$

$$R_x(m) = E[x(n) * x(n+m)] \quad (4)$$
$$R_{xd}(m) = E[x(n) * d(n+m)] \quad (5)$$

$$0 = \frac{\partial}{\partial a_i} E[e^2(n)] = 2 * \Sigma_{j=0}^{N} R_x(j-i) * a_j - 2 * R_{xd}(i) \quad i = 0, 1, ..., N \quad (6)$$

$$\Sigma_{j=0}^{N} R_x(j-i) * a_j = R_{xd}(i) \quad i = 0, 1, ..., N \quad (7)$$

$$\begin{bmatrix} R_x(0) & R_x(1) & \cdots & R_x(N) \\ R_x(1) & R_x(0) & \cdots & R_x(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ R_x(N) & R_x(N-1) & \cdots & R_x(0) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} R_{xd}(0) \\ R_{xd}(1) \\ \vdots \\ R_{xd}(N) \end{bmatrix} \quad (8)$$

$$\bar{T} * \bar{a} = \bar{v} \quad (9)$$
$$\bar{a} = \bar{T}^{-1} * \bar{v} \quad (10)$$

The LMS filter attempts to solve a similar starting problem iteratively. Rather than setting the gradient equal to zero to find the weights that yield the minimum mean square error, we find the optimum weights by updating them in the direction of the gradient, i.e., the direction of steepest descent. When performed iteratively, this converges to the solution of the Wiener filter [9]. However, in the online LMS version, which is of most interest to us, we make the approximation that at each moment in time n, the expected value of a term is the value of the

term at the time *n*. This essentially makes this a stochastic gradient update.

Since it will be important to this discussion to understand how the LMS filter update equations are derived, we go through the process [9]. We first define *NumTaps* as the amount of taps in the LMS filter, **h** as the weights vector, d as the received signal vector, y as the received signal predicted on the basis of the channel response and training signals, e as the error, and **x** as the vector of the most recent *NumTaps* values of the training signal. *LenSig* is the length of the signal that will be processed. We begin the derivation with the equations representing error and the cost function. The error is the difference between the received signal and the predicted signal, and the cost function is the expected value of the mean square error, as shown in equations 11 and 12. We take the gradient of this cost function. Equation 13 shows the general formula for gradient descent that we are to apply. Equation 14 and Equation 15 show the gradient with respect to the elements of the weights vector, **h**, of the cost function. Equation 16 shows the very important and defining approximation of the LMS: approximate the expectation value at *n* of the signal over the full range of samples, from 1 to *NumTaps*, with only the contribution of the terms at time *n*. This gives the update for the weight vector shown in equation 17. We will refer to this approximation as the LMS approximation or the Stochastic Gradient approximation. This approximation essentially allows the algorithm to be run in real-time because the update for each weight depends only on its signal value that is being stored in the delay line and does not require holding and using the entire cut of the signal.

$$e(n) = d(n) - y(n) = d(n) - \hat{h}^H * \bar{x}(n) \qquad (11)$$

$$C(n) = E[|e(n)|^2] \qquad (12)$$

$$\hat{h}_{n+1} = \hat{h}_n - \mu * \nabla C(n) \qquad (13)$$

$$\nabla_{\hat{h}} C(n) = \nabla_{\hat{h}} E[e(n) * e(n)^*] = 2 * E[\nabla_{\hat{h}}(e(n)) * e(n)^*] \qquad (14)$$

$$\nabla_{\hat{h}}(e(n)) = \nabla_{\hat{h}}(d(n) - \hat{h}^H * \bar{x}(n)) = -\bar{x}(n) \qquad (15)$$

$$\nabla C(n) = -2 * E[\bar{x}(n) * e(n)^*] \approx -2 * \bar{x}(n) * e(n)^* \qquad (16)$$

$$\hat{h}_{n+1} = \hat{h}_n - \mu * \nabla C(n) = \hat{h}_n + \mu * \bar{x}(n) * e(n)^* \qquad (17)$$

A block diagram showing the process is shown in Figures 4 [16] and 5 below. Two signals, a training and a received signal, are inputted. The training signal is put through a delay line that stores and carries over the signal at each unit of discrete time, n. At every moment in time, each value stored in the delay line is multiplied by a weighting coefficient associated with its position in the delay line. All of these are added, resulting in the output of the filter at that moment in time. This is compared to the received signal at that moment in time to get the error through integer subtraction, which is then used to update the weights.

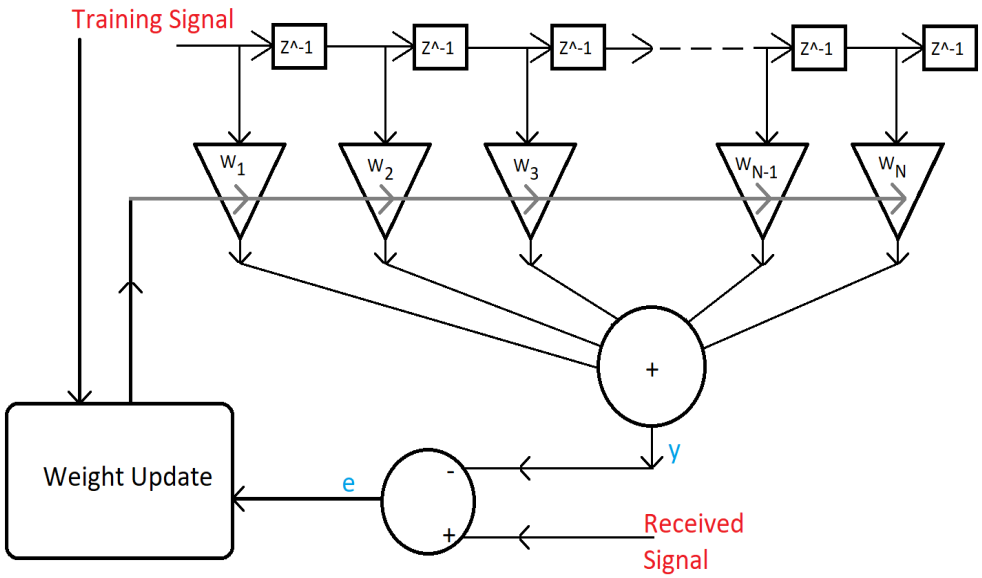

Figure 4: A block diagram of the least mean squares equalizer.

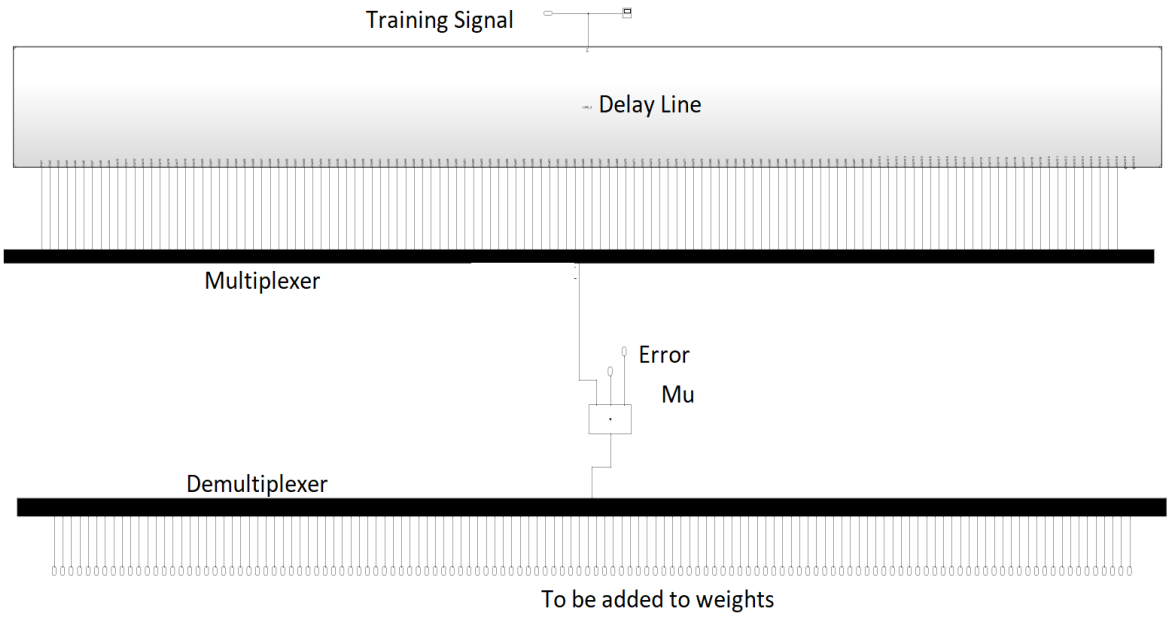

Figure 5: An implementation of the weight setting block in Simulink.

A key advantage of the LMS filter is its simplicity. The LMS converges to the Wiener filter solution by applying a stochastic gradient approximation. The LMS requires (2* *NumTaps* + 1) additions and (2* *NumTaps* + 1) multiplications per cycle [12],[15]. This leads to a computational complexity of operational amount in the order of *LenSig* NumTaps* and to a space complexity in the order of *2* NumTaps.* More importantly, this reduced operational complexity is a very big advantage because it allows the algorithm to be implemented in parallel as an ASIC or in an FPGA, greatly lowering the computation time. A parallel implementation of the LMS filter gives it an effective time complexity of *LenSig* and space complexity of *2* NumTaps*. The effective time complexity relates to the amount of time it takes to compute considering all of the operations that are occurring in parallel, and the space complexity comes from the delay-line and the multiplication hardware.

To range using the LMS, we can run the LMS so that it finds the weights when the input is the training signal and the output is the received signal. Since the position in the delay line is essentially the training signal at a time of {- index} + n, then one can understand the value at

index *t* as being what a signal delayed by *t* time-units relative to a reference signal would look like arriving and interfering with signals arriving at other delays. Therefore, ideally, the weights would converge such that the weights at the indexes corresponding to the delays of these signals or multipath components are equal to the amplitudes of these components, and all other weights are zero. Another way to understand it is that the adaptive filter is trying to converge to the weights that when convolved with the training signal give the received signal; this is the channel response vector that contains the information of where the LOS and NLOS signals are.

Again, to be ideal, this method has similar precision requirements as the correlation method; namely, that if a signal arrives with a certain delay, then it only appears in the filter weight vector as one weight. This once again is inaccurate because of the linearity of this system and the bandwidth limitations. Since the signals are band-limited and all of the operations are linear, then during updates, the weight coefficients vector only adds band-limited updates and does not develop any non-zero higher-frequency components, hindering its ability to reproduce Kronecker delta functions in the channel response and confining it to a series of sinc functions representing the filtered delta functions. In the frequency domain perspective, when the weights are convolved with a band-limited training signal, thereby multiplying in the frequency domain, only the low-frequency components of the weight vector are not multiplied by zero, and so as long as these are correct, the process converges. The result is a large number of local minima corresponding to the combinations of higher-frequency components of the weight vector with the proper lower-frequency components. Since the weights are generally initialized to zero, the higher-frequency components remain at zero. This, in turn, leads to the same bandwidth limitation experienced by the correlation method.

A demonstration of the LMS filter's abilities, and shortcomings, in ranging is now

presented. In figure 6 below, we see demonstrations of the channel response output of the LMS filter when the ground truth channel response Kronecker delta functions are well separated, when the separation is about the Nyquist resolution, and when the separations are below the minimum resolution. The bottom left corner shows the signals associated with the unresolvable test-run. We see that the band-limited signals themselves do indeed converge, but that this does not mean anything for the higher-frequency information of the channel response.

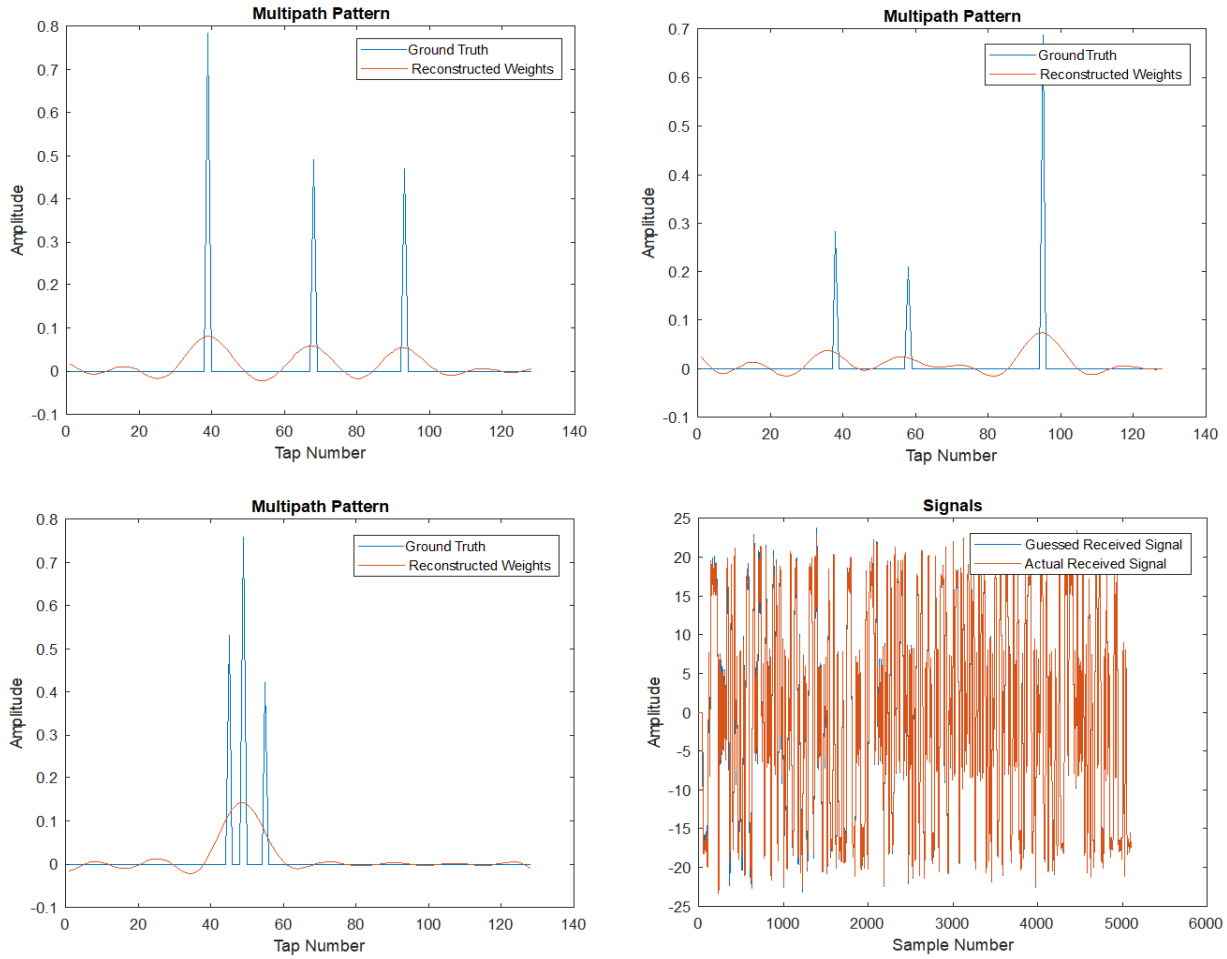

Figure 6: The channel responses predicted by LMS when the pulses in the ground truth vary in separation in the top half and in the bottom corner. The bottom right portion compares the reconstructed signal with the received signal in the setup of the bottom left.

Super-resolution Algorithms

Given the limitations of the linear methods presented, we examine super-resolution algorithms in order to understand the mechanisms through which they can obtain super-resolutions, in hopes of using their principles to obtain a super-resolution while maintaining some of the advantages of the LMS filter. The methods discussed include MUSIC/ESPRIT, compressive sensing, and Maximum Likelihood.

We begin by discussing MUSIC and ESPRIT. MUSIC and ESPRIT are used for frequency, time-of-arrival (ToA) estimations, and direction-of-arrival (DoA) estimations [17], [18],[20],[21],[28], and they provide significant insight into the principles behind super-resolution algorithms. MUSIC, which stands for Multiple Signal Classification, was developed in the 1970s. MUSIC is a solution to a family of problems in which one tries to estimate a set of parameters that the received signal depends on. MUSIC accomplishes this by exploiting knowledge of the structure of the data and the sensor arrays, and is commonly applied to frequency estimation in the presence of White Gaussian noise. MUSIC works by taking an eigenvalue decomposition of the autocorrelation matrix, and by knowing that the number of complex-exponential signals is p, then the p largest eigenvalues span the signal subspace and the remaining eigenvalues span the noise subspace. This provides us with a set of frequencies for the signals that are not bound to the frequency bins, allowing for a more accurate estimation [17], [18],[20]. This algorithm has a complexity of $O(N^3)$, where N is the total amount of samples in the window [30].

We now discuss compressive sensing and its assumption of constraints on the system. Compressive sensing is based on setting up a system of underdetermined linear equations, which typically have infinitely many solutions, and finding the correct solution to reconstruct a signal.

In order to be able to find a solution of this system of equations, the signal needs to be sparse in some domain and needs to be incoherent, allowing us to specifically recover the sparse solution. We recover the sparse signal by performing L1 minimization techniques. This makes compressive sensing distinct from L2 minimization, or in other words, least squares minimization. The L*n* norm is the square-root of the sum of all of the absolute values of a vector, each to the power of *n*. The L1 norm is therefore the sum of all of the absolute values, and although the L0 norm is not rigorously defined this way, it is the sum of all nonzero elements in the vector. The L1 norm's gradient in all domains linearly leads to the global minimum, alleviating concerns over a local minimum. Furthermore, in most problems, the L1 norm is equivalent to the L0 norm, so the L1 norm minimum also represents the sparsest solution [8].

There are several methods used to solve for the solution of an L1 minimization problem. These include matching pursuit [23],[24], basis pursuit [25],[26], LASSO [22], and others. There are also iterative solutions for the sparse solution. Compressive sensing has a complexity of $O(N^3)$ [30], where N is the number of samples in the window of interest.

Another super-resolution algorithm worth discussing is maximum-likelihood [28],[29], [14]. This method is based on a grid-search. With this, the number of pulses is known, and it essentially runs through all possible values and keeps track of which is statistically the likeliest. It has a complexity of $O(M^2*N)$, where M is the number of NLOS signals and N is the number of samples [30].

Method

Having established the principles by which super-resolution algorithms operate, we now attempt to form a set of constraints and apply them to the LMS equalization structure in hopes of being able to maintain many of the advantages of the LMS structure while also enforcing the constraints that would allow for a super-resolution.

To choose how we enforce the constraints, we take into account that we want to base our approach on the LMS filter. Since the LMS filter is itself based on an approximation, we want to select a criterion that we can deal with more easily. Optimization for sparsity in the super-resolution algorithms uses mathematics that is too sophisticated for clear use with the LMS algorithm, so we look at the assumption that we know the number of pulses in the channel response. We pick this because it is much more physically simple to deal with this constraint and handle a channel response vector that has a set amount of relevant nonzero values in an algorithm than it is to exploit the mathematical properties of sparsity (or sparse matrices).

By setting this constraint and solving this problem, we are no longer solving indoor localization; rather we are solving a problem that may map certain indoor localization problems, but is a more canonical problem that may be useful in indoor localization systems. How exactly to use this for practical localization, where the channel vector is made up of an unknown amount of pulses, or comprises of smoother shapes, is viewed as a different problem that would make use of this, just as any realistic problem makes use of canonical problems. In practice, this may be by running this algorithm multiple times and varying the number of pulses in the constraint, or by replacing the delta functions with a narrow Gaussian or Lorentzian pulse, or by experimenting with different environments and making case-by-case modifications to the canonical problem.

Having defined the problem, we will first define the relevant variables before explaining our approach. The number of pulses in the channel response is defined as p. The number of taps in the filters that we will refer to is *NumTaps*. The length of each signal is *LenSig*, and the amount of times the algorithm is repeated on a signal (indicating the amount of times the receiver needs to receive a signal) is R.

In order to enforce the constraint on the LMS filter, an obvious modification to the LMS is to have a separate channel weight update function that takes the channel vector as its input and outputs a modified version that is closer to the constrained result. However, it is difficult to find the correct function to apply on the weights. The function has to be simple and relatively linear. For most cases, it is difficult having an independent function that updates the weights accurately without taking the information from the LMS update pattern into account. Although there may be a way to effectively update the channel response this way, it is clear that we should look elsewhere.

The LMS algorithm we used is based more on writing the multipath pattern as a vector containing delays and a vector containing weights for each multipath component (with the exception that all components at a given delay are combined into one element). Thus, a channel response vector with five pulses is written as two five-element vectors. The idea, then, is that rather than enforcing the constraints by trying to make the equalizer organically converge to a channel response vector with the decided amount of pulses, we force the constraint by defining the channel response domain only in terms of five pulses, which we then converge to the correct solution using convergence techniques based on the LMS. The problem can be approached much more simply when written in this form, because it is very difficult to come up with a function that finds pulses in channel response domain and shifts them.

We perform this by essentially having two LMS-based update systems: one updating the weight vector, and one updating the delay vector. The updates are intertwined as part of the same iterative loop and the same system of variables at each unit of time. The amplitude and delay vectors both update in the same iteration of the loop using the same error value, which is derived from both the amplitude and the delay vectors at the time of interest. Furthermore, the update-function may call upon both vectors in order to update the delay and the amplitude.

The idea behind this is that we set up separate least mean squares updates for the delay and for the amplitudes so that the system converges to the corresponding amplitude and delay vectors that minimize the mean square error. In the derivation, x(n) is the sent signal, d(n) is the received signal, y(n) is the predicted signal, a_i is the amplitude vector of the equalizer, b_i is the delay vector of the equalizer, *p* is the size of the amplitude and delay vectors (which represents the amount of pulses in the channel response vector), and e(n) is the error. We assume that the expected value of the noise is zero, so we ignore its contribution. To set up the system, we write the error as a function of both the amplitude and the delay vector, as shown in equation 18. We take the derivative of the cost function with respect to an arbitrary element i of the amplitude vector and the delay vector, as seen in equations 19 and 20. This gives us the updates equations shown in equations 21 and 22.

$$e(n) = d(n) - y(n) = d(n) - \Sigma_{i=1}^{P} a_i * x(n - b_i) \qquad (18)$$

$$C(n) = E[e(n)^2] \approx d(n)^2 + (\Sigma_{i=1}^{P} a_i * x(n - b_i))^2 - d(n) * \Sigma_{i=1}^{N} a_i * x(n - b_i) \qquad (4b)$$

$$\frac{\partial C(n)}{\partial a_i} = E[(\frac{\partial e}{\partial a_i}) * e^*] \approx e^* * [0 - x(n - b_i)] \qquad (19)$$

$$\frac{\partial C(n)}{\partial b_i} = E[(\frac{\partial e}{\partial b_i}) * e^*] \approx e^* * [0 - a_i * \frac{\partial x(n - b_i)}{\partial n} * \frac{\partial (n - b_i)}{\partial b_i}] = e^* * (-a_i) * x'(n - b_i) * (-1) \qquad (20)$$

$$a_{i+1} = a_i + \mu_a * e^* x(n - b_i) \qquad (21)$$

$$b_{i+1} = b_i - \mu_d * e^* * a_i * x'(n - b_i) \qquad (22)$$

This derivation depends on the derivative of the training signal. Since we want this algorithm to be linear and discrete, we estimate the derivative of the training signal at each point as the difference between adjacent points in the training signal. This provides an estimate of a scaled version of the derivative. Note that since the LMS structure does not have infinitely small time bins, the oversampling needs to be high in order for this to be able to approximate a continuous time variable accurately. With that in mind, the update equations can be written as shown in equations 23 and 24.

$$a_{i+1} = a_i + \mu_a * e^* x(n - b_i) \qquad (23)$$

$$b_{i+1} = b_i - \mu_d * e^* * a_i * [x(n - b_i + 1) - x(n - b_i)] \qquad (24)$$

The algorithm can be summarized as shown below:

Our Algorithm

Inputs: Received signal d(n); training signal x(n); step sizes for the amplitude and delay μ_a and μ_d; number of pulses in the channel response P; total length of signal (including repetitions) R

Initialization: attenuation a_i to arbitrary values; delays b_i to values within the estimated range of the delays.

Output: attenuation a_i; delays b_i; error e(n)

Iteration: n=1:R

$$e(n) = d(n) - y(n) = d(n) - \Sigma_{i=1}^{P} a_i * x(n - b_i)$$

Find the amplitude and delay updates and update based on equations 23 and 24

Unlike the LMS, in this algorithm, the delays change as we update. With obvious physical constraints regarding time and clock operation in mind, it is clear that replacing the *NumTaps* size unit-tap delay line with a *p* tap delay line with tap delays corresponding to the delay vector is not an option due to information lost as delays change and due to difficulty in implementation. However, the solution is much simpler and closer to the original LMS. By connecting the signals we select from to a delay line, we essentially capture the information at every clock cycle within that time period. Then, each delay of the delay line can be connected to a selector (or multiplexer), allowing us to feed the selector the values of the delay vector to obtain the signal at that delay. Note that this caps the delays in the delay vector to the length (or maximum delay) of the delay line. In our implementation, we use 2 delay lines: one with an input of the training signal at a certain time *n*, and the other having an input of the training signal at time *n-1* subtracted by the training signal time *n*. This allows us to easily select from a training signal delay line and from a derivative delay line.

By making the proper approximations regarding the derivative, we have kept the algorithm linear and in a close parallel to that of the original LMS filter. An important advantage to this algorithm is that in hardware implementation, we require much less storage for the values of all of the weights and far fewer multiplication operations because we no longer have to multiply all of the outputs of the delay line with their respective weights. This may prove very valuable even outside the Fourier resolution if the person requires many taps in the filter/delay line to capture a wide range of distance/time but does not want to dedicate too much hardware resources to memory (RAM) or multiplication.

The initialization of the delay and amplitude vectors is very important in this algorithm. The values they are initialized to matter a lot in where and if the algorithm converges. We

initialize the delays such that they are in the vicinity of the received pulse. We do so by finding the peak of the received Gaussian or correlation of the ATSC signal, and then going back several samples and initializing the LOS delay to be that sample. The other delays are initialized by linearly adding a small amount to each adjacent delay value such that all of the delays are essentially within the pulse. The amplitudes are all initialized to a value close to the average value of the ground truth amplitudes. In a realistic system, this can be estimated based on signal power.

Our algorithm has a complexity in the order of $p*LenSig$. This is due to the selection of the values in our implementation before multiplication and addition. When implemented in parallel, it runs for a period of time in the order of $LenSig$. It has a spatial complexity in the order of $NumTaps + p$.

Simulink

In order to demonstrate how this algorithm can be implemented in parallel, we use Simulink to build this algorithm and to test it and measure its capabilities. We will choose one of a variety of different implementations of this algorithm, and we will present this implementation in this discussion. There are a variety of implementations because certain operations can be written in more than one way, and the way one chooses to implement the algorithm ultimately depends on resource availability trade-offs, which matter significantly for someone wishing to implement this algorithm in an ASIC or an FPGA.

An important implementation decision is that of the derivative of the training signal. We defined the linear approximation of the derivative of the training signal as the difference of the values of the training signal at adjacent samples; when we look at this as literally defined in the

MATLAB code, this can be implemented by placing a subtraction operation between every two adjacent taps of the delay line. However, we realize that this is a redundant process, since the derivative shifts as the training signal shifts in the delay line, leading to the subtraction of the same pairs of numbers occurring *NumTaps* number of times. In our parallel implementation, a single subtraction between two adjacent samples of the training signal occurs before the samples of the training signal and the difference are inputted into the delay lines, with a possible tap-delay before the input for synchronization. This forms a trade-off with regards to physical space in the chip between having many adders and having the extra flip-flops for the other delay line. Overall, an important advantage of our parallel implementation is that a delay line is based on flip-flops, which contain much simpler circuitry compared to adders. This is shown in figure 7 below.

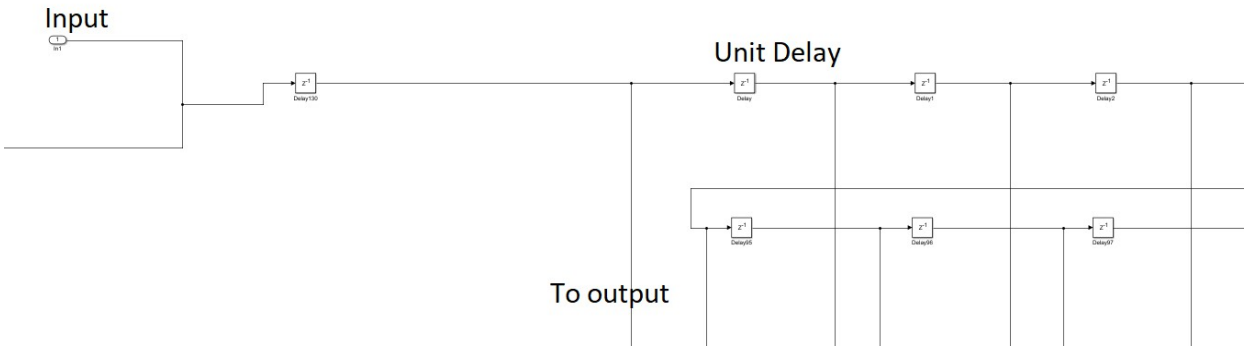

Figure 7: A portion of the delay-line in Simulink. Each square represents a 1-tap delay in the propagation of the signal.

The storage of the delay and amplitude vectors is done using some form of RAM or simple memory circuits. Since there are typically not too many elements in these vectors, this is not a problem for the hardware designer. The memory blocks output the value stored, and are inputted values that they add to their own (not replace with) as an update. This is shown in figure 8 below.

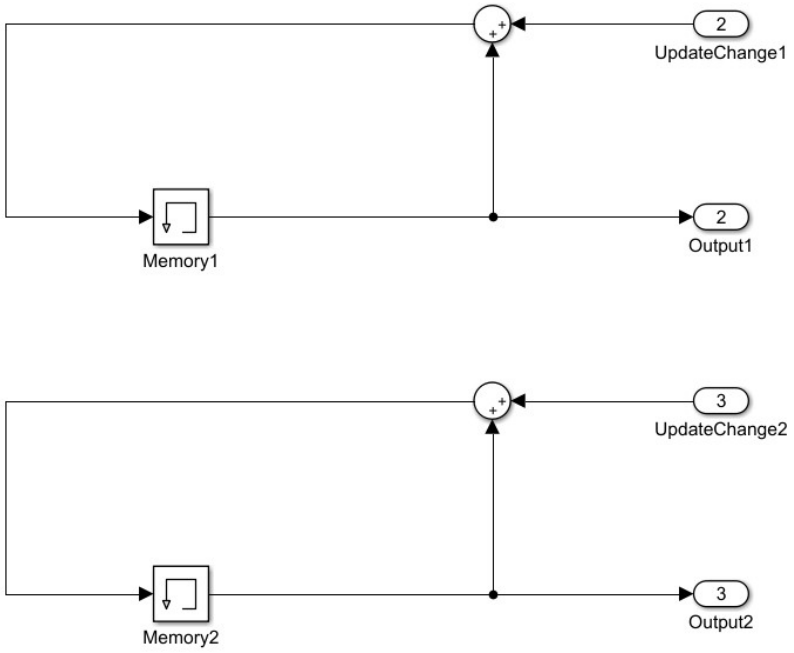

Figure 8: A representative portion of what the memory blocks look like in Simulink.

The design relies on having multiplexers/selectors that connect to all *NumTaps* outputs of the delay lines that outputs the values of the signals in the delays corresponding to the rounded delays in from the delay vector. In our simulation, the selection for all *p* of the delays occurs in one multiplexer operating at a higher frequency in order to save hardware space at the expense of clock-rate. However, one can have *NumMux* multiplexers share the task as well. A multiplexer is shown in figure 9 below.

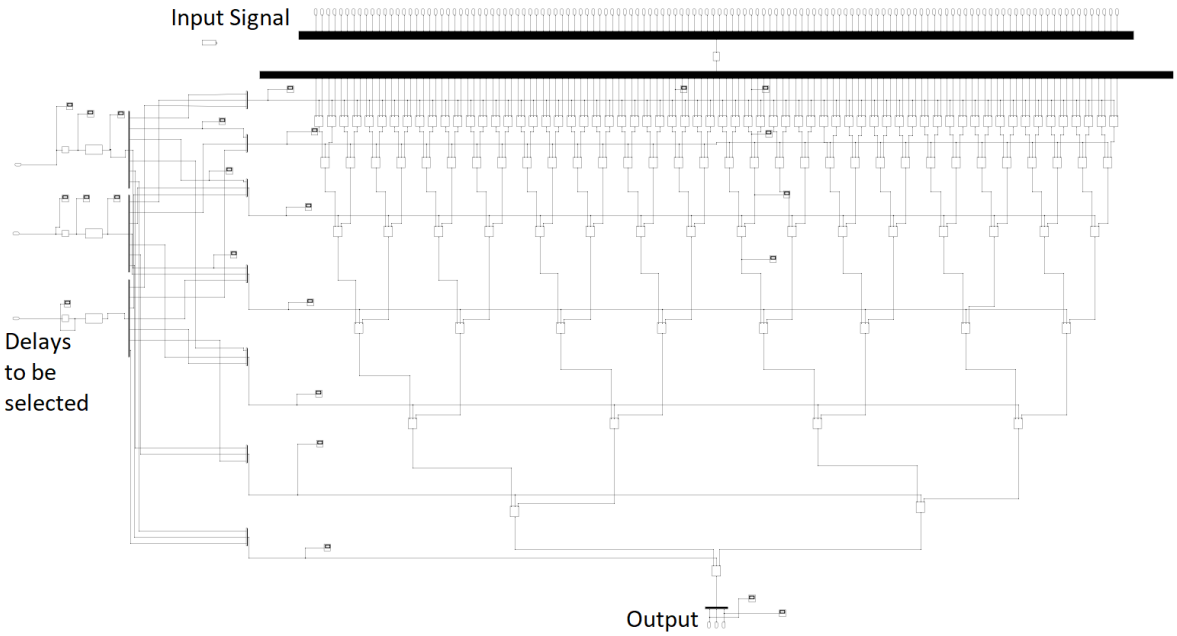

Figure 9: A selector in Simulink. This takes in 128 inputs and selects the 3 defined in the left. It is based on a series of switches.

We have separate amplitude-update and delay-update blocks. Each of these performs its update operations and returns the update to the corresponding memory blocks. These can be seen in figures 10 and figure 11 below.

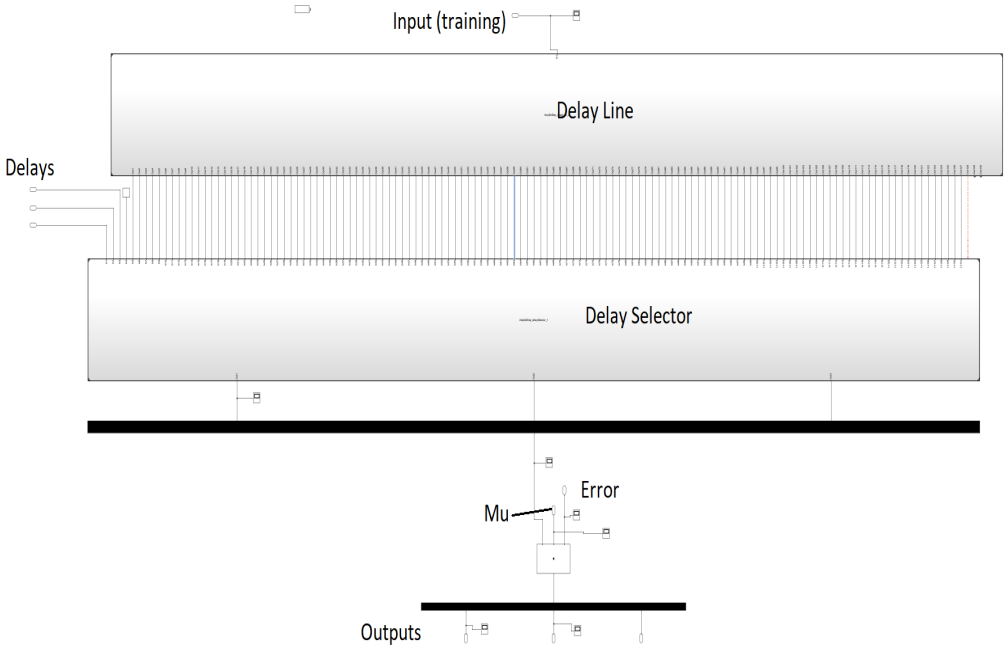

Figure 10: The weight-setting block in Simulink. It is similar to that of LMS, but 3 delays are selected first.

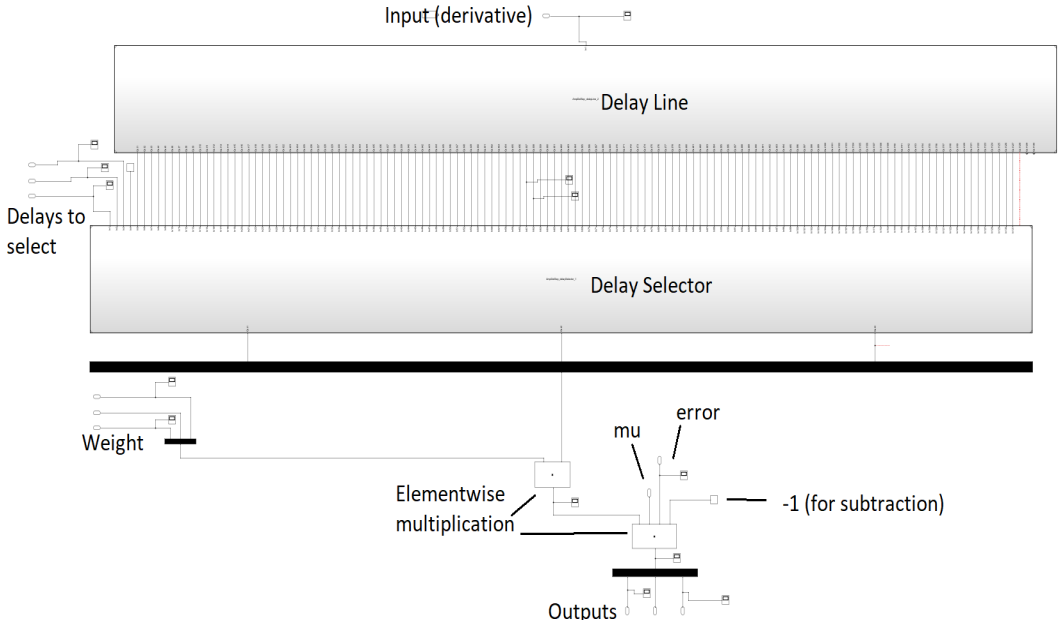

Figure 11: The delay setting block in Simulink.

The error is found in the module shown in figure 12 below.

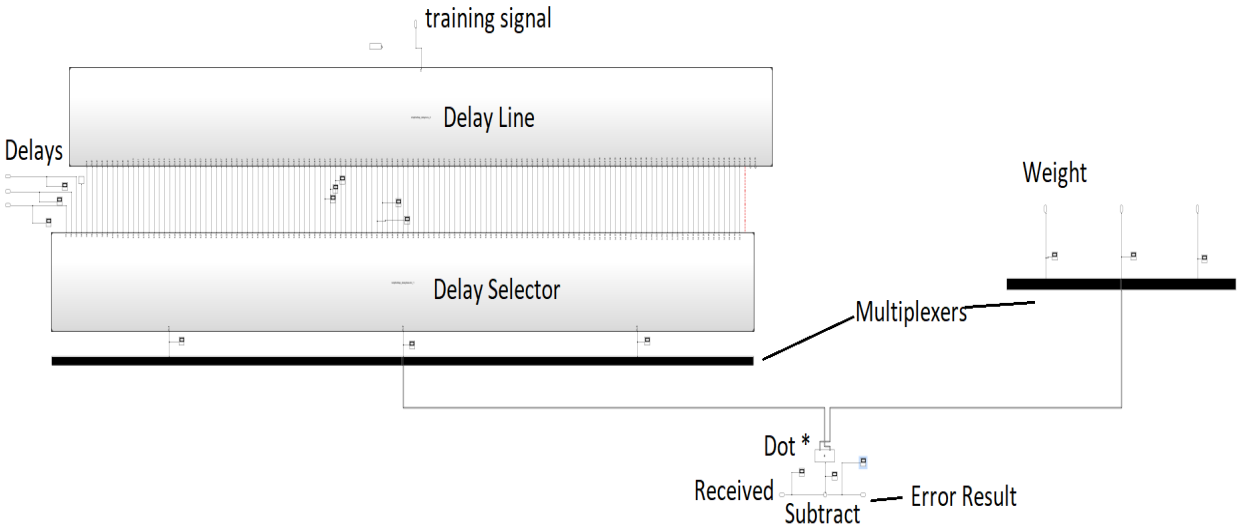

Figure 12: The block that finds the error.

Putting everything together, we get the system shown in figure 13 below. We can see by the extensive utilization of the delay line that this is a parallel implementation, as planned.

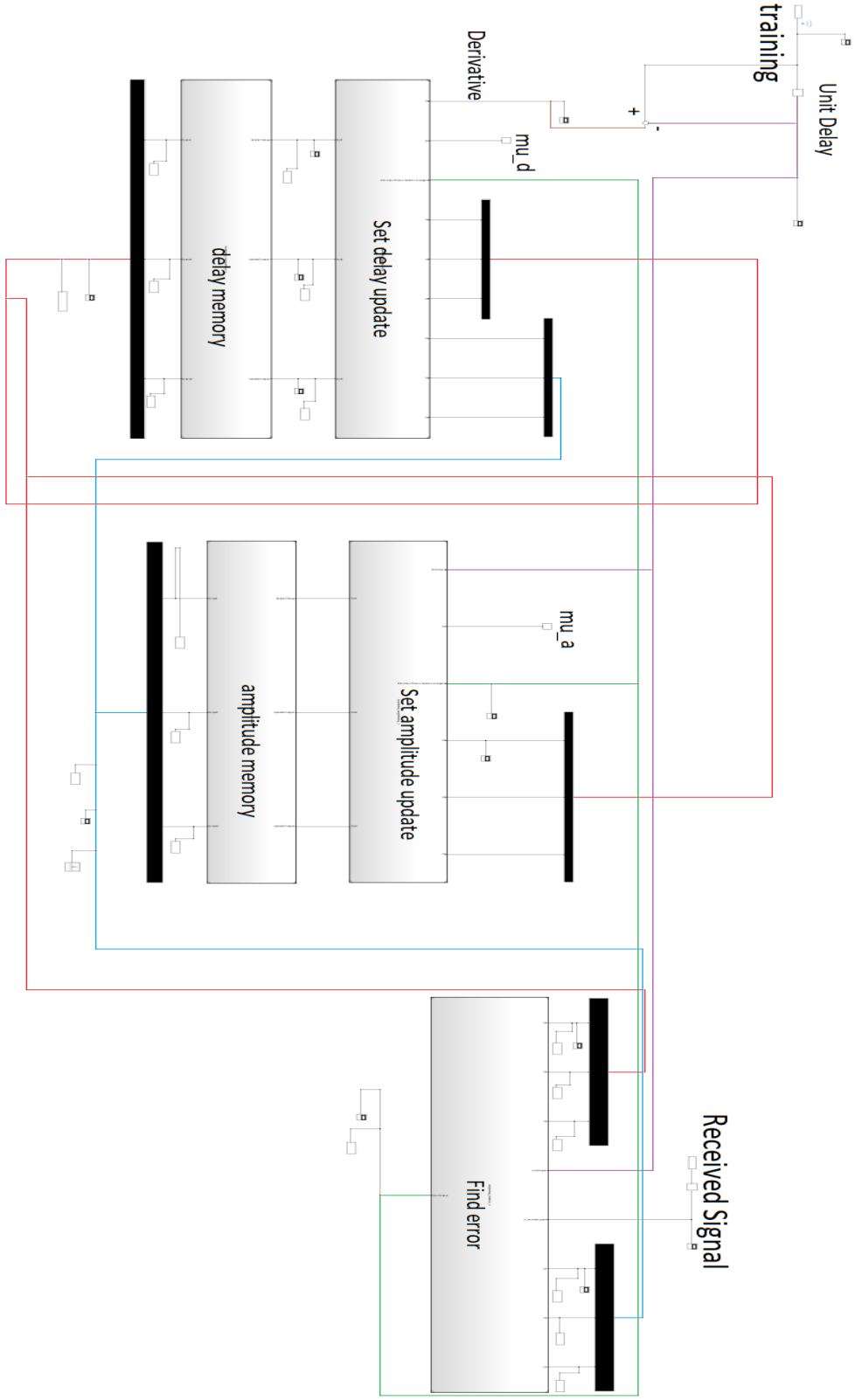

Figure 13: The whole system based on the submodules in Simulink.

It is useful to analyze this implementation by quantifying how quickly weights update with respect to the quickest clock rate in terms of some of the design parameters, such as *NumMux*. Firstly, the amount of clock cycles required to select a full set of signal values from the delay-line is the ceiling of (*p*/*NumMux)*. Then, if the algorithm is run for *R* LenSig* cycles of the algorithm, and each cycle is in the order of the number of cycles it takes to select a full set of signals, then this algorithm could be run with timing in the order of *R*LenSig*p/NumMux*.

Results

Having come up with the algorithm, we must verify it, first through some simulations. The signal we used in our simulations are a Gaussian pulse. It is important that there be a sufficient time-interval between two transmitted pulses and PN signals in order to avoid inter-symbol interference.

In many of our simulations, we assume that there is no noise. Since we are trying to judge the algorithm as a super-resolution algorithm, it is sufficient for us to examine the performance when there is no noise. In some of our simulations, we address noise. The noise may be additive White noise, mainly stemming from the thermal noise added at various processes in the receiver. White noise is completely random and incoherent; it theoretically exists randomly with a Gaussian distribution of power in all frequencies, but realistically, due to filtering in the receiver, it is random and incoherent but within the bandwidth. Another consideration is colored noise. This can stem from various sources, but one consideration in our simulations is noise in the channel response vector. In our noiseless simulations, the channel response vector used to generate the received signal contains an amount p of perfect Kronecker delta functions, with all other values being zero. However, it may be the case that there are p main pulses but the other values are not all zero. This results in colored noise at the receiver, because the noise is in the pattern of the received signal. In the simulations that take this noise into account, we assume that each otherwise empty bin has a small random value defined by a Gaussian distribution centered at zero added to it.

In our tests of the algorithm, we did not include any noise. Although noise is an important part of every system, our goal was to test the convergence of the algorithm more fundamentally. We ignored noise because we wanted our results to reflect only the core functionality of the

algorithm and the update mechanism.

To test the algorithm, we run the simulations varying the amount of pulses kept. In our simulations, there are 128 taps in the filter. When we defined the Gaussian training signal, we used $\sigma = 12$. This relates to the width of the pulse, and thus the bandwidth. The relationship with the pulse-width is that the FWHM of a Gaussian pulse is given by approximately $2.35482*\sigma$, so FWHM in this case is approximately 28.24 samples. The relationship with the bandwidth is that in the frequency domain, a Gaussian pulse transforms to another Gaussian pulse but with a $\sigma' = 1/\sigma$ (assuming a symmetric range), so the FWHM in the frequency domain of the training signal in our case is approximately .196. Since the pulses are Gaussian, when we try to resolve them, the results can be compared to σ and the FWHM of the training signal. The inability to resolve them through regular LMS is shown by example in figure 14 below, in which there are 5 pulses and $R = 6000$ repetitions.

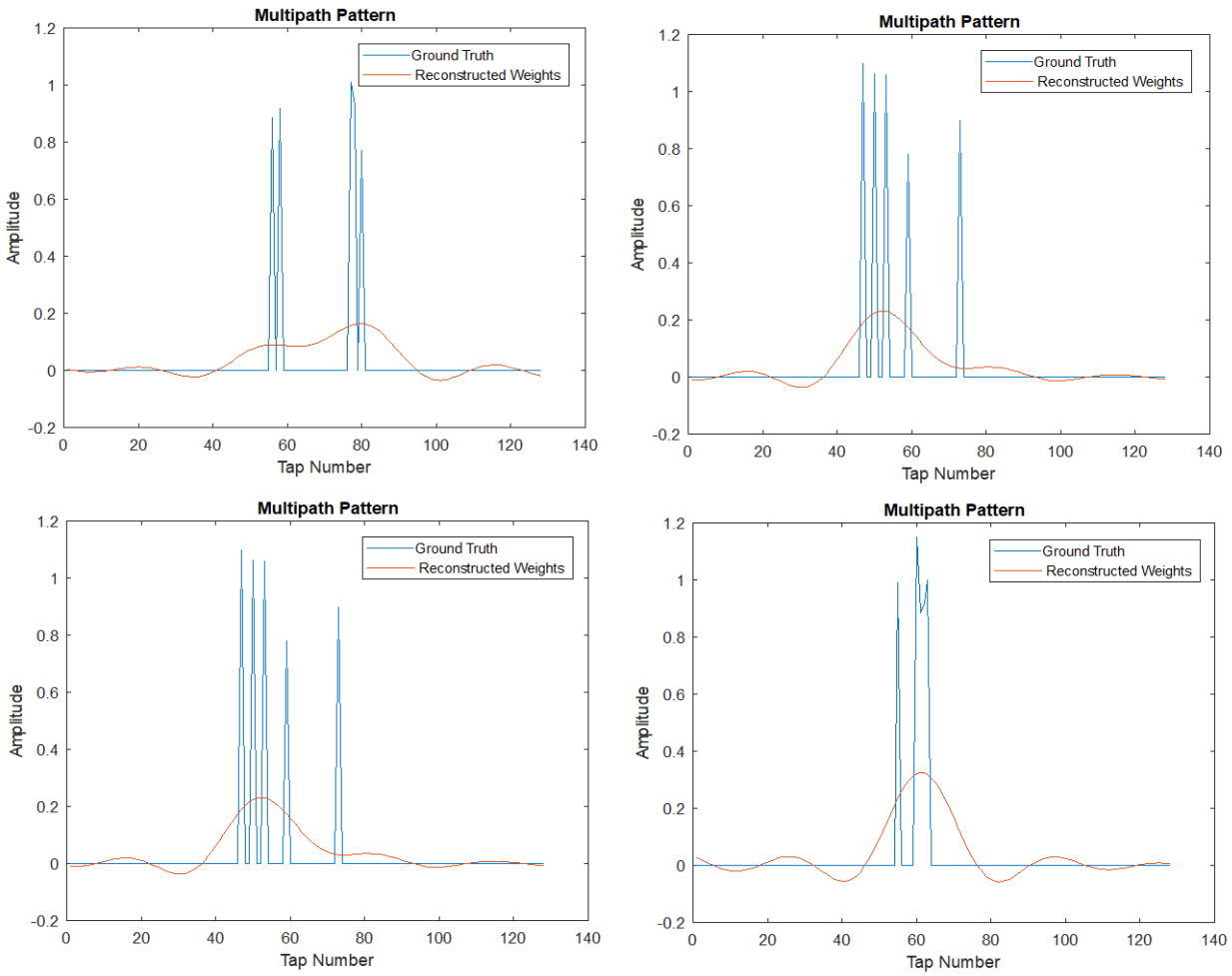

Figure 14: The channel responses LMS converges to when the signals used are Gaussian pulses.

We run the algorithm with p = 1, 2, 3, 5, 7, and 10 pulses. For each of these, we run 100 cases. For each p, we present 4 plots of the channel response vector, showing the ones used to generate the received signals, the amplitudes and delays the algorithm converges to superimposed with a different color, and the starting channel vector in another vector. For each of the 100 test-runs, we look at the delays, compare with the ground truth, find the root-mean-square errors (in unit samples) of the delays of each test-run, and plot a distribution showing how much error there was in estimating the errors. To understand the improvement in accuracy, this root-mean-square error would be compared to the FWHM, which is defined in terms of the σ of

the Gaussian pulses as 2.35482*σ.

The results of this measurement procedure are shown for Gaussian signals. Note that when we plot the channel response vector that the algorithm converges to, we round the delay values to the nearest integer delay in order for it be discretized. Note that this rounding approximation is also done in the algorithm for a decimal delay to represent a position in the delay-line, which is defined by an integer.

Although this algorithm attempts to recover the entirety of the channel response vector, we give some special privilege to the LOS because if an algorithm can estimate the first pulse in the channel response with more accuracy at the cost of the accuracy of some of the other components of the channel response, then it would be very beneficial to our end goals. Therefore, we also plot a distribution showing how much error there was in the delay of the LOS in unit samples by showing the distribution of the absolute values of the error of the LOS component. This also plays into why we care more about the delays that the algorithm converges to than the amplitudes.

Note that when we find the root-mean-square errors, the value presented may be an overestimation of the error in the information. This is because we find the root-mean-square error by performing an element-wise subtraction of the errors, squaring them, adding, and taking the square-root; however, if there is an error related to the numbering of pulses relative to one another, then an element-wise comparison would be less meaningful because having a set of delays that the algorithm predicts perfectly would contribute an error if the algorithm used one pulse to capture two closely spaced ground truth pulses that occur earlier in the channel response. This would be a bigger issue with more pulses. However, viewing the information in a plot allows one to see the accuracy more clearly.

When we perform simulations based on the Gaussian signals, we repeated the algorithm, as indicated by the variable R. We used $R = 4000$ for $p = 1$, $R = 4000$ for $p = 2$, $R = 4000$ for $p = 3$, $R = 10000$ for $p = 5$, $R = 10000$ for $p = 7$, $R = 14000$ for $p = 10$.

In figure 15 below, we see some sample plots of the channel response where $p = 1$.

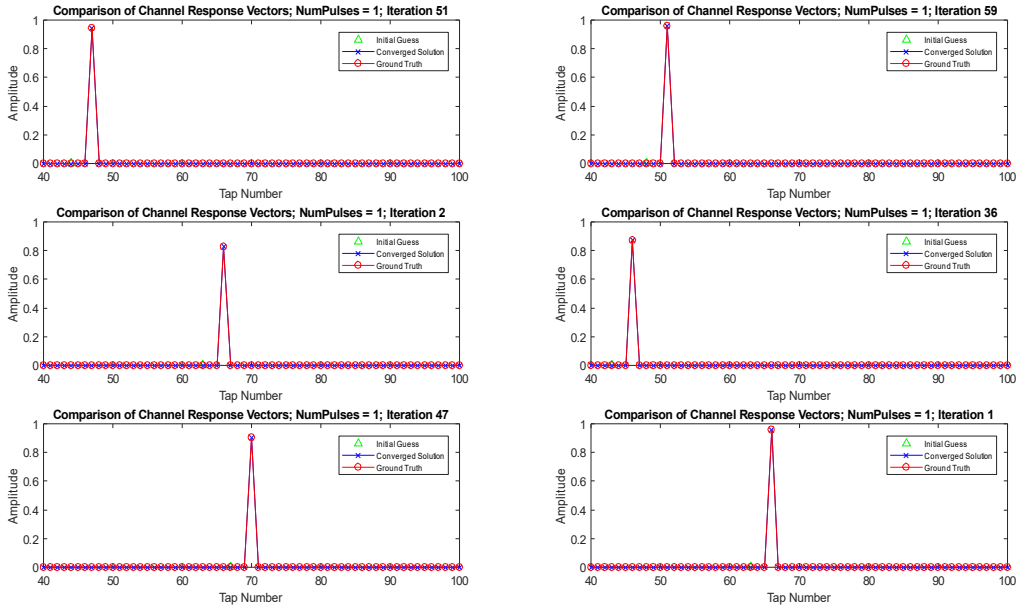

Figure 15: 6 examples of the algorithm's channel response output with p=1.

In figure 16 below, we see some sample plots of the channel response where $p = 2$.

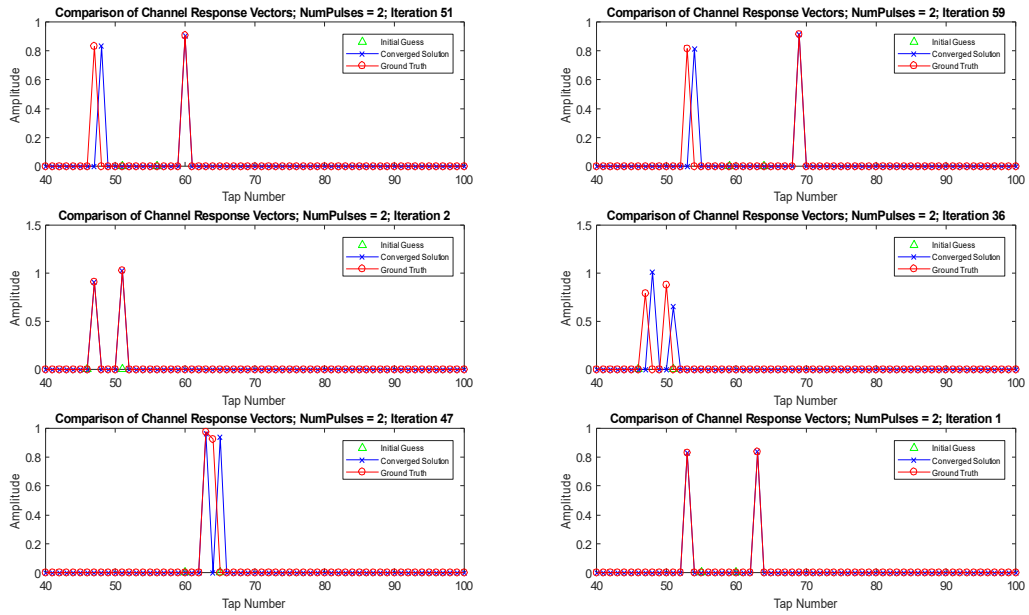

Figure 16: 6 examples of the algorithm's channel response output with p=2.

In figure 17 below, we see some sample plots of the channel response where $p = 3$.

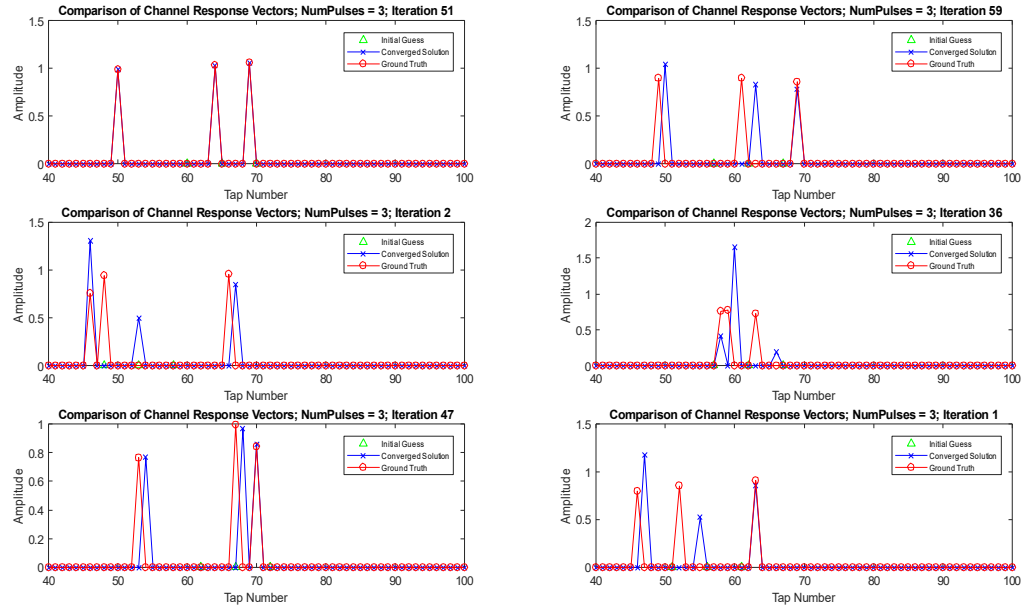

Figure 17: 6 examples of the algorithm's channel response output with p=3.

In figure 18 below, we see some sample plots of the channel response where $p = 5$.

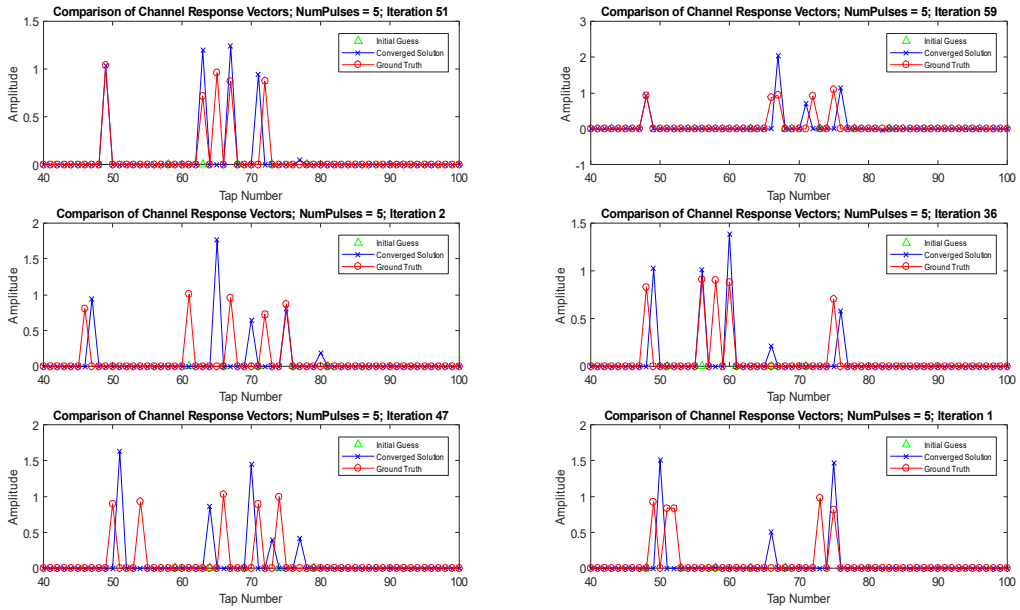

Figure 18: 6 examples of the algorithm's channel response output with p=5.

In figure 19 below, we see some sample plots of the channel response where $p = 7$.

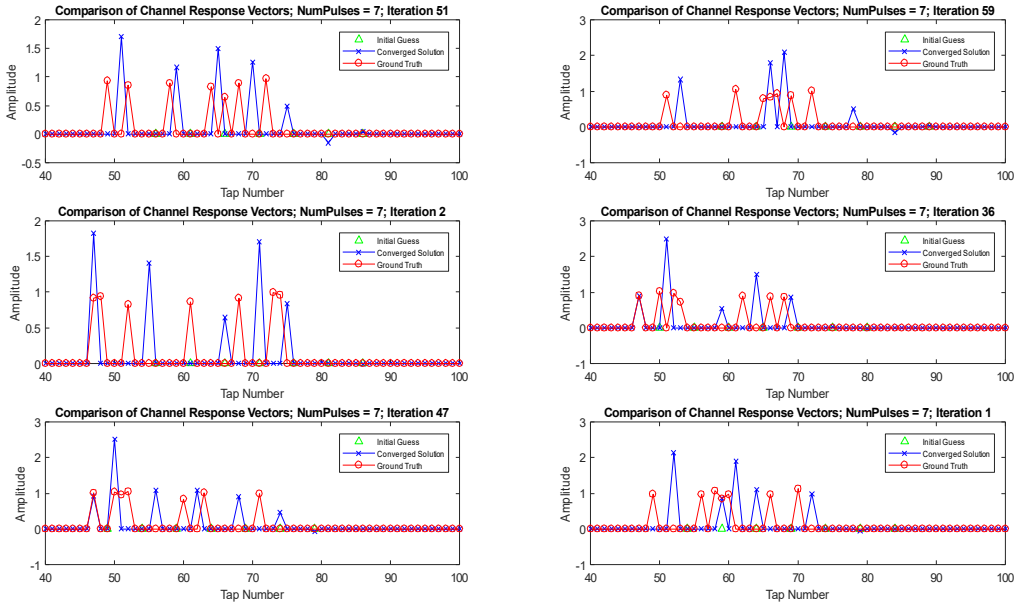

Figure 19: 6 examples of the algorithm's channel response output with p=7.

In figure 20 below, we see some sample plots of the channel response where $p = 10$.

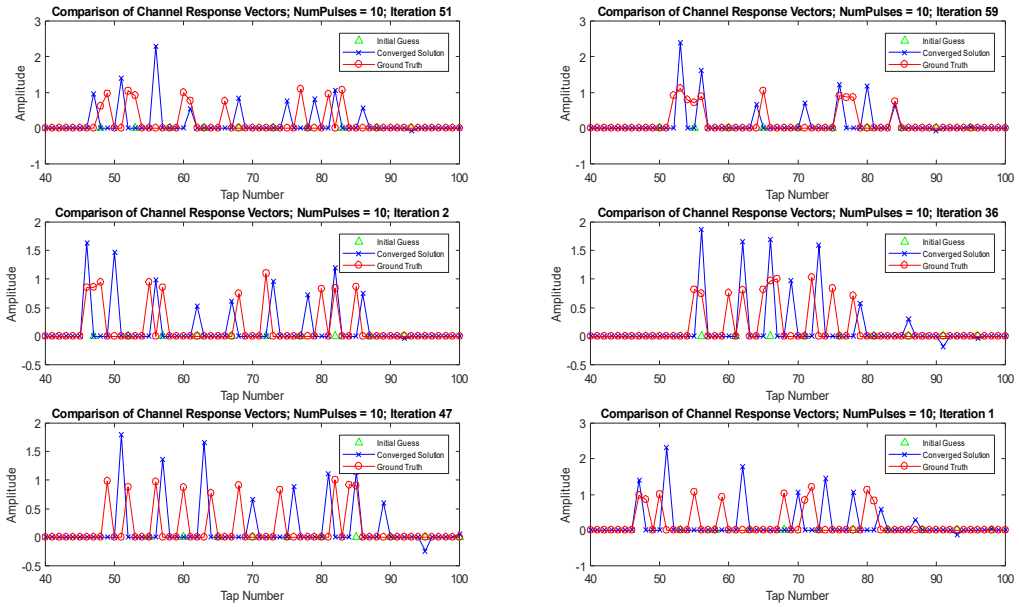

Figure 20: 6 examples of the algorithm's channel response output with p=10.

The plots of the histogram depicting the root mean square error of the time-delays at each value of p are shown together below in figure 21 for ease of comparison.

The plots of the histogram depicting the absolute value of the error of the LOS delay at each value of p are shown in figure 22 below.

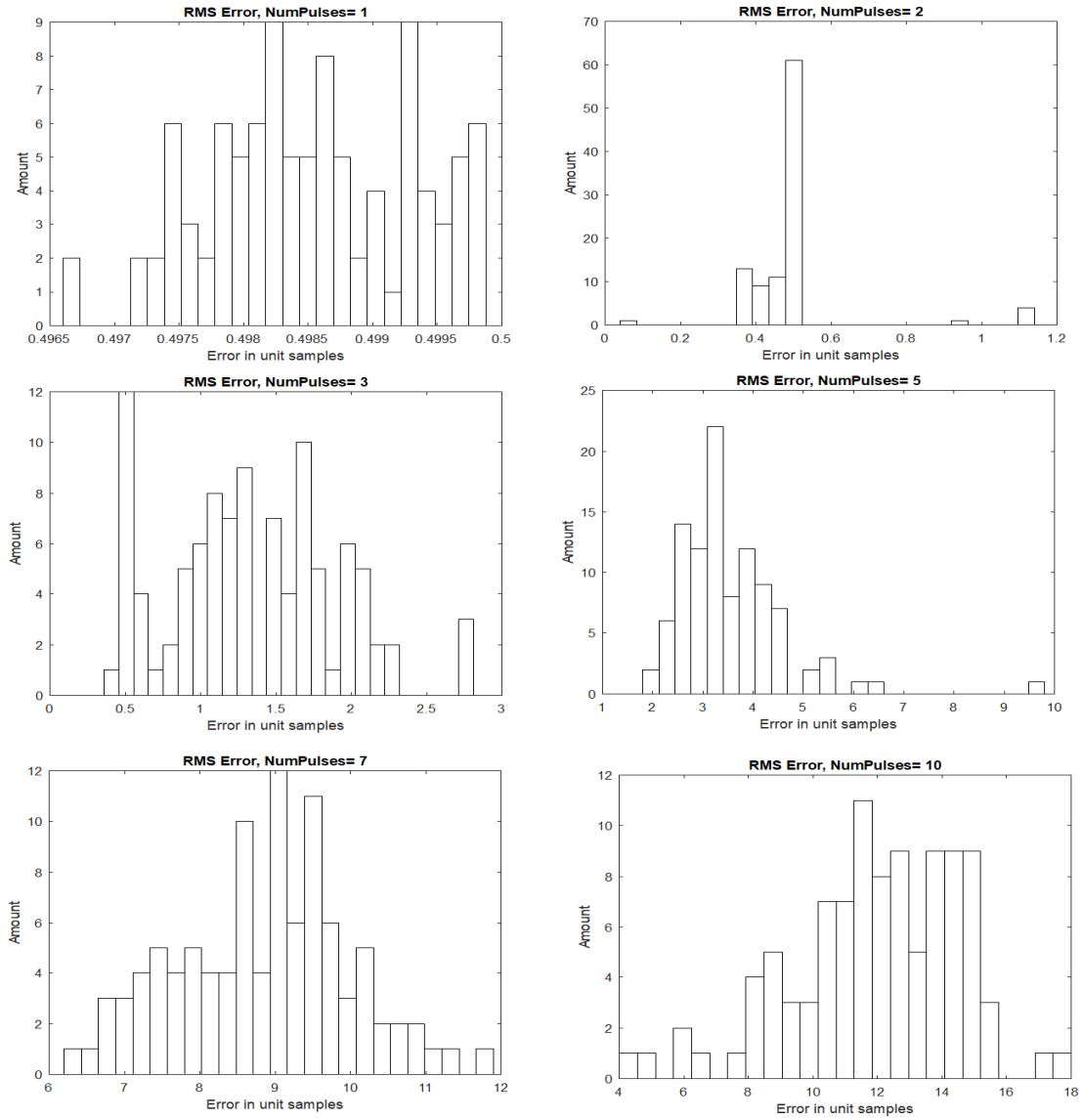

Figure 21: The RMS error distributions for different values of p.

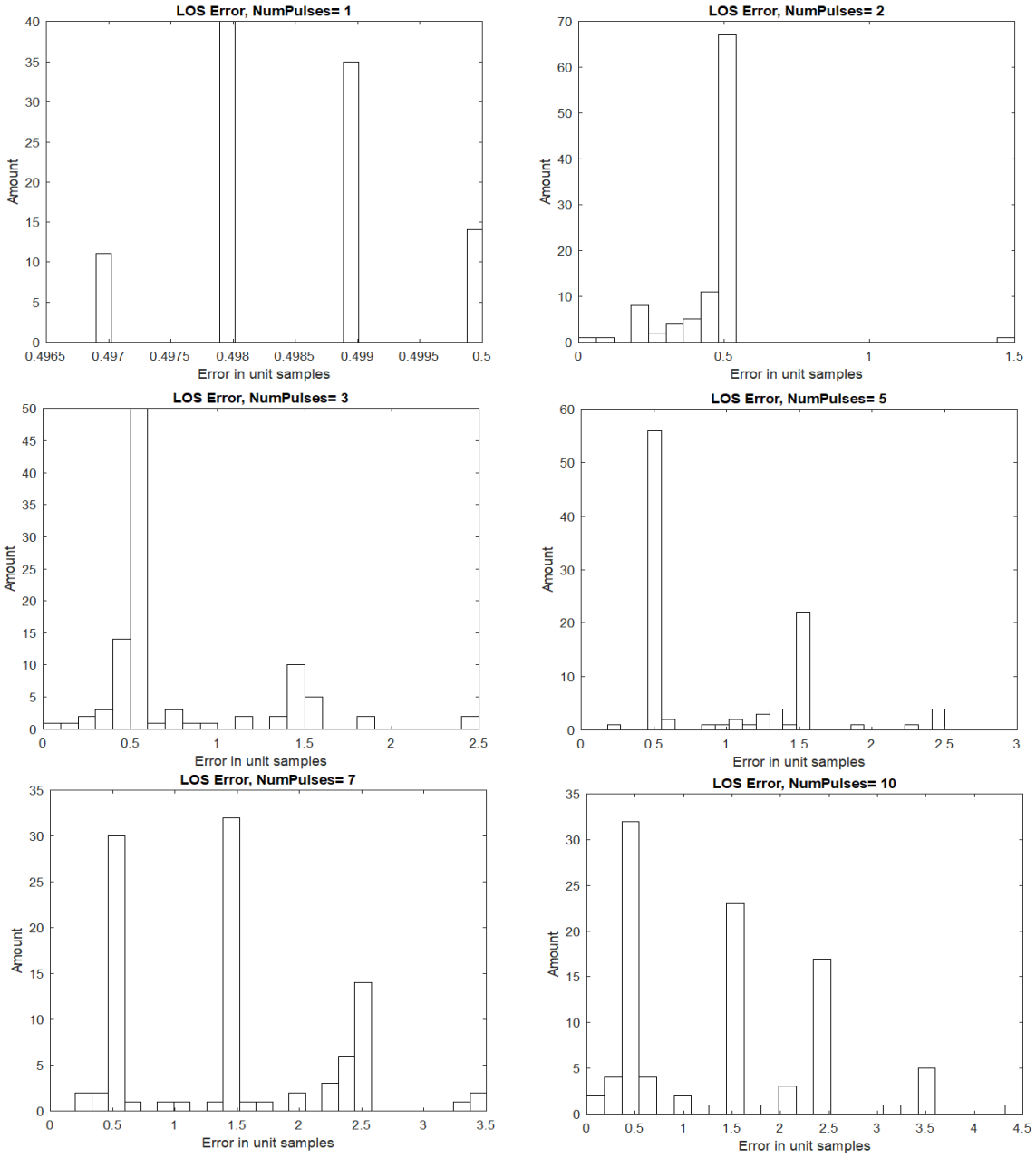

Figure 22: The LOS error distributions for different values of p.

Simulink Results

Since we want to show that the algorithm can be implemented in parallel, we present Simulink results comparing the outputs of the Simulink implementation using 1 repetition of the uncorrelated ATSC PN 511 signal, with zero padding. If the results match, then we can conclude that this is a proper implementation of the algorithm as presented in MATLAB. The results plotted in figure 23 below show the channel responses after the algorithms have been completed using different symbols (that can be overlapped) and with the values of the amplitudes and time-delays labeled.

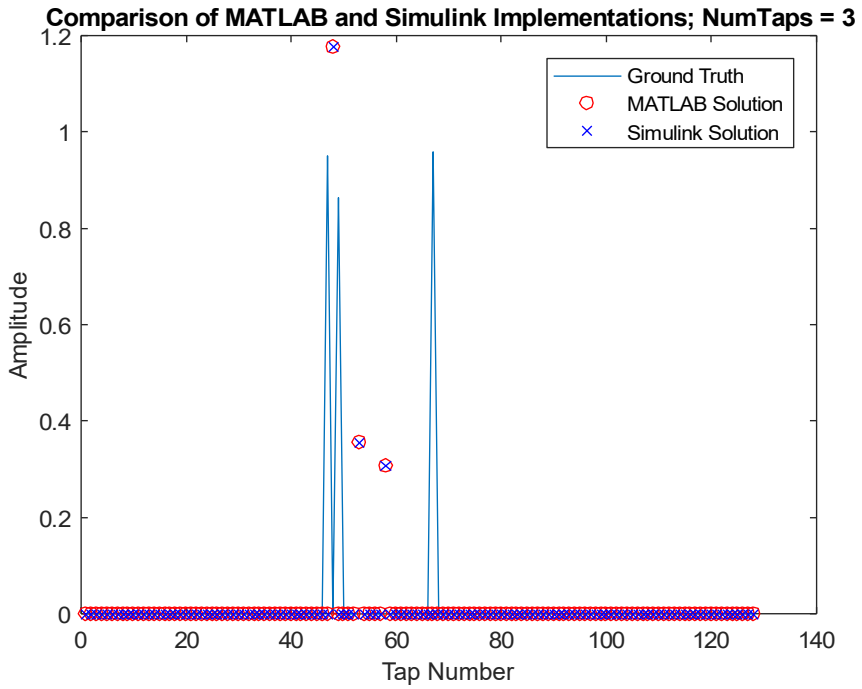

Figure 23: A comparison of the channel response output of the MATLAB and Simulink implementations with a given signal input.

Discussion

Having generated our results, we now analyze them.

Since the pulses are Gaussian, when we try to resolve them, the results are best compared to the values of σ and of the FWHM of the training signal. We see clearly from the plots of the mean square error that compared to the FWHM of the signal, we are relatively accurate.

Note that although the histograms show the root-mean-square grow as p grows, this does not tell the whole story. The RMS error peaks approximately at .5 for p=1, at .5 for p=2, at 1.4 for p=3, at 3.2 for p=5, at 9 for p=7, and at 11.5 for p=10. This seems to indicate that the algorithm becomes less relevant as p increases. However, as mentioned before, the RMS error between the ground truth delay vector and the estimated delay vector essentially matches and compares the two vectors based on their indices instead of the values of the delays. Therefore, an error that involves placing an extra pulse or not including a pulse in an early index of an estimation creates a mismatch between the information in the two vectors and makes the RMS error less meaningful. Furthermore, it becomes less meaningful when we consider that the RMS uses amplitudes that may converge close to zero. Convergence to zero essentially removes them from consideration in the delay update.

In order to get a better assessment of how well the algorithm works with higher p, we look at the plots of the channel vectors. Although this does not provide us with a numerical representation of the accuracy, it gives us a qualitative view of what is happening. By looking at the channel response vectors we plotted, we see that the accuracy is high when p is low. When p is higher, we observe that the accuracy is still somewhat high, but we see that if the ground truth pulses are closely spaced, the algorithm my basically combine their information into one pulse. This results in some extra pulses off away from the last ground truth pulse. This is partially a

result of the initialization. Another thing of note is the amplitudes of the pulses. We see that with the way that our delays are initialized, it leads to a relatively good LOS delay, but many of the amplitudes are incorrect. Some amplitudes are significantly greater than their respective ground truth amplitudes, while some are close to zero. Since the delay updates depend on the amplitudes, when the amplitudes are close to zero, the delays don't update significantly, essentially removing them from consideration when it comes to delay update and allowing some of them to be stuck past the last ground truth pulse. This is especially the case as p increases and many of the initial delays become significantly larger than the highest ground truth delay. These then approach 0 instead of moving into the range of the ground truth delays, leading them to factor into the calculations of the RMS delay without contributing much. By ignoring these pulses, the algorithm essentially acts as though it as fewer pulses.

Another way we can assess the algorithm that is especially useful at higher p is to look at the absolute values of the errors of the LOS. Firstly, it is much easier to map the LOS in ground truth to the LOS in converged solution by looking at the first element. This comparison is also meaningful because the LOS physically corresponds to some information related to the channel, meaning that we are comparing relevant physical information instead of just matching positions in the vectors. By looking at the values of the of the LOS error histograms, we see that the magnitude of the LOS error peaks approximately at .5 for p=1, at .5 for p=2, at .5 for p=3, at .5 for p=5, at 1.5 for p=7, and at .5 for p=10. These are significantly less than the RMS errors at these same p values. This can largely be attributed to the problems with using RMS, but it can also be attributed to the initialization supporting an accurate LOS component. By observing a set of channel vectors, we notice that the LOS tends to be more accurate than many of the later pulses. Although the algorithm is best if it is able to estimate the entirety of the channel vector

with high accuracy, we can assign extra importance to the improvement in accuracy of the LOS even though some of the more delayed pulses may be off because the LOS is special in our application.

It is worth noting that since the selector requires an integer number for the delay to be selected, the delays in our vectors are rounded before being used to select a part of the signal. This means that an error in the delay of .49 is rounded to an error of 0 when selecting, which explains why we see many LOS delays converge to an error of .49 or 1.49 or 2.49.

This algorithm has some advantages and some disadvantages over conventional super-resolution algorithms.

A disadvantage is the dependence on the initialization delays. This can lead to a decrease in robustness that can make it unreliable as the sole ranging method in certain high-risk situations. A trivial way to understand this is that if the algorithm is initialized to the ground truth or to values close to the ground truth, it converges perfectly, but not necessarily when the initialization is based on an estimation. Another disadvantage is that the problem the algorithm solves is still largely canonical. Although this has already been established, it is still worth mentioning that in a real-world situation, we don't always know exactly how many channel response vector pulses there truly are, so some extra work and care needs to be in place to make that estimation.

Despite this algorithm's disadvantages, it has many advantages that can make it very useful. An important advantage is that is can be continuously run in real-time as a stochastic gradient descent algorithm. This is important in the context of ranging and localization because it allows for more efficient tracking of a moving source because we would not have to keep on obtaining large samples and redoing the convergence.

The ability to be run in real-time is very well complemented by its ability to be implemented in parallel. Running it in parallel makes this algorithm able to handle a fast data input. We see the ability to run in parallel in the Simulink implementation. The result of the Simulink simulations comparing the outputs of the same signals run on the MATLAB and the Simulink implementations shows the same outputs, indicating that the two implementations are of the same algorithm. Furthermore, many of the features of the LMS parallel implementation appear in this parallel implementation. This reaffirms the algorithm's potential to be implemented in parallel as an FPGA or an ASIC, mirroring some of the main advantages of the LMS.

In this regards, this algorithm has some trade-offs with the LMS when it comes to hardware. This algorithm uses less memory and has much fewer multiplication operations than the conventional LMS, but has multiplexers.

Ultimately, this algorithm serves as an LMS-based approximation for a super-resolution algorithm, as opposed to a more mathematically rigorous algorithm like compressive sensing. Since it is an approximation, even when the correct number of pulses is chosen, there is no guarantee that absolute convergence to an error of 0 occurs. A more complex mathematical solution is more likely to be able to find an exact solution to this problem, but with reduced complexity comes the ability to more easily understand the inner workings of the algorithm at any particular time and to readjust initializations or parameters as is fit.

The process presented is essentially the backbone of the algorithm one may use. In order to exploit the advantages of this algorithm to the fullest, one would add some extra functions in order to improve robustness or to obtain some more utility. For example, one can make changes such as adding extra processes that detect if an obvious error has been made and then reinitialize it. Simple ways to do this are to see if the delay separations of adjacent delays are going too far

beyond the FWHM or the range resolution in a received signal that cannot be resolved, because this contradicts the inability to be resolved through more conventional methods. Another functionality to add for a practical system may be the ability to run this algorithm while keeping a varying amount of pulses. Implementation would require the systems and the digital designers to be more clever, but it is nevertheless very realistic.

A system that uses this algorithm as its primary means of ranging may also optimize itself to fully exploit this algorithm's advantages. For example, when the delays are close to the ground truth, the algorithm converges more easily. An engineer designing an energy efficient ranging system can then operate the transmitter at a lower duty cycle once the delays converged a first time, allowing for lower energy emissions.

Conclusion

We attempt to design an algorithm with the advantages of the LMS filter but that is able to constrain for a known amount of pulses in the channel response. By applying a similar approximation to that used in the LMS in a way that modifies both delays and amplitudes, we are able to design such an algorithm. We use it to resolve closely spaced Gaussian pulses. The algorithm is able to map the channel response of the system with a super-resolution higher than those that can be obtained using linear methods. Furthermore, the algorithm is very accurate for estimating the delay or position in the channel response vector of the LOS component, which can be very useful in ranging applications.

References

[1] F. Colone, R. Cardinali, P. Lombardo, “Cancellation of clutter and multipath in passive radar using a sequential approach”, *IEEE Conference on Radar*, April 2006.

[2] Z. Farid, R. Nordin, and M. Ismail, “Recent Advances in Wireless Indoor Localization Techniques and System”, *Hindawi Publishing Corporation: Journal of Computer Networks and Communications*, 2013.

[3] A. Jishi, Y. Huang, Y. Wang, “Multipath Mitigation Using Weight Suppression for Equalization in Indoor Positioning Systems”, *IEEE Antennas and Propagation Symposium (AP-S)*, July 2018.

[4] N. Jardak, N. Samama, “Short Multipath Insensitive Code Loop Discriminator”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 278 - 295 , Jan. 2010.

[5] F. Fereidoony, M. Sebt, S. Chamaani, S. Mirtaheri, “Continuous basis compressive time-delay estimation in overlapped echoes”, *IET Signal Processing*, vol. 11, no. 5, pp. 566-571, 2017.

6] F. Fereidoony, M. Sebt, S. Chamaani, S. Mirtaheri, “Model-based super-resolution time-delay estimation with sample rate consideration”, *IET Signal Processing*, vol. 10, no. 4, pp. 376-384, 2016.

[7] E. Candes, J. Romberg, T. Tao, “Stable Signal Recovery from Incomplete and Inaccurate Measurements”, *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, March 1, 2006.

[8] E. Candes, J. Romberg, T. Tao, “Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information”, *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489-509, 2006.

[9] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*. Chichester, West

Sussex, United Kingdom: John Wiley & Sons, Inc., 1998.

[10] O. Oshiga, S. Severi, G. T. F. de Abreu, “Optimized super-resolution ranging over ToA measurements”, *IEEE Wireless Communications and Networking Conference*, 2014.

[11] M. Bhuiyan, E. Lohan, M. Renfors,“A Reduced Search Space Maximum Likelihood Delay Estimator for Mitigating Multipath Effects in Satellite-based Positioning”, *Proceedings of 13th IAIN World Congress*, 27-30 October, 2009.

[12] J. Dhiman, S. Ahmad, K. Gulia, “Comparison between Adaptive filter Algorithms (LMS, NLMS and RLS)”, *International Journal of Science, Engineering and Technology Research (IJSETR),* vol. 2, no. 5, May 2013.

[13] Advanced Television Systems Committee Inc. “ATSC Recommended Practice:Receiver Performance Guidelines”, Internet: https://www.atsc.org/wp-content/uploads/2015/03/Receiver-Performance-Guidelines.pdf, April 7, 2010, [Mar. 12, 2019].

[14] C. Knapp, G. Carter, “The generalized correlation method for estimation of time delay”, *IEEE Transactions on Acoustics Speech & Signal Processing*, pp. 320-327, 1976.

[15] G. Yadav, B. Krishna, M. Kamaraju, “Performance of Wiener Filter and Adaptive Filterfor Noise Cancellation in Real-Time Environment ”, *International Journal of Computer Applications*, vol. 97, no.15, July 2014.

[16] Mathworks. “Equalization” Internet: https://www.mathworks.com/help/comm/ug/equalization.html, [Mar. 12, 2019].

[17] M. Devendra, K. Manjunathachari. “DOA estimation of a system using MUSIC method”, *International Conference on Signal Processing and Communication Engineering Systems*, Jan. 2015.

[18] F. Ge, D. Shen, Y. Peng et al. “Super-resolution time delay estimation in multipath

environments", *IEEE Transactions on Circuits and Systems*, vol. 54, no. 9, pp. 1977-1986, Sept. 2007.

[19] J. Fuchs, "Multipath Time-Delay Detection and Estimation", *IEEE Transactions on Signal Processing*, vol. 47, no. 1, January 1999.

[20] T. Li, and Yinhui Tang, "Frequency estimation based on modulation FFT and MUSIC algorithm", *First International Conference on Pervasive Computing, Signal Processing and Applications*, 2010.

[21] C. Mathews, M. Zoltlwski, "Eigenstructure techniques for 2-D angle estimation with uniform circular arrays", *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2395-2407, 1994.

[22] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[23] T. Cai, L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise", *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, July 2011.

[24] T. Sarlos. "Improved approximation algorithms for large matrices via random projections". *Symposium on Foundations of Computer Science.* 47th Annual IEEE Symposium, pages 143–152, Oct. 2006.

[25] S. Chen, D. Donoho, "Basis pursuit", *Proceedings Asilomar Conference on Signals, Systems and Computers*, 1994.

[26] S. Chen, D. Donoho. M. Saunders, "Atomic Decomposition by Basis Pursuit", *SIAM Review*, vol. 43, no. 1, pp.129-159, 2001.

[27] H. Farrokhi, "TOA estimation using MUSIC super-resolution tech-niques for an indoor audible chirp ranging system", *Proceedings of the IEEE International Conference on Signal*

Processing and Communications, pp. 987-990, 2007.

[28] R. Demirli and J. Saniie, “Model-based estimation of ultrasonic echoes, part I: analysis and algorithms”, *IEEE Transactions on Ultrasonics Ferroelectrics and Frequency Control*, vol. 48, no. 3, pp. 787-802, May 2001.

[29] J. A. Stuller, “Maximum-likelihood estimation of time-varying delay—Part I”, *IEEE Transactions on Acoustics, Speech, and Signal Processing* , vol. 35, no. 3, pp. 300-313, Mar. 1987.

[30] K. Fyhn, S. H. Jensen, M. F. Duarte, “Compressive time delay estimation using interpolation”, *IEEE Global Conference on Signal and Information Processing*, pp. 624-624, Dec. 2013.

[31] International Telecommunications Union, “Radio Regulations”, Internet: http://search.itu.int/history/HistoryDigitalCollectionDocLibrary/1.43.48.en.101.pdf, 2016, [March 14, 2019].