Low Correlation Codes for Sonar Systems

by

Rajiv Pratap

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Engineering
in Electronic and Computer System Engineering.

Victoria University of Wellington
2016

Abstract

Sonar is a vital technology for the detection of objects in the water. Sonar systems have been redefined over many decades, but research is still being conducted into optimal detection methods. Codes, and the filters that process the codes, have been at the forefront of this research. An important objective has been the minimization of interference caused by reflections. Matched filters are commonly used in sonar systems. They are equivalent to correlation filters, which are bound by the Welch bound. The Welch bound governs the minimum peak correlation for points outside of detection.

This thesis investigated matched filters and their bounds, and it was found that by relaxing the condition for detection, properties beyond the Welch bound could be achieved. By relaxing these conditions, the Welch bound no longer applies, and so a modified Welch bound was developed to accurately investigate the nature of these codes. In this thesis, methods to generate codes for these new codes were investigated. Generating codes for a matched filter is a non-convex problem, so gradient based methods were utilised. Methods to improve correlation and power characteristics were developed, along with methods for mapping a sequence for use with a digital transmitter having particular limitations. Mis-matched filters were used to improve signal characteristics that may be lost due to this mapping.

The performance of the generated codes was evaluated, and the relationships between input parameters and output properties of the resulting signal were observed. These performance assessments demonstrate that tradeoffs are required between various properties, and a balance is needed to obtain codes useful for sonar. The optimization was parametrized by an example set of requirements for sonar. The signals were found to meet the

given requirements, and when compared to codes typically used in sonar, the optimized signals were shown to have significantly better correlation properties. Furthermore, compared to the general bounds for the properties of codes, it was found that the new codes had nearly optimal properties, and performed better than equivalent codes bounded by the Welch bound.

The performance of codes were also investigated in a water tank to verify their feasibility. There were several additional considerations which limit codes that can be tested, and once these are taken into account the test provided a robust method to verify the design process. Initial tests showed results that differed from simulations, but after the inclusion of zero padding before upscaling, the results from empirical testing agree with simulation.

Summarizing the research in this thesis, a new set of codes were developed using a gradient based optimization method. The codes were mapped to a digital transmitter, and the filter adjusted using a mis-matched filter. The optimization was shown to generate near optimal codes which met all the given sonar system requirements.

Acknowledgments

Many thanks to my supervisor, Dr Paul Teal, whose supervision and ingenuity guided research throughout the project.

Also thanks to my family and friends, for the continual support throughout my studies and education.

Contents

Chapter 1

Introduction

Underwater sonar has been a vital technology for detection of objects in the surrounding environment. Utilised in nature and in modern technology, sonar is a simple yet accurate scheme that provides a large amount of information regarding nearby obstacles in a short amount of time. Underwater sonar uses acoustic pressure waves [57] pulsed into the water, which are reflected by objects in the water and received back as echoes. By aligning the received pulses with the sent pulse, the time delay between reflections can be resolved with high accuracy and can be used as an indicator for the distance between the transmitter and obstacles.

Sonar detectors were initially developed for military purposes, to assist the US Navy in detecting enemy submarines during WWII [52]. Several classified reports were produced, later declassified, providing an in depth analysis on sonar systems and the signal processing used to maximise detection [10]. This sparked a new field of research into signal processing, and produced innovations in detection systems with broad applications.

Modern day applications of sonar still include military use, but the scope has broadened to include commercial fishing as well as recreational fishing, stimulating research in these new applications [1, 29, 41]. For fishers, it is useful to know if fish are nearby before attempting to catch, as this saves time and resources. Underwater detectors provide a useful aid to

detect fish, providing direction for fishing spots, indicating the depth of the sea as well as potential obstacles which may interfere with fishing [31].

Unlike the military application, which typically involves detection of large objects, fishing requires the detection of signals of various relative strength [32]. Fish typically reflect a weaker signal, compared to reflections off the sea floor [51]. This introduces difficulties in accurate detection, especially if reflections overlap. A major objective of this thesis is to investigate methods to produce ideal codes for the scenarios discussed above. To understand this further, it is useful to first look at the sonar system in detail.

1.1 Sonar systems

Sonar systems vary in design and functionality; however the core concept of these systems remains the same. The typical sonar system is based on an echo, and is illustrated in Figure 1.1. A transmitter pulses a signal into a medium, where it travels through and is reflected by any obstacles along the path. A receiver picks up these reflections as one long signal. Once reflections are identified, the relative delay between transmitting and receiving pulses can be calculated according to the velocity equation in Equation (1.1). Using the time delay, and the known velocity constant in water, the distance d of a reflected pulse can be calculated. This distance is used to identify any potential obstacles, or points of interest such as fish.

$$d = vt \tag{1.1}$$

where v is the velocity of sound in the water,
t is the elapsed time between transmission and reception, and
d is the distance the pulse travelled, both forward and back.

Although the concept of sonar systems is simple, robust detection is difficult. In simple cases each reflection can be distinguished easily as peaks

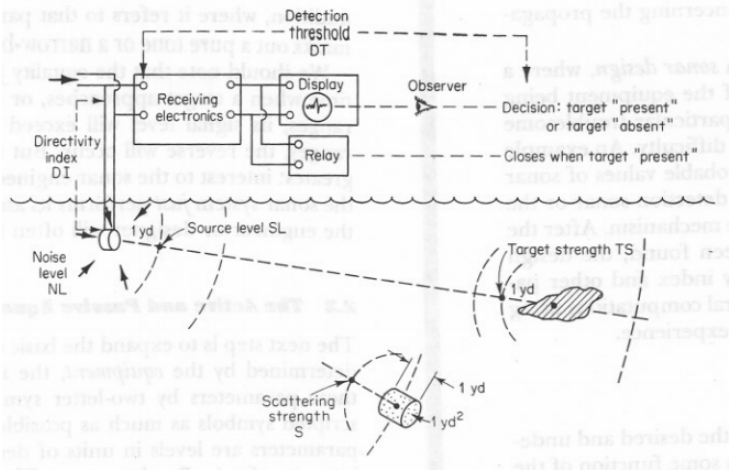


Figure 1.1: Diagram of sonar system, indicating basic components. [10]

in the signal, however this can become difficult when reflections overlap and are distorted. To aid in distinguishing these signals, a filter is typically used to compare the transmitted signal against the received signal. The output of this filter is then used to identify the likelihood of any point in the received signal being a reflection of the transmitted signal. The use of this filter can be very powerful - if designed effectively, received pulses can be identified with high accuracy, identifying delays by as little as 1 sample. However the design of the filter response is challenging, as both the filter coefficients and the transmit pulse must be designed.

One implementation of filter design is the matched filter, introduced in the following section, Section 1.1.1.

1.1.1 Matched Filter

The matched filter is a common filter used in sonar detection [34]. The matched filter is the optimal filter to maximize the signal to noise ratio, or SNR, of a known signal in Gaussian noise. If the noise in underwater sonar is assumed to be Gaussian, then the matched filter is the optimal filter for detecting reflections of a transmitted pulse. Several sources have analyzed

noise for underwater environments [8, 39], and although not truly Gaussian the noise can be modeled as Gaussian as a first approximation.

Details of the matched filter are explored further in Section 2.1. For now, it is important to note that the output to a matched filter is equivalent to the correlation algorithm between a transmitted signal and the received signal, shown in Equation (1.2) below.

$$\begin{aligned} R_{f,g}[k] &= \langle f[n], g^*[n-k] \rangle \\ &= \sum_{-\infty}^{+\infty} f[n] g^*[n-k] \end{aligned} \tag{1.2}$$

For optimal output of the filter, the filter output should be optimized for desired sonar properties such as resolution. Using the matched filter, this is equivalent to optimizing the autocorrelation and cross correlation of transmitted signals. This provides a simple set of inputs to vary, the transmitted codes, which can then be converted to matched filter coefficients. However this optimization of correlation properties are not trivial, as explored in Chapter 3.

Several considerations exist for the design of underwater sonar systems, which guide properties and objectives for optimization in the sonar system. These considerations are explored in the next section, Section 1.1.2.

1.1.2 Underwater Sonar

Underwater sonar adds several complications to sonar design. In underwater sonar, pulses typically reflect off marine life and off the ocean floor. These different sets of objects reflect pulses at different intensities. Reflections off the ocean floor and off solid obstacles are relatively strong in power, while reflections from fish are typically weak.

Additionally, there are typically multiple reflections for fish due to the swim bladder [5], as well as the shape and type of fish [11]. The multiple weak reflections are likely to overlap, and for detection reflections need to

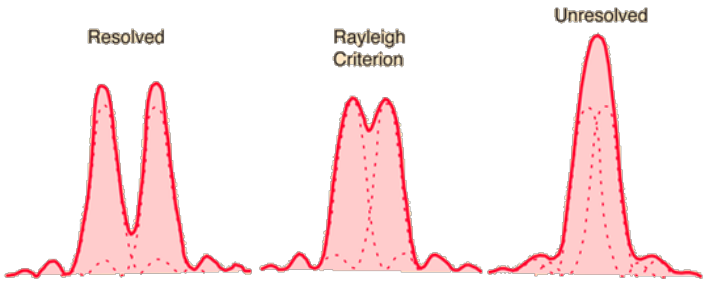


Figure 1.2: Illustration of the Rayleigh Criterion, for resolving signals.

be resolved for separate fish, and ideally grouped for the same fish. This requirement can be described as fish to fish resolution.

For fish in similar locations, reflected signals are expected to be similar in strength. To resolve them, the Rayleigh Criterion must be met. The Rayleigh Criterion was originally conceived for the optical resolution of objects [14], however has been shown to apply to signal resolution in general [7]. An example of this resolution is shown in Figure 1.2. This figure demonstrates that for signals of similar strength to be clearly resolved as separate signals, the points of overlap must not exceed half the strength of the peak of the signal. This is equivalent to the -3 dB point of each signal. To resolve fish with high distance resolution, the reflected signals off each fish must be able to be resolved with minimal distance between peaks. This requirement is equivalent to minimizing the distance of the -3 dB point from the centre peak, for each signal.

Furthermore, any fish near the sea floor need to be detected as well. This involves detection of a strong signal from the sea floor, and a weak signal from the fish. Empirically this has been noted to differ as much as 40 dB in signal strength between each peak [12, 28, 49]. Since the fish in this scenario is located near the sea floor, it is expected that the signals will overlap. This introduces a new requirement for resolution, to be able to resolve a weaker signal while overlapping with a stronger signal. This is not trivial, as these signals differ by as much as 40 dB, requiring overlapping points in the strong signal to be lower than the peak of the weaker signal. In other words

in the strong signal these overlapping points must be below -40 dB from the main peak. It should be noted that this scenario does not require the same distance resolution as for fish to fish resolution, instead has a looser requirement for resolution. This provides more flexibility in the design.

1.2 Proposed Sonar System

Typical sonar systems use a pressure-based transmitter to create pressure waves, and a receiving hydrophone to pick up reflected waves, measuring the time for them to return. Beamforming is used in conjunction to provide spatial information. However these sonar techniques have limitations.

A major limitation of typical sonar systems is the angular range. A single sonar projector can only transmit effectively in a limited range of angles, limiting the angles at which objects can be detected by the sonar system. One way to improve this range is to use multiple transmitters and receivers. However simply repeating projectors across the bottom of the boat would result in overlapping ranges for the projector, necessary to eliminate blind spots. This would result in transmitted pulses interfering with each other. If the pulses interfere, then the beam pattern would be affected — the beam pattern could not be used in beamforming calculations for position or for target strength estimation.

The solution to this problem is to use codes orthogonal for all time delays on each transmitter, so that if transmitted or reflected signals overlap the signals do not significantly interfere with each other, allowing the beam pattern to remain consistent and beamforming to remain accurate. The system is shown in Figure 1.3. This system is set up on the bottom of a boat or fishing vessel, and detects objects below the vessel in a 2D slice. As the boat moves along a series of slices are taken, to eventually form a map of the ocean floor. The design of these codes is non-trivial, as noted in Chapter 2. Additionally the codes are required to meet resolution requirements for underwater sonar environments, as described earlier in Section 1.1.2.

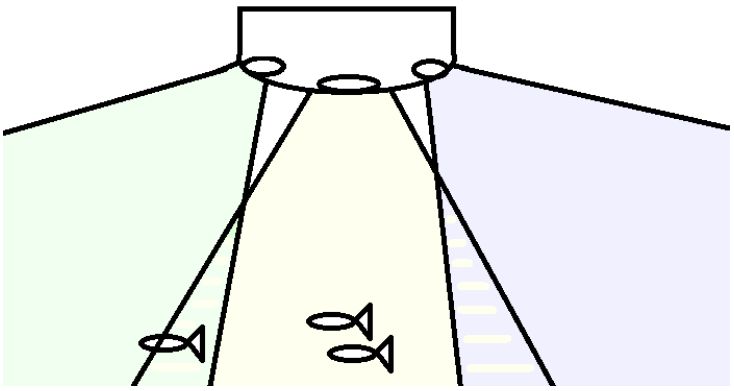


Figure 1.3: Visualisation of multiple transmitter / receiver system (2D slice). Each ellipse represents a transmitter and receiver pair, with the effective angular range shown.

1.3 Additional Considerations

There are various considerations around sonar systems affecting the quality of detection. In particular, noise is a significant factor that should be minimised for optimal detection. Much research has been undertaken to investigate various noise properties, and noise sources in the ocean [42, 55], however in most cases the noise can be assumed to be Gaussian noise [50]. To reduce the effect of this noise, it is useful to look at the receiving filter.

Several filters have been researched for sonar, including the Eckart Filter [9], the matched filter [16], and the mismatched filter [2]. It has been found that for white Gaussian noise (AWGN), the Matched Filter is the most effective filter for maximising signal to noise ratio (SNR). The matched filter is equivalent to a correlation filter that measures the correlation between a template signal, in this case the transmit signal, and a received signal. By using the transmit signal as a template the optimal detection can be made for that transmit signal.

Another consideration is the transmit signal itself. Through empirical experiments it was found that different transmit signals produce different filter responses in the matched filter, which affects the matching of non-

perfect signals such as those distorted by noise or interference. To optimize SNR and minimize interference, a method is needed to find the optimal transmit pulse. Signal design for detection is an area of active research and is explored further in Section 1.4.

1.4 Codes and sequences

Code and sequence design is a relatively abstract field with a broad range of applications in signal processing. In particular it is useful in communication systems and detection systems such as for sonar.

Pulses used in sonar systems are typically optimised to produce ideal responses in receiving equipment. Various codes have been explored such as Gold codes [17], Zadoff-Chu sequences [36], and other poly-phase codes with good aperiodic properties [13]. These codes have good correlation properties with a wide range of applications, including application in sonar. These codes follow a pattern which make them easy to generate, however they are also limited in their properties with high peak errors for aperiodic systems. Optimization techniques have been applied for receiving filters [46] to improve properties, and although these optimized filters show improvement they are still limited in properties for resolution.

Bounds for general codes have been determined in the Welch bound [54], describing bounds on the peak correlation error. The Welch bound shows bounds on what properties can be achieved, however do not show how to achieve them. The Welch bound shown in Equation (1.3) shows the optimal bound on correlation properties, indicating the minimum possible level of the peak error in correlation detection.

$$c_{max} \geq \sqrt{\frac{M-1}{M(2N-1)-1}} \tag{1.3}$$

where

c_{max} is the correlation bound,

M is the number of codes, and
N is the length of each code.

Applied to underwater sonar for fish detection, we can observe the expected performance for fish detection systems. Typical parameters are M=2 and N=[100-1000]. In the Welch bound, this equates to $c_{max} \geq$-26.988 dB for N=100 and $c_{max} \geq$-36.017 dB for N=1000. Referring back to Section 1.1, the expected difference of strength in reflections are -40 dB for fish to bottom detection, and -3 dB for detection between fish. In this case, the requirement to resolve both fish and the sea floor is not met, and so detection cannot be reliably performed.

This set of parameters is limited to codes governed by the Welch bound - codes that ideally have delta like autocorrelation and zero cross correlation. However, it has been empirically observed that some codes can have better than Welch bound performance if the target properties are modified. By targeting a main lobe of detection in autocorrelation instead of a delta function, the correlation properties outside the main lobe have been observed to be significantly lower. This result has stimulated the research into alternative codes that can achieve better than Welch bound performance, and this development is a core result of this thesis.

Having observed the scope of the system, and potential bounds of these systems, it is now useful to see how these requirements convert to pulse and filter design. The next chapter, Chapter 2, details design considerations for the sonar system designed in this thesis, exploring the matched filter system and how sonar properties affect the filter.

Chapter 2

Code Design

Several codes exist for sonar, as noted in Section 1.4, using the matched filter as a detection filter for the sonar system. The properties of the codes and filter are governed by the Welch bound [54], describing a bound on the minimum peak correlation outside the peak autocorrelation. These filters target a delta-like detection, however investigation in sonar properties have suggested an alternative approach may overcome the Welch bound, and have been verified empirically. To aid investigation into this new approach, the Welch bound is investigated and extended for these new properties.

To begin this investigation into the design of codes, it is useful to first look at the detection system, matched filtering.

2.1 Matched Filtering

Introduced in Section 1.1.1, the matched filter is the optimal filter for maximizing SNR in the detection of a known signal, in Gaussian noise. This is particularly useful for sonar, as obstacle information is not typically obtained from the pulse itself, rather from the time delay of reflections, and from beamforming. Detection of reflections involve detection of a known signal, the transmitted pulse, so a filter can be used to ideally detect this reflected pulse with high accuracy, in Gaussian noise.

To understand why the matched filter is optimal, and the benefits it provides towards sonar, it is useful to first derive the matched filter itself.

2.1.1 Derivation of the Matched Filter

The derivation below is one possible derivation for the matched filter, derived by Rabbani using matrix algebra [38]. The process looks at a general filter, then substitutes the filter into the definition of SNR and maximizes to find the optimal filter for SNR.

The derivation begins by observing the general linear filter, as shown in Equation (2.1). This takes an input x and convolves with the filter h to produce an output sequence y. The input is defined by (2.2). Here, the input sequence x is defined as the signal s plus the noise v. This noise can be described by a covariance matrix, as described by (2.3). The general filter defines the filter output as it varies over time, however for detection the filter should have, at some point in the output, a single optimal output maximizing SNR.

$$y[n] = \sum_{k=-\infty}^{\infty} h[n-k]x[k] \tag{2.1}$$

$$x = s + v \tag{2.2}$$

$$R_v = E\{vv^{\mathrm{H}}\} \tag{2.3}$$

The filter output for a single point is defined in (2.4). This shows the output of the filter, y, as the convolution with a general filter h with the input vector x. This output will be shown to be optimal for a particular solution to the filter h. The output y is separated into the signal component and noise component of the output, y_s and y_v respectively.

$$y = \sum_{k=-\infty}^{\infty} h^*[k]x[k] = h^{\mathrm{H}}x = h^{\mathrm{H}}s + h^{\mathrm{H}}v = y_s + y_v \tag{2.4}$$

The SNR is defined as the power of the desired signal over the power of the noise. As noise varies, the expected power of the noise is taken for

analysis. This is shown in (2.5). Substituting the definition of each output component gives (2.6).

$$\text{SNR} = \frac{|y_s|^2}{E\{|y_v|^2\}} \tag{2.5}$$

$$= \frac{|h^{\text{H}}s|^2}{E\{|h^{\text{H}}v|^2\}} \tag{2.6}$$

h is a constant filter, and so the bottom line can be expanded as shown in (2.7). Additionally the covariance matrix is substituted in.

$$E\{|h^{\text{H}}v|^2\} = E\{(h^{\text{H}}v)(h^{\text{H}}v)^{\text{H}}\} = h^{\text{H}}E\{vv^{\text{H}}\}h = h^{\text{H}}R_v h \tag{2.7}$$

Using (2.7), the SNR can now be represented by (2.8). This equation is modified in (2.9) to introduce $R_v^{1/2}R_v^{-1/2}$ in the top line, which is equivalent to muliplying by 1, and in the bottom line the covariance matrix is split into two terms of $R_v^{1/2}$. This step appears counterproductive, however will assist with simplification further in the derivation.

$$\text{SNR} = \frac{|h^{\text{H}}s|^2}{h^{\text{H}}R_v h} \tag{2.8}$$

$$\text{SNR} = \frac{|(R_v^{1/2}h)^{\text{H}}(R_v^{-1/2}s)|^2}{(R_v^{1/2}h)^{\text{H}}(R_v^{1/2}h)} \tag{2.9}$$

The next step uses the Cauchy-Schwarz inequality, as shown in (2.10), where equality is met if a and b are collinear, as indicated in (2.11).

$$|a^{\text{H}}b|^2 \leq (a^{\text{H}}a)(b^{\text{H}}b) \tag{2.10}$$

Equality if

$$a = \alpha b \tag{2.11}$$

where α is a scalar.

Using this inequality to expand the absolute square term, the following result in (2.12) can be obtained. This step also groups terms $R_v^{1/2}h$,

and groups $R_v^{-1/2}s$. This inequality enables the expression to be greatly simplified, to the expression in (2.13).

$$\text{SNR} = \frac{|(R_v^{1/2}h)^{\text{H}}(R_v^{-1/2}s)|^2}{(R_v^{1/2}h)^{\text{H}}(R_v^{1/2}h)} \leq \frac{\left[(R_v^{1/2}h)^{\text{H}}(R_v^{1/2}h)\right]\left[(R_v^{-1/2}s)^{\text{H}}(R_v^{-1/2}s)\right]}{(R_v^{1/2}h)^{\text{H}}(R_v^{1/2}h)}. \tag{2.12}$$

$$\text{SNR} = \frac{|(R_v^{1/2}h)^{\text{H}}(R_v^{-1/2}s)|^2}{(R_v^{1/2}h)^{\text{H}}(R_v^{1/2}h)} \leq s^{\text{H}}R_v^{-1}s \tag{2.13}$$

As stated earlier, equality is met if vectors are collinear. Observing (2.13), the bound is met if $R_v^{1/2}h = \alpha R_v^{-1/2}s$, where α is a scalar. Rearranging for h gives the optimal filter for SNR, in Equation (2.14).

$$h = \alpha R_v^{-1}s \tag{2.14}$$

If the noise is Gaussian, then the covariance matrix R_v is unitary, leaving the expression for the optimal filter to maximize SNR in Gaussian noise. The Gaussian noise matched filter is shown in (2.15).

$$h = \alpha s \tag{2.15}$$

Observing the output of this filter substituted into the linear filter, as shown in (2.16), it can be seen that the matched filter in Gaussian noise is equivalent to the correlation algorithm (2.17).

$$y[n] = \sum_{k=-\infty}^{\infty} s^*[n-k]x[k] \tag{2.16}$$

$$R_{f,g}[n] = \sum_{-\infty}^{+\infty} f[k]g^*[k-n] \tag{2.17}$$

This equivalence provides a powerful tool to observe the expected output of a matched filter, when designing a signal. With the matched filter defined, applications to sonar can be investigated to maximize properties in sonar detection. These are investigated in the following Section 2.1.2.

2.1.2 Application in Sonar

In sonar, the matched filter is used as a likelihood detector for reflected signals. A designed signal is transmitted into the water, where it is reflected by objects and obstacles. These reflections are picked up by the receiver, and passed through the matched filter.

The matched filter returns the correlation based on the relative offset of the matching signal, in this case the transmitted pulse, and the reflected signal composed of several reflected pulses. If a reflected pulse is similar to the transmitted pulse then the filter will output the autocorrelation function, with a peak when the matching signal and the reflected pulse are aligned. This peak allows for high accuracy alignment in time, facilitating the calculation of time delay in transmitting and receiving pulses, which in turn allows for calculation of distance with high resolution.

Simple detection mechanisms use a threshold to detect the output of a filter [24], though these mechanisms can be assisted by prefilters [22]. For optimal detection, the output filter should only output above the threshold for detection, and zero otherwise. In reality, codes have a minimum correlation level outside the peak detection, bounded by the Welch bound [54]. The properties of correlation affect overlapping signals and the strength of signals that can be reliably resolved.

To overcome these bounds, an alternative approach was explored by modifying the definition of detection, to detect signals with varying resolution. Early experiments suggested that improvements could be made to the correlation properties in the matched filter, by targeting a main lobe of detection instead of a delta-like detection. This main lobe showed that lower correlation could be achieved beyond the detection region. In the main lobe, this worsens the level of strength that signals that can be resolved reliably, however improves the level of strength that signals can be resolved outside the main lobe.

To implement this, the filter shape can be designed to fit a modified box+delta shape for autocorrelation, and minimally flat cross correlation

when detecting other codes. This provides a simple target to optimize the code set, and is discussed further in Section 2.2. Before investigating these properties, it is first useful to investigate an alternative to the matched filter used in sonar systems. This filter is discussed in Section 2.1.3.

2.1.3 Mis-Matched Filter

Alternatives to the matched filter have been investigated, in cases where it may not be feasible to optimize correlation properties to a sufficient level. [30] Research as early as 1969 have investigated mis-matched filters, filters initially designed to be matched to a signal and are then modified for better properties [30]. The codes in previous research were short and so could be optimized manually per offset, however more recent research have shown more practical applications and quantized losses [40, 47].

Notably, with a mismatched filter the matching is no longer against the same code, so there is a loss in the central peak. However mis-matched filtering allows for significant side lobe peak reduction, resulting in an overall improvement in performance. The details of implementing the mismatched filter are developed further in Section 3.3.

2.2 Properties

The output of a matched filtered sonar system is equivalent to the correlation of a target signal, in the case of sonar the correlation with the transmitted sonar pulse. The correlation properties are shaped to maximize the likelihood of correct detection of objects and obstacles in the water. Regions of detection are categorized into the main lobe in autocorrelation, a region of detection centered around the peak in autocorrelation, and side lobes, areas outside the main lobe of detection which are considered errors in detection.

Ideal properties for use in this system are described below, discussing

how the properties correspond to correlation properties.

- Fish to Bottom Detection
- Fish to Fish Detection
- Interference Rejection
- Pulse Length
- Power Shaping

As discussed in Section 1.1.2, reflections of a signal from a fish near the ocean floor is about -40 dB less than the reflection from the ocean floor itself. To correctly resolve any overlapping detections in the filter, the side lobes need to be less than -40 dB relative to the main lobe peak. This ensures that if a strong signal from the ocean floor is reflected and an overlapping weaker signal is also reflected from a fish, the weaker signal can still be correctly detected without the possibility of mistaking it for the side lobe in the strong signal. The filter response comparing the strong and weak signals can be seen in Figure 2.1.

In contrast, for fish to fish resolution similar strength signals require overlapping components to be below -3 dB, according to the Rayleigh Criterion discussed in Section 1.1.2. This -3 dB roll off from peak detection is set as close to the peak, to maximize time resolution.

Fish to fish resolution is easily met if the fish to bottom detection is met, as when side lobes are below -40 dB they will be well below -3 dB. A box-like function can be used to distinguish regions for side lobes below -40 dB from a main lobe for detection at 0 dB. However this function does not allow for detection of similar signals within the main lobe. Instead, the function can be modified to shift the main lobe outside a single point of detection down to -3 dB, equivalent to a box function plus a delta function. This box+delta function can be observed in Figure 2.2, where the side lobes are set below -40 dB, the main lobe is set below -3 dB and the central peak

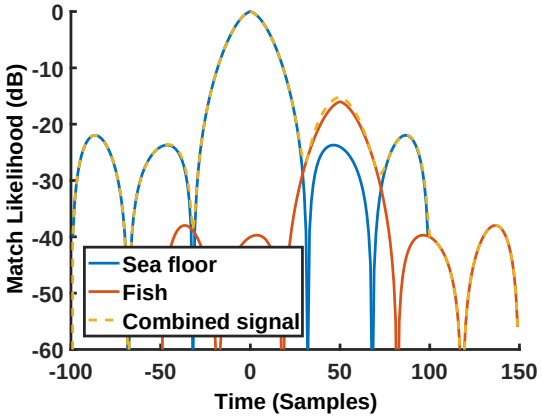


Figure 2.1: Illustrative filter response comparing chirp pulse reflections from fish and from sea floor. Note the larger pulse (blue) and its side lobes, to distinguish overlapping pulses the peak of the weaker pulse (red) must have higher power than side lobes of the larger pulse.

detection is at 0 dB. This allows for signals of significant power difference to be resolved in the side lobe, while signals of similar strength can be resolved in both the side lobes and in the main lobe, providing multiple levels of detection.

In the sonar system, the physical arrangement of projectors are used to increase the angular range of detection. To provide a complete range without gaps, there will be a small overlap between projectors. This will result in codes transmitted from one projector to be potentially received on neighbouring hydrophones. For correct detection, each receiver should only detect the one code and reject codes transmitted from neighbouring projectors. In the receiving filter, this requirement is interference rejection, requiring minimum cross correlation between pulses. The filter should be designed to equally minimize the output when receiving codes from other projectors.

Sonar systems detect for a range of depths and resolutions, and so have different intervals where a pulse can be transmitted before the first reflection needs to be detected. For short distances, the time between the

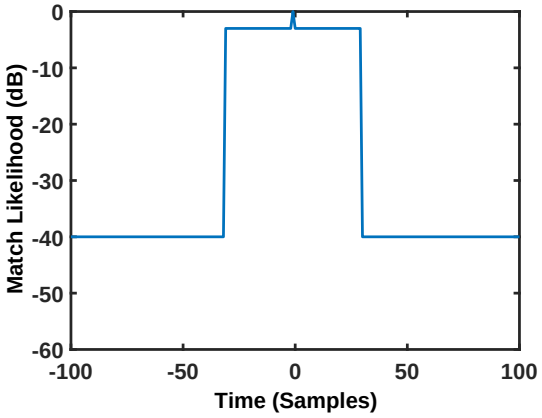


Figure 2.2: Box+delta function to define ideal detection of a pulse. Multiple levels of detection allows for variable resolution between strong pulses and for between strong and weak pulses.

beginning of a transmit pulse and the time when the first reflection is detected is short, so a short transmit pulse is required. For longer distances, a longer time window is available before a reflection is detected so the transmit pulse can be longer. If the sampling frequency is kept constant, then a longer signal allows for a larger number of samples in the code.

In the Welch bound, the bound on the maximum correlation error, it can be observed earlier from Equation (1.3) that larger values of code length N allow for a lower maximum correlation error. For sonar, a longer pulse length allows for better correlation properties, so in the matched filter allows for lower side lobes. This suggests that codes should be designed for the longest code length possible for a given range, to allow for better side lobe properties. This is in contrast to the previous timing requirement, limiting code length based on the available time window, and so a balance must be met between optimal codes and requirements for sonar systems.

Sonar transmitters ideally have perfect recreation of the designed signal in the transmit pulse for detection, however for sonar these systems usually require high power transmitters to achieve long range detection. This imposes limitations on the signal that can be transmitted, based on the

power system supplying the transmitter. Power systems are typically designed to provide a constant supply of power [33], so systems typically struggle to respond to high variations in power that needs to be supplied. In these cases, if there is too high a variation in the power drawn, the signal is distorted since power cannot be supplied. This distortion is especially present at the beginning of a signal, when there is a sharp variation from zero power supplied to the start of the signal. To minimize distortion, the beginning of a transmitted signal should ramp up slowly in power. This reduces variation in power, so the power system can supply enough current, and can be achieved by shaping the magnitude of the designed code.

Another limitation in power is transformer ringing, or ferroresonance [15]. This ringing is common in high power systems with magnetic and capacitive components, caused by energized components such as transformers and capacitors oscillating energy between each other. This oscillation occurs even when the switch to the input circuit is open, and so cannot be solved by simply shutting off the input. To mitigate this phenomenon, the magnitude of the code can be ramped down at the end to reduce trailing ringing after the pulse has ended. This minimizes the residual power oscillating in the transformer, reducing the power of the ringing, and minimizing potential interference outside the transmitting interval.

These properties guide the design of codes, to design for correlation as well as for the code itself. Beyond these properties, there is an additional consideration for the sonar system, concerning the transmitter. A novel digital transmitter was designed and developed outside of the work presented in this thesis, however has several constraints on signals that can be transmitted. This is investigated in the following Section 2.3.

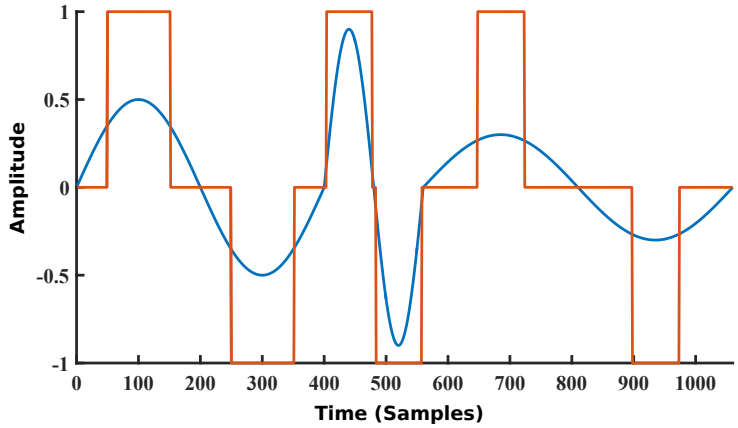


Figure 2.3: Example transmitting cycles in the digital transmitter, of how amplitude is changed by changing pulse width, and how phase is changed by changing each cycles frequency.

2.3 Digital Transmitter

Transitional transmitter systems take complex samples and transmit at the passband frequency, as amplitude and phase offsets. However these transmitter schemes are not ideal, as they are power limited, and require a large amount of data to be stored. An alternative transmitter was proposed to overcome these limitations, a 'digital transmitter'. This digital transmitter transmits whole cycles and modulates the frequency of each cycle, to attain phase properties when sampled.

This can be seen in Figure 2.3. Phase is controlled by setting the length of each cycle, stored as a coefficient of half the length labeled half-pulse width or HPW. The amplitude is controlled by setting the pulse width of positive and negative pulses, each centered at the peak and trough of an equivalent sine wave, and stored as a pulse width coefficient AMP. When filtered, the red digital transmitted signal smooths to a signal similar to the blue signal of sinusoids.

The digital transmitter was proposed and developed before the work presented in this thesis had begun, and is a required component in the

designed sonar system. The design of this transmitter has been developed, however codes to drive this transmitter had yet to be designed. A significant limitation of this digital transmitter is the non-regular cycles generated for the signal. The sampling operation is regular, and so can be determined relative to the transmit rate, however the cycles generated are non-regular, each cycle's position in time is determined by the cumulative lengths of previous cycles. Additionally, cycles are not guaranteed to overlap with a sample point, or may span several sample points. The cumulation of these problems make it difficult to design a signal using linear methods.

Another property of signals generated by this digital transmitter is the amplitude of the signal. The signal is a square-like function, which when filtered for the first harmonic produces a sinusoidal wave. The amplitude coefficient in the digital transmitter is not the same as the amplitude of a sinusoidal wave, however is bound by a function. This function can be found by taking the Fourier series of the waveform, and observing the first harmonic. Observing the digitally transmitted function as (2.18):

$$f(t) = \begin{cases} +1 & \left(1 - \frac{AMP}{AMP_{max}}\right)\frac{T}{4} \leq t \leq \left(1 + \frac{AMP}{AMP_{max}}\right)\frac{T}{4} \\ -1 & \left(1 - \frac{AMP}{AMP_{max}}\right)\frac{T}{4} + \frac{3T}{2} \leq t \leq \left(1 + \frac{AMP}{AMP_{max}}\right)\frac{T}{4} + \frac{3T}{2} \\ 0 & else \end{cases} \quad (2.18)$$

For Fourier analysis, it is easier to observe the interval from 0 to $T/4$, then repeat for the whole waveform, as it is symmetric along the sine wave. Since the waveform is sinusoidal, there are no DC components nor cosine

components in the series. The computation simplifies to the following.

$$\begin{aligned}
b_{n=1} &= \frac{-2}{T}\int_0^T x(t)sin\left(\frac{2\pi t}{T}\right)dt \\
&= 4\times\frac{-2}{T}\int_{1-\frac{AMP}{AMP_{max}}}^{\frac{T}{4}} sin\left(\frac{2\pi t}{T}\right)dt \\
&= 4\times\frac{-2}{T}\frac{2\pi}{T}cos\left(\frac{2\pi t}{T}\right)\Big|_{1-\frac{AMP}{AMP_{max}}}^{\frac{T}{4}} \\
&= \frac{16\pi}{T}\left(cos(\frac{\pi}{2}) - cos(\frac{\pi}{2}\left(1-\frac{AMP}{AMP_{max}}\right))\right) \\
&= \frac{16\pi}{T}\left(-cos\left(\frac{\pi}{2}-\frac{\pi}{2}\frac{AMP}{AMP_{max}}\right)\right) \\
&= \frac{16\pi}{T}sin\left(\frac{\pi}{2}\frac{AMP}{AMP_{max}}\right)
\end{aligned}$$

Scaling factors can be ignored, as the signal should be scaled to maximum power for maximum range. Observing the mapping relationship, the relation simplifies to (2.19), providing a simple equation for the control coefficient and the effective amplitude of the filtered signal. This function is monotonic increasing and one-to-one [44], providing useful properties in mapping.

$$\text{Amplitude} = sin\left(\frac{\pi}{2}\frac{AMP}{AMP_{max}}\right) \tag{2.19}$$

However, like with phase, the non-regular cycles of the digital transmitter limit the usefulness of the amplitude property, specifically in cases when a single cycle spans multiple samples. When this occurs, a single cycle will control the amplitude of multiple samples when downsampled, which limits samples which can be mapped accurately.

A solution proposed in this thesis is to use an alternative method to transmit signals, using a method similar to minimum shift keying, or MSK modulation [20]. This alternative is referred to hereafter as the MSK digital transmitter. The MSK digital transmitter generates cycles at the carrier frequency, varies the amplitude per cycle then circularly phase shifts each cycle depending on the related sample. This can be seen in the example

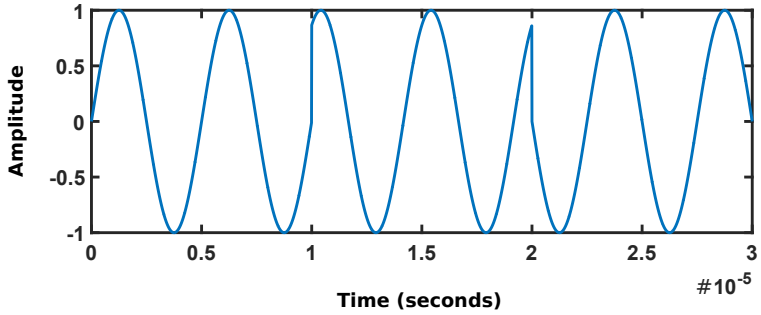


Figure 2.4: Example of MSK modulation. Note the discontinuity at points t=1 $\times$ 10^{-5} seconds and t=2 $\times$ 10^{-5} seconds.

in Figure 2.4. By circularly phase shifting the sample, the overall cycle length remains the same. This simplifies the mapping of a sequence to the transmitter, as the lengths of samples are now regularly spaced. However a major limitation of the MSK digital transmitter is the discontinuities between samples. When cycles are generated at the centre frequency, the waveform is continuous, however phase shifts cause the cycles to no longer be continuous at the borders of samples.

The MSK digital transmitter is an alternative option for a transmitter, in the case that the original digital transmitter is not accurate enough. Note that this alternative is not ideal in signal properties, due to discontinuities between samples, however is offered as a viable alternative. For the remainder of the thesis, the original digital transmitter is used for design, to target compatibility with existing resources.

With the characteristics of the digitally transmitted signal defined, it is now possible to produce signals for the system that accurately represent designed codes. This is developed further in Section 3.2.

Before discussing methods to generate codes, it is first useful to observe theoretical performance of codes. This introduces another method to evaluate codes, to check for optimal performance. This theoretical performance is explored in the next section, Section 2.4, deriving bounds for expected correlation properties in codes.

2.4 Welch Bound (and beyond)

Typical matched filtered systems rely on the properties of autocorrelation and cross correlation to produce an effective likelihood detector. For these systems, ideal detection occurs when the matched signal is perfectly aligned to the reflected pulse, and would otherwise reject the pulse. Here, the ideal code characteristics correspond to a delta like autocorrelation and zero cross correlation. In reality correlation properties have a minimum level of correlation outside the centre of autocorrelation, contributing towards errors in matched filter detection. These correlation properties are bound by the Welch bound, which indicate the optimal bound to minimize peak correlation error.

2.4.1 Welch Bound

In 1974, Welch developed bounds on correlation describing the limits of a matched filter [54]. He noted that although several researchers had looked at autocorrelation, there was little that included cross correlation of signals. The aim of the correspondence was to establish bounds on how small cross correlation and autocorrelation values could be.

Welch's derivation of the bounds begins with a set of vectors describing the code set as $\{(a_1^v ... a_L^v) : v = 1, ..., M\}$, where there are M codes each of length L. The components of a_1^v are complex numbers, and the norm of each code is 1, as indicated below.

$$c_{v,v}(0) = \sum_{i=1}^{L} |a_i^v|^2 = 1$$

The first theorem starts with an inner product of M vectors each of length L and norm 1. For now, relative offsets between vectors are ignored, and are considered later.

Define correlation as

$$c_{v,\lambda} = \sum_{i=1}^{L} a_i^v \bar{a}_i^\lambda$$

and

$$c_{max} = \max_{v \neq \lambda} |c_{v,\lambda}|$$

Welch's first theorem (inner products) is,

$$c_{max} \geq \frac{1}{M-1}\left(\frac{M}{L} - 1\right)$$

$Proof : (Welch1)$

The proof begins with the expected bound if all error is set to worst case, c_{max}.

The equation in (2.20) compares the worst case sum of correlation with the actual sum of correlation across all combinations of vectors in the set. The sum across each point in the cross correlation algorithm is taken, for cross correlation between each and every vector, including the same vector. The number of cross correlation pairs is $M \times M$. Since each vector has a norm of 1, if it matches with itself then the autocorrelation will have a value of 1. There are M vectors so this matched correlation occurs M times, summing to $1 \times M = M$. The vectors will not match in combinations $M^2 - M$ or $M(M-1)$, and each contribute a value c_{max}^2. The sum of correlation for non-matching vectors becomes $M(M-1) \times c_{max}^2$, describing the left side of the inequality. The right side shows the squared norm of correlation, summed across all combinations of vectors in the set.

$$M(M-1)c_{max}^2 + M \geq \sum_{v,\lambda} |c_{v,\lambda}|^2 = B \tag{2.20}$$

Here it is useful to look to another derivation, which proves an equivalent result using a simpler method.

2.4.2 Capocelli

At this point, it is easier to understand the Welch bound for complex hermitian space using the derivation in Capocelli [3]. Here the correlation is represented as an inner product, and simplifies B as an inequality.

$$c_{i,j} = \langle x^{(j)}, x^{(i)} \rangle$$

$$B = \sum_{v,\lambda} |c_{v,\lambda}|^2 = \sum_{v,\lambda} |\langle x^v, x^\lambda \rangle|^2$$

Before deriving the bound on the sum of correlations, a few lemma are introduced to aid in derivation.

Lemma 1: (Row-Column Equivalence) Let $y^{(1)}, y^{(2)}, \dots y^{(L)}$ show the rows of the $M \times L$ matrix representing M codes each of length L, where rows follow the form $x^{(1)}, x^{(2)}, \dots x^{(M)}$, then

$$\sum_{i=1}^{M} \sum_{j=1}^{M} |\langle x^{(i)}, x^{(j)} \rangle|^2 = \sum_{k=1}^{L} \sum_{l=1}^{L} |\langle y^{(k)}, y^{(l)} \rangle|^2 \tag{2.21}$$

Proof: (Lemma 1) Using $\langle x^{(i)}, x^{(j)} \rangle = \langle x^{(j)}, x^{(i)} \rangle^*$,

$$\sum_{i=1}^{M} \sum_{j=1}^{M} |\langle x^{(i)}, x^{(j)} \rangle|^2 = \sum_{i=1}^{M} \sum_{j=1}^{M} \langle x^{(j)}, x^{(i)} \rangle \langle x^{(i)}, x^{(j)} \rangle$$

Expand the inner product,

$$= \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{k=1}^{L} x_k^{(j)} x_k^{(i)*} \sum_{l=1}^{L} x_l^{(i)} x_l^{(j)*}$$

Shift sum over L out, then group (i) and (j)

$$= \sum_{k=1}^{L} \sum_{l=1}^{L} \sum_{i=1}^{M} x_l^{(i)} x_k^{(i)*} \sum_{j=1}^{M} x_k^{(j)} x_l^{(j)*}$$

The terms including summation over (i) and (j) are equivalent to the inner product between columns, and so can be simplified

$$= \sum_{k=1}^{L} \sum_{l=1}^{L} \langle y^{(l)}, y^{(k)} \rangle \langle y^{(k)}, y^{(l)} \rangle$$

Finally using the complex inner product as earlier,

$$= \sum_{k=1}^{L} \sum_{l=1}^{L} |\langle y^{(l)}, y^{(k)} \rangle|^2$$

The next lemma will also prove useful:

Lemma 2: If $a_1, a_2, \ldots, a_L$ are real numbers, then

$$\sum_{k=1}^{L} (a_k)^2 \geq \frac{1}{L} (\sum_{k=1}^{L} a_k)^2 \tag{2.22}$$

with equality if and only if $a_1 = a_2 = \ldots = a_L$.

Proof: (Lemma 2) Consider a random variable X that has a value of a_k with probability $1/L$ for $1 \leq k \leq L$. Observing the square function of this variable, we can see that it is strictly convex over the range, and so using Jensen's inequality [21, 25],

$$E[X^2] = \sum_{k=1}^{L} \frac{1}{L} (a_k)^2 \geq E[X]^2 = (\sum_{K=1}^{L} \frac{1}{L} a_k)^2$$

Multiplying both sides by L gives the lemma.

Using these two lemmas the bound on B can now be proven. Capocelli produces a result describing the sum of correlations as a generalised bound, and this is one typical application of the Welch bound [53]. However, Welch proves this bound and more by further deriving a bound on correlation for matched filters. The next few steps will prove Capocelli's result, which will then be used to derive Welch's bound.

Welch's Bound (Generalized): For sequences of the same energy, i,e,

$$||x^{(i)}||^2 = \langle x^{(i)}, x^{(i)} \rangle = L$$

for $i = 1, \ldots, M$, the following bound applies.

$$\sum_{i=1}^{M} \sum_{j=1}^{M} |\langle x^{(j)}, x^{(i)} \rangle|^2 \geq M^2 L \tag{2.23}$$

Proof: (Welch Bound (Generalized))

The proof begins using lemma 1, using row-column equivalence for the inner products. This shows the sum correlation of combinations of row vectors, in this case the sum of correlation between codes, is equivalent to the sum of the correlation between columns.

$$\sum_{i=1}^{M} \sum_{j=1}^{M} |\langle x^{(i)}, x^{(j)} \rangle|^2 = \sum_{k=1}^{L} \sum_{l=1}^{L} |\langle y^{(k)}, y^{(l)} \rangle|^2$$

Here the correlation between all column vectors is absolute squared, and summed over every combination. This effectively sums the power of correlation for every combination of vectors.

$$\sum_{k=1}^{L} \sum_{l=1}^{L} |\langle y^{(l)}, y^{(k)} \rangle|^2 \geq \sum_{k=1}^{L} |\langle y^{(k)}, y^{(k)}|^2 = \sum_{k=1}^{L} (||y^{(k)}||^2)^2$$

The sum over these combinations is expected to be less than or equal to the sum of the inner products of the same vectors, as the right hand sum is contained in the left hand sum. Equality is reached if the column vectors are orthogonal. The right side of the inequality is simplified as a norm operator. Using lemma 2 in (2.22),

$$\sum_{k=1}^{L} (||y^{(k)}||^2)^2 \geq \frac{1}{L} \left(\sum_{k=1}^{L} ||y^{(k)}||^2 \right)^2$$

The term $||y^{(k)}||^2$ is the squared norm of a single column in the $M \times L$ matrix of codes, and is equivalent to taking the squared magnitude of each sample and summing them together along dimension M. This allows for the following substitution,

$$\frac{1}{L}\left(\sum_{k=1}^{L}||y^{(k)}||^2\right)^2 = \frac{1}{L}\left(\sum_{k=1}^{L}\sum_{i=1}^{M}|x_k^{(i)}|^2\right)^2$$

Rearranging the sum for M and L,

$$= \frac{1}{L}(\sum_{i=1}^{M}\sum_{k=1}^{L}|x_k^{(i)}|^2)^2$$

Using the definition of the 2-norm as the sum of absolute squared elements, to replace the sum across L,

$$= \frac{1}{L}(\sum_{i=1}^{M}||x^{(i)}||^2)^2$$

Capocelli's derivation uses the assumption that each sample as a unit magnitude, and so the squared norm of each code sequence is L. Taking the sum of L across M codes gives the following result,

$$= \frac{1}{L}(ML)^2 = M^2L$$

From which shows Capocelli's result,

$$\sum_{i=1}^{M}\sum_{j=1}^{M}|\langle x^{(j)}, x^{(i)}\rangle|^2 \geq M^2L$$

Modifying the assumption of the norm of the sequence, we can obtain an equivalent result to Welch.

$$\frac{1}{L}\left(\sum_{k=1}^{L}||y^{(k)}||^2\right)^2 = \frac{1}{L}\left(\sum_{i=1}^{M}||x^{(i)}||^2\right)^2$$

The norm of each sequence is now 1, so $||x^{(i)}||^2 = 1$. Taking the sum across M, then squaring the result yields,

$$= \frac{1}{L}(M)^2 = \frac{M^2}{L}$$

Finally, this derivation gives the result

$$\sum_{i=1}^{M}\sum_{j=1}^{M} |\langle x^{(j)}, x^{(i)}\rangle|^2 \geq \frac{M^2}{L} \tag{2.24}$$

2.4.3 Back to Welch

With a core inequality proven, the inequality in (2.24) can now be substituted into (2.25). This provides the following inequality,

$$M(M-1)c_{max}^2 + M \geq \frac{M^2}{L} \tag{2.25}$$

This result is no longer dependent on varying sequences, only on parameters M, L and c_{max}. This greatly simplifies the inequality, and can be rearranged for the following result in (2.26).

$$c_{max} \geq \frac{1}{M-1}\left(\frac{M}{L} - 1\right) \tag{2.26}$$

This is the result Welch presents in his correspondence as the bounds on correlation for codes, without shifts. He extends his proof, postulating that for non shifted codes, there are M codes to check for correlation. For periodically shifted codes, the set of codes increases from M to ML, to account for every periodically shifted version of each code.

For periodic correlation, the correlation is defined as:

$$c_{v,\lambda}(\tau) = \sum_{i=1}^{L-\tau} a_i^v \bar{a}_{i+\tau}^{\lambda} + \sum_{i=1}^{\tau} a_{i+L-\tau}^v \bar{a}_i^{\lambda}$$

$$c_1 = \max_{v\neq\lambda} \max_{0\leq\tau\leq L} |c_{v,\lambda}(\tau)|$$

$$c_2 = \max_{v} \max_{1 \leq \tau \leq L} |c_{v,v}(\tau)|$$

and

$$c_{max} = \max(c_1, c_2)$$

then from earlier in (2.25), comparison of correlation between ML vectors,

$$ML(ML-1)c_{max}^2 + ML \geq \frac{(ML)^2}{L} \tag{2.27}$$

Divide both sides by ML

$$(ML-1)c_{max}^2 + 1 \geq \frac{ML}{L} \tag{2.28}$$

Simplify right hand fraction, and rearrange

$$c_{max} \geq \sqrt{\frac{M-1}{ML-1}} \tag{2.29}$$

This shows the bound for periodic correlation, including relative shifts between vectors.

For aperiodic sequences, the length L is replaced with the length $(2N-1)$, to allow for partial overlap between sequences.

$$c_{max} \geq \sqrt{\frac{M-1}{M(2N-1)-1}} \tag{2.30}$$

2.4.4 Discussion of bounds

For sonar systems, pinging is typically done with pulses separated by silence. The silence between pings means the waveform is non-periodic over the time interval of comparison. When reflections are processed in the receiving matched filter, the correlation algorithm compares these non-periodic waveforms using aperiodic correlation. For sonar systems, the aperiodic Welch bound (2.30) is applicable.

To understand how the bounds affect requirements, it is useful to see where the bounds sit for some example parameters. For a set of 2 codes, of

N	10	100	1000	10000
c_{max} (dB)	-15.6820	-25.9879	-36.0173	-46.0203

Table 2.1: Example parameters for Welch Bound

lengths [10,100,1000, 10000], the following bounds apply in Table 2.1. Here it can be seen that to meet the bounds for fish to sea floor detection, for side lobes to be below -40 dB, the code length must be greater than 1000 samples. However for some requirements the code length may need to be less than 1000 samples, in which case resolution requirements cannot be met even with an ideal code.

However this limitation applies to codes bound by the Welch bound. An alternative approach is to loosen the bound in a region around the aligned detection, to allow for improved side lobe characteristics. This alternative approach is not bound by the Welch bound, as there is not a single point of detection, instead there are multiple regions for detection and for bounds. A modified bound is to be used to determine what code characteristics are achievable.

2.4.5 Beyond Welch

The Welch bound shows a limit in the error of a matched filter, for the gain of non-zero lags in autocorrelation and all lags in cross correlation. A looser definition of detection can provide improvements in these errors, reducing the error for non-centre autocorrelation. This can be achieved by widening the detection point at non-zero lag in autocorrelation to a small range of values centred around zero. This is referred to as the main lobe.

This main lobe can be incorporated into the bound by modifying Equation (2.27). The left side of this equation represents the sum of worst case square correlation, for ML vectors in autocorrelation for codes with zero relative lag, and $ML(ML-1)$ vectors in non-zero relative lag and in cross correlation of codes. However with a range of detection, instead of just

zero lag between offsets, there is a range between $\pm D$ samples around zero with a different worst case correlation of up to $c_{max,D}$. For values in autocorrelation outside the $\pm D$ range and for all offsets in cross correlation, these values can be defined as the error in correlation c_{max}. This results in the following inequality (2.31). This inequality is simplified further in (2.32), by removing ML from both sides and simplifying L on the right side.

$$ML(ML-1-2D)c_{max}^2 + ML(2D)c_{max,D}^2 + ML \geq \frac{(ML)^2}{L} \tag{2.31}$$

$$ML - (1+2D)c_{max}^2 + 2Dc_{max,D}^2 + 1 \geq M \tag{2.32}$$

Rearranging for the side lobe correlation c_{max}, the modified Welch bound takes the form of Equation (2.33). Allowing for aperiodic correlation, the bound takes the form of (2.34). Note as the detection range D goes to zero, the bound takes the form of the original Welch bound. In this modified bound, the improvements in correlation are not immediately obvious. The properties are better illustrated as a 3-dimensional map.

$$c_{max} \geq \sqrt{\frac{M-(1+2Dc_{max,D}^2)}{ML-(1+2D)}} \tag{2.33}$$

Or for aperiodic correlation,

$$c_{max} \geq \sqrt{\frac{M-(1+2Dc_{max,D}^2)}{M(2N-1)-(1+2D)}} \tag{2.34}$$

Figure 2.5 displays the values of c_{max} for various input parameters. Note the white space is negative infinity dB. Here it can be seen that if the main lobe is set to a high value, and the lobe size is non-zero, then the correlation properties can usually be better than if the lobe size is zero.

One case where this does not hold true is if the main lobe worst case correlation is set below the correlation of side lobes if the main lobe is zero. For example, in this case if the main lobe is zero the correlation is roughly -15.5 dB. If the main lobe is set below this value, then points outside the main

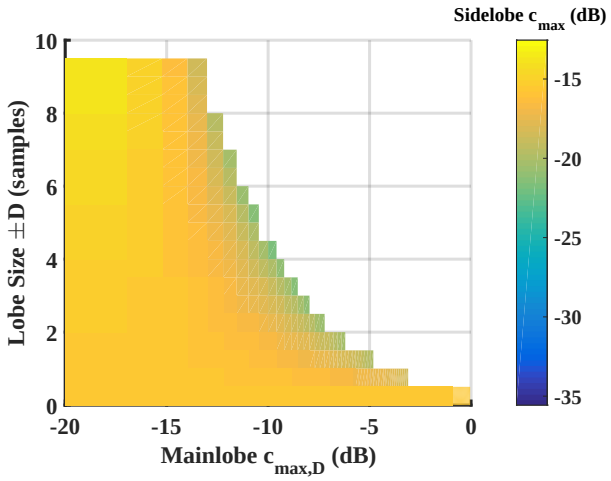


Figure 2.5: Demonstration of various input parameters for the modified Welch bound, for a sequence of length N=10. Note the empty space in the top right corresponds to negative infinity - these points are not bounded so can theoretically reach zero correlation in side lobes.

lobe compensate for the main lobe being below the standard Welch bound, increasing to above -15 dB. This highlights a limitation of the modified bound, correlation in the main lobe must be above the standard Welch bound, to show improvements outside the main lobe.

Another particularly noteworthy characteristic is that for a large enough lobe size, and if the values are allowed to be high enough, then values outside the main lobe can reach negative infinity decibels, or in non-dB scale have a value of zero. This suggests that given a loose enough definition of detection, a pair of codes can be designed to not have any error in the matched filter. In practise, the lobe size is typically constrained to less than 10% of the sequence length, in this figure would be less than 1 sample, and so is still bound. However the relation suggests interesting properties that may be useful if interference properties are more heavily weighted than detection properties in the matched filter.

These modified bounds suggest that better code characteristics can be

achieved for sonar detection, and provide motivation for Chapter 3 to produce codes approaching these bound. The next chapter details methods to produce these codes.

Chapter 3

Generating Codes

Several codes already exist in the domain of sonar detection, and have been shown to have good performance, as well as simplicity. In particular, polyphase codes as investigated by Wirth [58] have been shown to display good properties in side lobes, and can be further improved by modifying the filter. However the nature of these modifications are only for a specific class of codes, and are not guaranteed to have good cross correlation properties as needed for a multiple projector system.

A new set of parameters and a novel approach to detection motivate a new set of codes to be designed and generated, to achieve even greater performance.

3.1 Matched Filter code generation

The matched filter produces an output based on the template signal to be matched, and the received signal to be compared. The filter function is equivalent to the correlation function, as shown in Equation (3.1). This equation is dependant on two variables, a and b correlated together, and so for the problem of optimizing both a and b for correlation properties the problem is not a convex problem. This makes the problem difficult to solve directly using existing convex optimization toolboxes.

$$R[n] = \sum_{i=-\infty}^{+\infty} a[i]b^*[i+n] \tag{3.1}$$

Since the problem is non-convex, a different approach must be taken to find a solution. Iterative methods were observed to improve correlation properties of two codes, suggesting that improvement can be made. Iterative methods are not as efficient as convex optimization, nor as optimal, as iterative methods tend to settle towards local minima instead of global minimum solutions. However, iterative methods do not rely on a problem being strictly convex, instead requires a function be continuous and ideally locally convex around a local minima. This allows non-convex functions to be optimized using these methods [48], and provides a usable option for optimization of correlation.

The method investigated for this problem was the gradient method. This iterative method simply requires an error function that is first order differentiable, and to settle on a solution the error function should have local minima.

3.1.1 Gradient Method

The gradient method, in particular the Method of Steepest Descent [35], is an algorithm to iteratively step toward an optimal minimum solution in a differentiable function, as shown in Equation (3.2).

$$x^{(k)} = x^{(k-1)} - \mu \nabla F(x), \quad k = 1, 2, ... \tag{3.2}$$

The objective function describes the error for optimal properties, so that minimizing the error optimizes the properties. The derivative of this error is calculated, and the code is shifted in the opposite direction of the error by a small fraction μ of this gradient. This method depends on the function being first order differentiable, and locally convex.

This method can be applied to the design of sonar codes, by designing error functions based on the correlation properties of the sequence.

The equations for correlation are known, and so an analytical first order derivative can be calculated for the optimization functions, and efficiently implemented to optimize for the properties. The algorithm can be extended to optimize for several metrics at once, by calculating multiple gradients, applying a weighting function to each and including them all in the iteration process. This can be seen in Equation (3.3).

$$x^{(k)} = x^{(k-1)} - \mu(\lambda_1 \nabla F_1 + \lambda_2 \nabla F_2 + ...), \quad k = 1, 2, ... \tag{3.3}$$

The algorithm is applied to the peak error of each of the error functions, such that the points of maximum error are pushed down. In the matched filter, signal detection involves a detection based on the maximum of this filter, and the aim for the sonar system is to reduce the number of incorrect decisions or detections. This reduction in error is achieved by optimizing for the worst case scenario, by minimization of points of maximal error. By iteratively finding points of maximum error and minimizing each point, the peak error can be reduced to a sufficient target value in relatively few steps. This reduces the worst case error for the function, and so reduces the properties of worst case error.

Applying the gradient descent algorithm to the properties outlined in Chapter 2 yields the optimization problem in Equation (3.4). In the matched filter the properties for detection and interference rejection are based on the correlation properties of the transmitted code, so the equations for correlation are incorporated into the objective function as the correlation function $R(l; x, y)$. The power of the code is also an objective, and so the scaled magnitude of each sequence is also present in the function as $|Ta|$ and $|Tb|$.

$$\underset{a,b}{\text{minimize}} \qquad \max_{l} \left\{ \begin{array}{c} \lambda_{c_{1,2}} |R(l, H_a a, H_b b)|, \\ \lambda_{c_{1,1}} |R(l, H_a a, H_a a) S(l)|, \\ \lambda_{c_{2,2}} |R(l, H_b b, H_b b) S(l)|, \\ \lambda_{\text{power}} |Ta|, \\ \lambda_{\text{power}} |Tb|, \end{array} \right\} \tag{3.4}$$

where:
$R(l; x, y)$ is the cross correlation of lag l between x and y,
$S(l)$ is the scaling function for autocorrelation,
and T is real valued scaling function for the time sequence.

To implement the optimization process, a loop is set up to continually optimize a code vector describing the code set. The first step in the loop is to calculate the point of maximum error. The gradient of error for this point is calculated, as a partial derivative relative to the code vector. The code vector is then iterated in the negative direction of the error gradient, to reduce the error. This new code vector is now fed back to the beginning of the loop, to find the new point of maximal error and repeat the process.

The described process still requires a method to calculate the error for each property, as well as the gradient of each error. These are detailed in the following subsections, where derivations of error gradients are defined for each property to optimize. These gradients are implemented in the gradient method, to iteratively optimize a set of codes.

3.1.2 Error functions — Autocorrelation

The derivative of the error function for autocorrelation properties is based off the matched filter detecting a received pulse. The matched filter compares an incoming signal with the transmit pulse, with an output signifying the likelihood of match. This comparison is done using the cross correlation algorithm, and for ideal detection the received pulse should be identical

to the transmit pulse. The detection characteristics are simplified to the autocorrelation algorithm, as shown in Equation (3.5).

$$R[n] = \sum_{i=-\infty}^{+\infty} a[i]a^*[i+n] \tag{3.5}$$

Ideal detection in the matched filter is typically a delta function, where detection at zero lag is maximal and all other points are erroneous and minimised. The error points are traditionally equally penalised, however with the multiple properties for detection there are multiple regions to detect with different error levels. This is handled by having two regions to optimize for autocorrelation, each weighted differently depending on the target lobe level. A separate penalty is applied to each region, and a function defined to evaluate this penalty. The first function takes the maximum error across the main lobe and minimizes to optimize fish to fish detection, while the second function minimizes the maximum error across the side lobes to optimize fish to bottom detection.

These two functions differ only on the region which the maximum error can be found, and on the weightings that each function apply in the objective function. Once the point of maximum error is found, the point is optimized according to the error gradient to minimize this error. This error gradient for autocorrelation is derived below.

For the centre autocorrelation, of 0 offset, the normalised correlation value should be equal to 1. The error can be analysed for each position of the code, as elaborated below in the following error derivation, beginning with Equation (3.6). The error shown below is derived for code a; but is equivalently applicable to code b.

$$e_0 = a_1a_1^* + a_2a_2^* + ... + a_Na_N^* - 1, \tag{3.6}$$

where N is the length of the code. To simplify the expression, we use a variable k to represent the correlation calculation for a particular offset, at

this point for offset 0. Note that the autocorrelation for zero offset is real valued.

So

$$e_0 = k_0 - 1$$

where

$$k_0 = a_1 a_1^* + a_2 a_2^* + ... + a_N a_N^*$$

and so,

$$|e_o|^2 = (k_0 - 1)^2$$

The equation for $|e_0|^2$ represents the error at the 0 offset in the code, so the partial derivative is taken in terms of a and b to find the rate of change for each point in the code. There are no b terms in the autocorrelation of a, so the partial derivative in terms of b is simply zero. Continuing the derivation for a:

$$\frac{\partial |e_0|^2}{\partial a_1^*} = 2(k_0 - 1)\frac{\partial (k_0 - 1)}{\partial (a_1^*)}$$

The partial derivative of $(k - 1)$ is simply a single term, so the equation simplifies to:

$$\frac{\partial |e_0|^2}{\partial a_1^*} = 2(k_0 - 1)a_1$$

The zero offset autocorrelation gradient can be generalised for each point in the code.

$$\frac{\partial |e_0|^2}{\partial a_i^*} = 2e_0 a_i \tag{3.7a}$$

$$\frac{\partial |e_0|^2}{\partial b_i^*} = 0 \tag{3.7b}$$

For non-centre autocorrelation, of offsets not equal to 0, the correlation at that point should ideally be 0, so non-zero values are treated as error. As an example, the error at offset $j = -1$ will be derived first, then generalised.

$$e_{-1} = a_1 a_2^* + a_2 a_3^* + ... + a_{N-1} a_N^*,$$

where N is the length of the code. Taking the absolute squared error as the error function:

$$|e_{-1}|^2 = e_{-1}e_{-1}^*$$

This cannot simply be squared like with autocorrelation, as the value is not guaranteed to be real. The derivative of the squared magnitude is

$$\frac{\partial|e_{-1}|^2}{\partial a_1^*} = e_{-1}\frac{\partial e_{-1}^*}{\partial a_1^*} + e_{-1}^*\frac{\partial e_{-1}}{\partial a_1^*}$$

e_{-1} does not contain any a_1^* terms, so $\frac{\partial k_{-1}}{\partial a_1^*} = 0$. However the k_{-1}^* term contains a single a_1^* term, with a single variable in front. This simplifies the expression to:

$$\frac{\partial|e_{-1}|^2}{\partial a_1^*} = e_{-1}a_2$$

Similarly

$$\frac{\partial|e_{-1}|^2}{\partial a_2^*} = e_{-1}a_3 + e_{-1}^*a_1$$
$$\frac{\partial|e_{-1}|^2}{\partial a_3^*} = e_{-1}a_4 + e_{-1}^*a_2$$

By repeating for several a_i, it can be observed that the partial gradient term relates to the relative shift j. This can be seen in (3.8), and the autocorrelation for non-zero offsets are generalised as:

$$\frac{\partial|e_j|^2}{\partial a_i^*} = e_j a_{i-j} + e_j^* a_{i+j} \tag{3.8}$$

where a_n is treated as 0 if the indices $i-j$ or $i+j$ go outside the index range, i.e. beyond $[1, ..., N]$.

Once regions and corresponding penalties are taken into account, the subset of the error function becomes (3.9):

$$\frac{\partial|e_j|^2}{\partial a_i^*} = \begin{cases} \lambda_{\text{Main}}\left(e_j a_{i-j} + e_j^* a_{i+j}\right) & \text{if } |j| \leq \text{lobe size}, \\ \lambda_{\text{Side}}\left(e_j a_{i-j} + e_j^* a_{i+j}\right) & \text{else} \end{cases} \tag{3.9}$$

3.1.3 Error functions — Cross Correlation

The error function for cross correlation is based on the matched filter receiving a pulse in one projector that originated from another projector. Neighbouring projectors use different codes, and so to minimize error the filter output should be minimized, which for different codes corresponds to minimizing the cross correlation between codes. Unlike with the autocorrelation function, the cross correlation is equally minimized for all lags as all points are equally erroneous.

An additional consideration is the multiple angles of overlap. Repeating multiple projectors increases angular range, but to prevent blind spots the ranges of the projectors must overlap. The overlap occurs on both sides of the projector, and the matched filter output is not symmetric due to the transducer not being symmetric across angles. This is discussed further in Section 3.1.4. Because of this asymmetry the codes must be optimized for cross correlation through multiple transducer functions. The error gradient for cross correlation is derived below, and is applied to the maximum cross correlation across codes through all transducer functions applicable.

Cross-correlation is similar to shifted autocorrelation in error gradient, but is consistent for all offsets. As an example, the error at offset $j = -1$ will be derived first, then generalised for all offsets.

$$e_{-1} = a_1 b_2^* + a_2 b_3^* + ... + a_{N-1} b_N^*,$$

where N is the length of each code.

$$|e_{-1}|^2 = e_{-1} e_{-1}^*$$

Firstly looking at the partial derivative with respect to a_1^*, using the product rule for derivatives:

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = e_{-1} \frac{\partial e_{-1}^*}{\partial a_1^*} + e_{-1}^* \frac{\partial e_{-1}}{\partial a_1^*}$$

e_{-1} contains no a_1^* terms, and so $\frac{\partial e_{-1}}{\partial a_1^*} = 0$. However the e_{-1}^* term contains a single a_1^* term, multiplied with a single term b_2. This greatly simplifies the expression to:

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = e_{-1} b_2$$

Similarly

$$\frac{\partial |e_{-1}|^2}{\partial a_2^*} = e_{-1} b_3$$
$$\frac{\partial |e_{-1}|^2}{\partial a_i^*} = e_{-1} b_{i+1}$$

By repeating for several a_i, it can be observed that the partial gradient term relates to the relative shift j. Similar steps can be used to derive for the partial derivative in terms of b. This is generalised as:

$$\frac{\partial |e_j|^2}{\partial a_i^*} = e_j b_{i-j} \tag{3.10a}$$
$$\frac{\partial |e_j|^2}{\partial b_i^*} = e_j^* a_{i+j} \tag{3.10b}$$

where a and b are treated as 0 if the indices $i - j$ or $i + j$ go outside the index range of a, beyond [1,...,N].

3.1.4 Transducer Characteristics

A transducer is a process or device that converts energy from one form to another [56]. In sonar, these are incorporated into projectors to convert an electrical signal into a pressure wave, to transmit in the water for sonar detection. The transmitted pressure wave is reflected back by objects, and picked up by a receiving hydrophone, a specific type of transducer, to convert the pressure wave back into an electrical signal for processing.

For this reason transducers are necessary in a sonar system, however they have limitations. Most significantly, transducers do not perfectly transfer a signal; certain frequencies pass with less loss than others. Careful design of transducers can reduce this effect, however there will still be a non-trivial amount of signal distortion. This affects the magnitude and phase of the resulting signal, deviating from the properties of the designed sonar code.

To achieve the properties needed for accurate sonar, these transducer characteristics must be taken into account. These characteristics can be accurately measured, and so provide an opportunity to incorporate these in the sonar code. The transducer characteristics can be described by a frequency response, which varies depending on the transmit angle, and the receive angle. An example of these characteristics can be observed in Figure 3.1, showing the frequency response at various angles for a particular transducer. For a transmitter receiver pair, the signals are expected to be mostly along the main beam axis, or 0 angle, affected by the characteristics shown in Figure 3.2. For any cross talk, the angle for the transfer functions are expected to be at the edges.

The characteristics of the transducer can now be incorporated into the optimization of sonar codes themselves, to produce an optimal set of codes that when passed through a transducer will produce properties as per the design. This addition into the optimization requires several adjustments, which are detailed in the next section.

Derivation and implementation

The transducer characteristics can be represented by convolving the impulse response of the transmit and receiver transducers with each code. This convolution with an impulse response can be represented with the convolution matrix, in Equation (3.11). Here, $\mathbf{H}$ is an $(N + M - 1) \times N$ matrix and a is a column vector. N is the length of the code before passing through the transducer, and M is the length of the impulse response of the transducer.

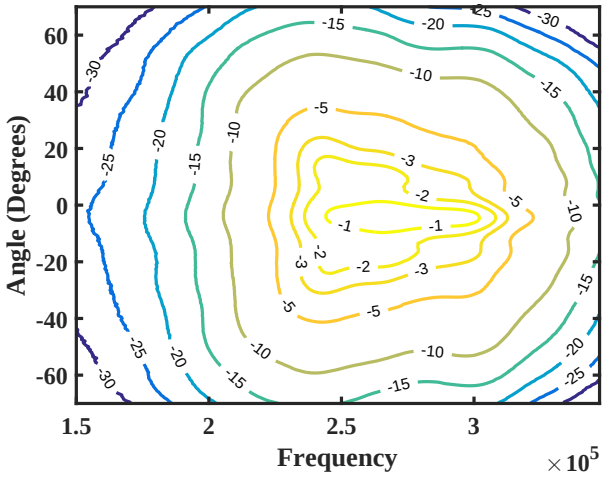


Figure 3.1: Transducer characteristics of a tested transmitter, affecting the gain of frequencies as they pass through. Note the contour plot shows the amplitude normalized by the peak gain, in dB.

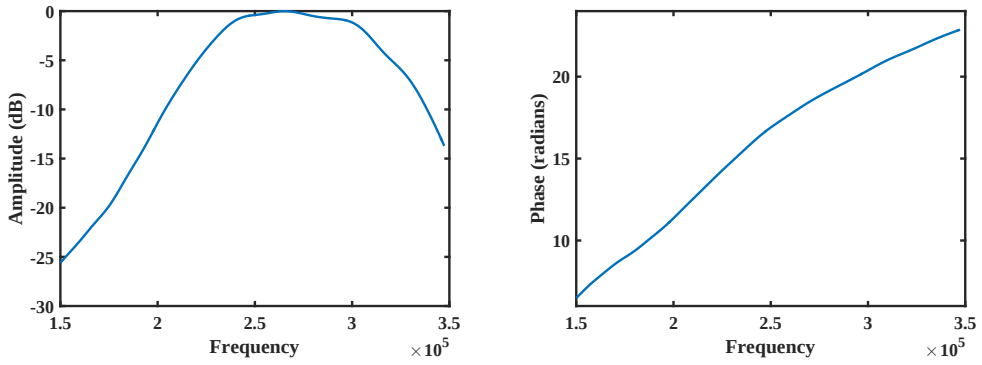


Figure 3.2: Transducer characteristics of a tested transmitter, taking a slice along the main beam axis ($\theta = 0$). Left shows the normalized magnitude in dB against frequency, right shows the phase delay in radians.

$$a_t = h * a = \mathbf{H}a \tag{3.11}$$

$$\mathbf{H} = \begin{bmatrix} h_M & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ h_{M-1} & h_M & 0 & 0 & 0 & \cdots & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & h_1 & h_2 & \dots & h_{M-1} & h_M \\ 0 & \dots & 0 & 0 & h_1 & \dots & h_{M-2} & h_{M-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \dots & \dots & \dots & 0 & \cdots & h_1 & h_2 \\ 0 & \dots & \dots & \dots & 0 & \cdots & 0 & h_1 \end{bmatrix}$$

Extending this to the derivation of the cross-correlation, we can use the chain rule to simplify the derivation. The derivative of a_t with respect to a is simply the $\mathbf{H}$ matrix itself.

Cross-correlation

For cross-correlation, $k_{t,-1}$ can be defined as:

$$e_{-1} = a_{t,1}b^*_{t,2} + a_{t,2}b^*_{t,3} + ... + a_{t,N-1}b^*_{t,N} = k_{t,-1},$$

where N is the length of each code. $k_{t,-1}$ can otherwise be represented as

$$k_{t,-1} = \mathbf{a_t^T}.\mathbf{b^*_{t,(-1)}},$$

where $\mathbf{b^*_{t,(-1)}}$ is offset by the relative shift between a_t and b_t in the cross correlation algorithm, in this case by -1. $\mathbf{b^*_{t,(-1)}}$ is zero padded such that the length of $\mathbf{b^*_{t,(-1)}}$ remains $N + M - 1$. In this case several zero samples are appended to the end.

The magnitude squared error is given by:

$$|e_{-1}|^2 = k_{t,-1}k^*_{t,-1}$$

Firstly looking at the partial derivative with respect to $a^*_{t,1}$:

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = k_{t,-1}\frac{\partial k^*_{t,-1}}{\partial a_1^*} + k^*_{t,-1}\frac{\partial k_{t,-1}}{\partial a_1^*}$$

Using the $\mathbf{H}$ matrix from Equation (3.11), as well as the alternative representation of $k_{t,-1}$, we can derive the partial derivative of $k^*_{t,-1}$ with respect to a_1^*. Since $\mathbf{b_{t,(-1)}}$ contains no a_1^* terms, it can be treated as constant with respect to a, so the next part is to find the derivative of a_t with respect to a. Observing (3.11), each row of the $\mathbf{H}$ matrix affects a different value in a_t, and each column corresponds to a value in a affecting the a_t vector. To find the derivative of a_t with respect to a, that is to find how a_t changes with a small change in each value in a, we can simply look at each column for each partial derivative of a_n. The derivation simplifies to:

$$\frac{\partial a_t^*}{\partial a_1^*} = \mathbf{H}(:,1)^*$$

$$\frac{\partial k^*_{t,-1}}{\partial a_1^*} = (\mathbf{H}(:,1)^*)^\top.\mathbf{b}^*_{t,(-1)}$$

$$\frac{\partial k^*_{t,-1}}{\partial a_1^*} = (\mathbf{H}(:,1))^H\mathbf{b}^*_{t,(-1)}$$

where $\mathbf{b_{t,(-1)}}$ is a $N+M-1$ by 1 vector.

Extended to the error derivative:

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = k_{t,-1}(\mathbf{H}(:,1))^H\mathbf{b}_{t,(-1)}$$

By repeating for several a_i, it can be observed that the partial gradient term relates to the index i, as shown below.

$$\frac{\partial |e_{-1}|^2}{\partial a_2^*} = k_{t,-1}(\mathbf{H}(:,2))^H\mathbf{b}_{t,(-1)}$$

$$\frac{\partial |e_{-1}|^2}{\partial a_i^*} = k_{t,-1}(\mathbf{H}(:,i))^H\mathbf{b}_{t,(-1)}$$

A similar approach is taken for the relative shift between signals a_t and b_t, the relative shift represented by j. This can be generalised as:

$$\frac{\partial |e_j|^2}{\partial a_i^*} = e_j (\mathbf{H}(:,i))^H \mathbf{b}_{t,(j)} \tag{3.12a}$$

$$\frac{\partial |e_j|^2}{\partial b_i^*} = e_j^* (\mathbf{H}(:,i))^H \mathbf{a}_{t,(-j)} \tag{3.12b}$$

where a and b are treated as 0 if the indices go outside the range of $[1, ..., N]$.

Autocorrelation

A similar derivation applies for autocorrelation:

For non-zero shifts,

$$e_j = a_t * a_{t,(j)}$$

$$e_{-1} = a_{t,1} a_{t,2}^* + a_{t,2} a_{t,3}^* + ... + a_{t,N-1} a_{t,N}^*$$

$$e_{-1} = k_{-1}$$

$$|e_{-1}|^2 = k_{-1} k_{-1}^*$$

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = k_{-1} \frac{\partial k_{-1}^*}{\partial a_1^*} + k_{-1}^* \frac{\partial k_{-1}}{\partial a_1^*}$$

Now,

$$\frac{\partial k_{-1}^*}{\partial a_1^*} = \mathbf{H}^*(1,:) \mathbf{a}_{t,(-1)}^*$$

and

$$k_{-1}^* \frac{\partial k_{-1}}{\partial a_1^*} = \mathbf{H}(1,:) \mathbf{a}_{t,(-1)}$$

so

$$\frac{\partial |e_{-1}|^2}{\partial a_1^*} = k_{-1} \mathbf{H}^*(1,:) \mathbf{a}_{t,(-1)}^* + k_{-1}^* \mathbf{H}(1,:) \mathbf{a}_{t,(-1)}$$

$$\frac{\partial |e_{-1}|^2}{\partial a_2^*} = k_{-1} \mathbf{H}^*(2,:) \mathbf{a}_{t,(-1)}^* + k_{-1}^* \mathbf{H}(2,:) \mathbf{a}_{t,(-1)}$$

This can be generalized for any shift j as:

$$\frac{\partial |e_j|^2}{\partial a_1^*} = k_j \mathbf{H}^*(i,:)\mathbf{a}^*_{t,(j)} + k_j^* \mathbf{H}(i,:)\mathbf{a}_{t,(j)}$$

Similar for e_0, replacing k with $k-1$, gives

$$\frac{\partial |e_0|^2}{\partial a_i^*} = 2(k_0 - 1)\mathbf{H}^*(i,:)\mathbf{a}^*_{t,(j)} \tag{3.13a}$$

$$\frac{\partial |e_j|^2}{\partial a_i^*} = e_j \mathbf{H}^*(i,:)\mathbf{a}^*_{t,(j)} + e_j^* \mathbf{H}(i,:)\mathbf{a}_{t,(j)} \tag{3.13b}$$

3.1.5 Error functions — Power

As discussed earlier in Section 2.2, the sonar system is limited in the power of codes that can be transmitted, due to limitations in the power supply and transformer. To avoid distortion, the code should be ramped up at the beginning to limit power draw, and ramped down to minimize transformer ringing after transmitting. The exact shape to optimize the pulse is not known, however at this stage the pulse is shaped using a piecewise sinusoidal ramp up then ramp down, as a first approximation.

It is useful to develop methods to shape the signal pulse using a general function, so future codes can be developed with a different shape by simply changing an input parameter. Algorithms to optimize the shape of these pulses are detailed in the following sections.

Power ramping using Peak-to-Average Power Ratio

The peak-to-average power ratio (PAPR) is a ratio that measures the peak deviation of the magnitude of a sample from the average value of all samples in a sequence or signal [43]. This can be observed in the equation

below.

$$\begin{aligned} \text{PAPR} &= \left(\frac{\text{P}_{peak}}{\text{P}_{avg}} \right) \\ &= \max_{1 \leq i \leq \text{len}(x)} \left(\frac{\text{P}_i}{\text{P}_{avg}} \right) \end{aligned}$$

where,

$$\begin{aligned} \text{P}_{peak} &= \max_{1 \leq i \leq \text{len}(x)} (x_i^* x_i) \\ \text{P}_i &= x_i^* x_i \\ \text{P}_{avg} &= \mathbf{x}^H \mathbf{x} \end{aligned}$$

and $\mathbf{x}$ is the sequence or signal to measure.

PAPR is useful as it provides a metric for the worst case error in power of a signal. However this assumes that every point in the signal is targeted to have the average power. For the case of pulse shaping, some modifications are needed. One simple modification is to take the difference of the power ratio relative to a fixed scaling function, then take the peak error from this. This scaling power ratio can be seen in Figure 3.14.

$$e_{i,\text{p2a}} = \frac{x_i^* x_i}{\mathbf{x}^H \mathbf{x}} - g_i, \tag{3.14}$$

where g_i is the ramp function.

One limitation with this function is that this error function directly optimizes the code. The purpose of the transducer ramping is to control the signal after passing through a transducer, rather than the code itself. To control the signal, the characteristics of transducer must be taken into account, so that the signal is optimized for the target shape.

Another major limitation using the PAPR ratio in the error function is that the error will scale with the ratio. For a non-constant g ramp function, this will result in the value peak error scaling with the magnitude of the ramp function. An improved implementation provides a constant envelope of error, such that the scaled error is also a scaled constant envelope. This can be implemented using a difference equation.

Transducer ramping using difference equation

To implement a constant envelope of error, a difference equation is used between the magnitude of each point in the code, and a ramping function scaled by the norm of the code. This is shown in Equation 3.15.

$$e_{j,p2a} = a^*_{t,j} a_{t,j} - a^*_t a_t \frac{1}{N_t} g_{r,j} \tag{3.15}$$

where a_t is a column vector for the signal of a code after passing through the transducer. The point j in the signal is optimized to match the shape of the ramp function.

To optimize the code based on this function, the error function is defined as $e^2_{j,p2a}$, and the gradient of this function is used in the gradient descent method. The gradient is calculated by the following derivation. The code a passes through the transducer, described using the convolution matrix for the impulse response of the transducer, $\mathbf{H}$.

$$\begin{aligned}
a_t &= \mathbf{H}a \\
a^*_t &= (\mathbf{H}a)^* = a^*\mathbf{H}^* \\
e_{j,p2a} &= a^*_{t,j} a_{t,j} - a^*_t a_t \frac{1}{N_t} g_{r,j} \\
\frac{\partial e^2_{j,p2a}}{\partial a^*_1} &= 2e_{j,p2a} \frac{\partial e_{j,p2a}}{\partial a^*_1} \\
\frac{\partial e_{j,p2a}}{\partial a^*_1} &= \mathbf{H}(\mathbf{j},:)^* a_{t,j} - \mathbf{H}(\mathbf{j},:)^* a_t \frac{1}{N_t} g_{r,j}
\end{aligned}$$

From these equations, both the error and the gradient of the error can be calculated, so the gradient of the squared error can be calculated and used in optimizing the codes.

The optimization strategy can be implemented in a similar manner to other strategies in the optimization, minimization of peak error. However in some cases this approach was found to have limitations. The impulse response of the transducer was found to concentrate a large proportion

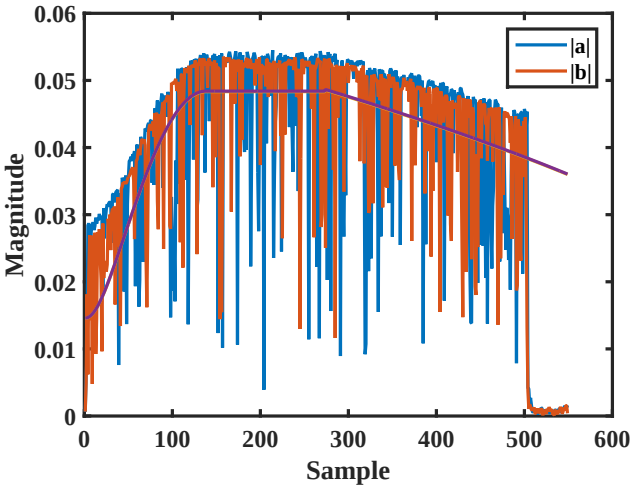


Figure 3.3: Optimization of two signals based on a ramping function, minimizing the positive peak magnitude of samples.

of the energy in the beginning of the impulse response. This resulted in a relatively large error at the end of the code, as the tail of the impulse response for the transducer would pull the signal down to zero, resulting in the maximum error being consistently located at the end of the code. This resulted in the optimization only optimizing the end section of the code.

An alternative approach is to optimize only the positive peak magnitude of samples in the code, which would ignore low values, such as the tail of the impulse response. An example of a signal optimized with this approach can be seen in Figure 3.3. The positive peak magnitudes of each signal are optimized for the ramp function shape, however the lower points are not. To maximize power, the amplitude should be as close to the ramp function as possible, including the negative peaks.

Another approach is to maximize the sum of the squared errors. This is effectively optimizing for the ℓ_2 norm of the signal, and allows for optimizing multiple points at once, but requires calculating and implementing the error gradient for each point in the signal in each iteration. The result of this optimization can be seen below in Figure 3.4.

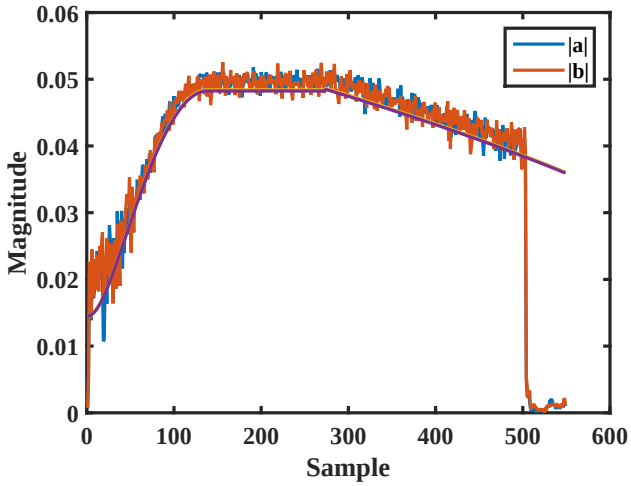


Figure 3.4: Optimization of two signals based on a ramping function, minimizing the ℓ_2 norm.

The signal follows the ramp function, and both positive and negative peaks are close to the target shape. The signal is still pulled down to zero at the end of the sequence, however it does not significantly affect the optimization of the other sections in the signal. This method of optimization, minimizing the ℓ_2 norm, was found to be the most effective for optimizing the ramping of the signal, and so is implemented in the optimization.

The gradient functions presented in this section provide a method to generate codes with optimal sonar properties in a matched filtered sonar system. An implementation of the optimization is found in Appendix A.1.2, and a method to efficiently use this optimizer is found in Appendix A.1.1. This method of generating codes assumes the sonar system can transmit code samples near perfectly, after taking into consideration limitations such as power scaling and transducer characteristics. However from Section 2.3 it was found that a digital transmitter complicates the accurate transmission of these codes, which if uncompensated will cause changes to the code's properties in the sonar system. To mitigate this, an algorithm was

developed to optimally map a sequence on to the digital transmitter. This is explored in the next section, Section 3.2.

3.2 Digital Transmitter

Limitations of the digital transmitter have been explored earlier in Section 2.3, where the most significant problem was due to the non-regular time characteristics of the cycle lengths. To transmit a code, the digital transmitter produces full cycles at roughly the centre frequency, then shortens or lengthens to achieve phase characteristics at the sample points. The amplitude is also adjusted to meet amplitude requirements for a sample.

The main problem is in the phase characteristics. By changing the length of one cycle, every cycle afterwards is affected. This suggests that each sample point should be optimized in order of the time of transmitting. However, if cycles are not carefully designed then subsequent sample points may be aligned with the beginning or end of a cycle, such that changing the length of the cycle has very little effect on the phase of the sample.

3.2.1 Mapping / Optimization Details

Digital Transmitter estimation

Before optimization can begin, it must be possible to observe the output of the digital transmitter when driven by a set of control coefficients. There are two properties to observe in the baseband signal after downmixing the output of the digital transmitter. These properties are the wrapped phase and the amplitude of samples in the signal. These properties need to be accurately estimated, to determine how the output affects the matched filter properties, which in turn indicates how to improve the properties.

Standard techniques simulate the transmitting process in the passband, then downmix the signal. For the digital transmitter, this would involve

mapping the coefficients to an upsampled signal, then downmix the signal and filter it to produce a base band equivalent signal for comparison in properties. It was found that the upsampled signal used significant memory and took significant time to run, even for short sequences. This was because the passband signal was upsampled by as much as 500 in some cases, so relative to the base band signal the sequence was extremely long. To effectively use this in later optimizations this simulated digital transmitter would need to be improved.

A prior designed algorithm, which was provided for the project, had aimed to solve this problem. The algorithm was designed to process the control coefficients as a frequency per cycle, then spline between the non-regular intervals of cycles to get sample information at the sampling rate. The algorithm is as follows.

1. Build up sequence of time lengths of cycles
2. Build timing vector based on cycle points
3. Invert time lengths to get frequency vector of each cycle
4. Spline between frequency vector points based on the timing vector
5. Spline between amplitudes of cycles based on the timing vector
6. Integrate the splined frequency to obtain phase, and combine with splined amplitudes to get sample information.

This pre-designed process provides an option to quickly build a set of frequencies of the signal, and integrating provides phase information. However, upon testing the algorithm it was found to not provide accurate phase information, especially around edges of the sequence. This was likely due to the combination of the spline operation and integrating operation. Splines are a tool used for interpolation, and because of this there can be a small error associated with the interpolated result and the true result. This error, when integrated, can result in large errors later in the sequence. Phase information is the important information to extract from the signal, as it is used in the baseband sequence which in turn is fed into the matched

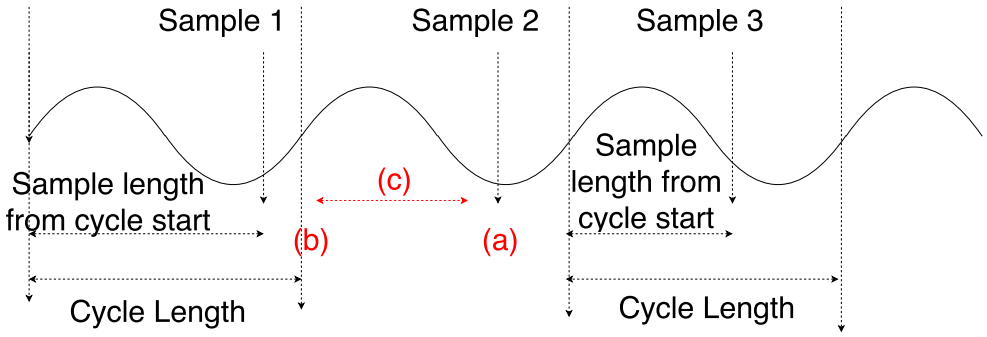


Figure 3.5: Procedure to estimate samples from a Digital Transmitter, at a low computational cost.

filter. To accurately observe the matched filter from the digitally transmited sequence, a more accurate estimation is necessary.

An alternate approach was developed to target accuracy in phase and amplitude information of the baseband sequence from a digital transmitter. For phase estimation, the phase was observed at a sample point as the ratio of cycle start to sample point, relative to the cycle start and cycle end points, as illustrated in Figure 3.5. The digital transmitted transmits whole cycle sinusoidal waveforms. The phase of a sinusoidal waveform changes linearly relative to time, and so a linear ratio of time from cycle start to sample point relative to the cycle length gives an indication of the phase. Converting this ratio relative to cycle length to a ratio relative to 2π provides phase information for a sinusoidal waveform.

Using the cycle ratio, the phase information can be extracted. This approach avoids the upsampling operation, which reduces memory usage and reduces the data that needs to be processed, significantly improving the processing time. Since the process skips the signal construction at the pass band, there is also no need for additional processing such as down mixing or filtering.

Amplitude in the digital transmitter is based on the amplitude in the input sequence. Earlier derivation in Section 2.3 showed that when the

digital transmitter sequence is filtered for the first harmonic, the AMP coefficient maps to amplitude according to Equation (3.16).

$$\text{amplitude} = sin(\frac{\pi}{2}\frac{AMP}{AMP_{max}}) \tag{3.16}$$

where $0 \leq AMP \leq AMP_{max}$.

For estimation, the amplitude coefficient was taken from a single cycle that overlapped the sample point of interest. This was then converted to an amplitude as per (3.16). This process is very efficient as the cycle that overlaps a sample point is already identified in the phase estimation, so the only additional processing is to parse a coefficient through a sine function. A disadvantage of this amplitude estimation is that it only takes the overlapping cycle into account. If there are several cycles per sample period then the cycles are likely to have an effect on the amplitude, especially if filtered. Empirical tests showed that these effects did not produce significant errors.

To implement the digital transmitter estimation system, a script was produced that tracked the baseband sample points and the end points of each cycle for a given set of coefficients. These variables listed baseband sample points relative to the upsampled rate, and the cycle end points also relative to the upsampled rate. Figure 3.5 is labelled (a) to (c) to indicate order of the algorithm, to determine phase properties. Amplitude was then calculated based on the cycle overlapping the sample point. The algorithm to determine digital transmitter baseband properties is as follows:

(a) Sample point identified, intercepting the current cycle.
(b) Previously completed cycle identified to reference the current cycle start, and cycle length of the current cycle identified.
(c) Difference between cycle start and sample point measured, taken as ratio relative to current cycle length.

The cycle's ratio is converted from relative to cycle length to relative to 2π, to obtain phase, and the amplitude is taken as the amplitude coefficient

of the current cycle passed through the function earlier in Equation (3.16). This tool was found to be fast, as well as accurate. For comparison, a short random sequence was generated, upsampled then downmixed, then compared to the digital transmitter equivalent sequence. These were found to be equivalent in phase and amplitude.

This efficient tool can now be incorporated into other algorithms as an efficient method to check the digital transmitter output, and to determine error between output and a target input for the mapping.

Mapping Options

Estimation of the digital transmitter provides a powerful tool for optimization of codes on this transmitter. A mapping approach was used to convert codes from the matched filtered optimization to a set of control coefficients for the digital transmitter. Initially a mapping seems trivial, however the non-linear nature of the digital transmitter adds several complications.

Direct mapping onto the digital transmitter is not possible, due to the non-fixed nature of the transmitted cycles. Cycles are not aligned with the sampling rate, and so it is not simple to determine which cycle affects a particular sample point. For long cycles spanning multiple sample points, a particular cycle could control the properties of these sample points, and as they are all controlled by a single cycle the sample points will be fully coupled. This limits what baseband properties these sample points can attain. The length of cycles also has an integrating effect on phase. Increasing the length of one cycle causes a delay in every subsequent cycle, which will affect the phase of every subsequent cycle. This cumulative effect must be taken into account when designing a sequence on the digital transmitter.

It is difficult to justify direct optimization of the control coefficients based on peak error. As discussed with direct mapping, a cycle can span several sample points which causes these sample points to be fully coupled, limiting what baseband phase and amplitude properties can be obtained.

This spanning of several sample points introduces an error that has a fixed minimum range.

In peak optimization, if other points of error are optimized below half the range, then the optimization algorithm will always target one of the extremes of these fixed points of error. As one point is optimized, the other worsens and vice versa. This causes an oscillation in the optimization algorithm, preventing further optimization.

Take an example of a single cycle spanning two samples. The two points will be fully coupled, so a change in phase of one sample results in the same change in phase in the other sample. If the absolute error of the first point is optimized for phase, and the error in phase for the second point is the negative of the first, the error in the second sample will increase as the first sample's error decreases. If the second point is now targeted for optimization, the reverse happens with the error increasing in the first point. These points will oscillate in optimization until the absolute error is equal for both points, after which the optimization will target either point without noticeable improvement. This locks the algorithm, preventing any other point of lower peak error from being optimized.

To overcome these limitations, a sliding window optimization was used to map phase and amplitude to control coefficients. This provides the advantages of optimization, while overcoming any potential points of oscillation or fixed errors. This also mitigates the integrating effect of phase. In peak minimization optimization, optimization effort may be spent on later parts of the sequence, and these points could then be shifted by changing a cycle early in the sequence, undoing the optimization from earlier iterations. Taking only a subset of samples, and optimizing in order of transmitting prevents this, as once a point is optimized later samples will not affect its properties. This process improves efficiency, optimizing only when a sample point is unlikely to be affected by others.

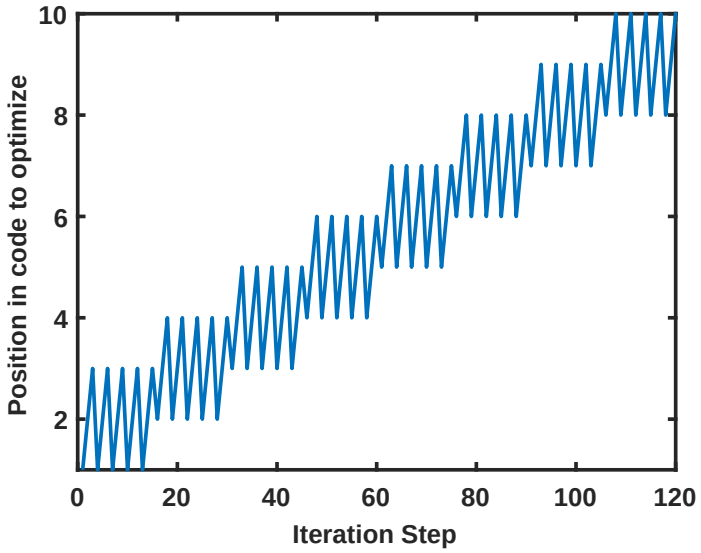


Figure 3.6: Example of a walk function used in optimization of mapping a sequence onto the digital transmitter.

Mapping Optimization Algorithm

The mapping optimization begins by setting up a vector of phase control coefficients at the centre frequency, and a vector of zero phase amplitude coefficients. A walk function is set up to loop over a small window a fixed number of times, then slide the window along by one sample and repeat the loop. This walk function is shown in Figure 3.6. The walk function determines which point in the target sequence should be optimized at the iteration step.

For each iteration step, the cycle containing the sample point is identified, and phase error calculated for the point. The error gradient is calculated below.

The relation between cycle ratio and phase ratio is shown in (3.17). This ratio is rearranged for HPW in (3.18), to indicate what the target HPW should be to obtain the correct phase. The phase is padded by a cycle, 2π, to ensure stability in the resulting gradient function. Note that although

the required coefficient for a particular sample is known from this result, it is unknown how neighbouring samples will affect it at this stage, hence the use of an optimization instead of a mapping. The HPW control coefficients control all cycles before and including the current cycle, and so the required change is taken in (3.19) as the amount m must change to meet phase characteristics. This is taken as the target HPW minus the current HPW. The equation is factorised in (3.20) and simplified to provide the final error gradient in (3.21).

$$\frac{m}{\text{HPW}} = \frac{2\pi + \Delta\phi}{2\pi} \tag{3.17}$$

$$\text{HPW} = \frac{2\pi}{2\pi + \Delta\phi} m \tag{3.18}$$

$$\Delta\text{HPW} = \frac{2\pi}{2\pi + \Delta\phi} m - m \tag{3.19}$$

$$= \left(\frac{2\pi}{2\pi + \Delta\phi} - 1 \right) m \tag{3.20}$$

$$\Delta\text{HPW} = \frac{-\Delta\phi}{2\pi + \Delta\phi} m \tag{3.21}$$

where
m is the length from the start of a cycle to the sample point,
$\Delta\phi$ is the phase error between the phase of the digitally transmitted signal at the sample point and the phase of the target sequence at the sample point, and
HPW is the half-pulse width, half the length of a particular cycle in the digital transmitter, and is the control coefficient for phase.

The error gradient can now be used to optimize a set of cycles for properties. A set of cycles governing the properties of each sample point are obtained, then optimized using the gradient. The cycles are identified as every complete cycle that is after the preceding sample point and before the current sample point. These cycles are all shifted by a small perturbation of

the error gradient, to optimize towards the correct phase. The amplitude of these same cycles are also set as a direct mapping, according to the previous equation describing the amplitude control coefficient (3.16). The inverse of this amplitude function is used to directly map the AMP control coefficients.

The walk function iterates across each sample point, optimizing the points. If the error step is less than a single upsampled transmit sample that can be generated by the digital transmitter, then the walk function skips the optimization step. The upsampled transmit rate of the digital transmitter can only transmit whole samples, so any fraction of a sample will be lost through rounding. Any further optimization would be undone through rounding, so for efficiency points are skipped when they are optimized to within a single sample.

This algorithm iterates through the entire sequence, optimizing each sample point to map phase and amplitude optimally. An implementation of this optimization can be found in Appendix A.1.3. The process maps as accurately as the algorithm allows, however it has been noted that few cases the mapping is not perfect, producing occasional errors in the sequence that become evident in the resulting matched filter characteristics. To overcome these imperfections, a mis-matched filter is designed to correct the properties.

3.3 Mis-Matched Filter

The mis-matched filter is very similar to the matched filter, but modifies the comparison vector to adjust the filter characteristics. It is notably useful because the filter itself is not limited by the constraints of transmitting, as the filter is implemented digitally. This provides the opportunity for significant improvement in the filter.

The most significant gain in using the mis-matched filter is to filter for the digital transmitter system. The output signal from the digital transmit-

ter has been shown to closely match the desired output, but error still exists that results in the designed matched filter properties being not optimal. Optimizing the mis-matched filter for the output of the digitally transmitted sonar codes allows for the error to be eliminated, at the cost of additional optimization and complexity in storing the filter coefficients.

Producing these codes is easier than for a matched filter, as the set of sonar codes is static. Optimizing a filter against static codes is a convex problem, unlike for matched filters, and so the use of a mis-matched filter allows for a wider set of tools to use for optimization.

3.3.1 Semi-Definite Programming Tools

The use of static codes to optimize a mis-matched filter simplifies the problem to a vector variable cross correlated with constant vectors. This problem is shown in Equation (3.22). In implementation, A would typically include the 0 angle pulse to detect, and several extreme angles for interfering pulses between 60 and 70 degrees. The result of multiplying the concatenated correlation matrix with the control coefficients will be the matched filter plots concatenated one after another without overlap. This provides a simple output to quickly locate the maximum.

$$\min ||Ax - T_1|^2 T_2|_\infty \tag{3.22}$$

where,
x is a N by 1 vector for filter coefficients to optimize,
A_{11} is $(2N-1)$ by N correlation matrix for a single correlation, which is used in A,
A is made up of several correlation matrices A_{ij} concatenated together, one for each expected received pulse.
T_1 is the template for ideal correlation, in detection correlation this is a kronecker delta function and for rejection correlation it is 0, and
T_2 is the template scaling function, for scaling different regions for ideal

correlation. For detection correlation this controls the ratio of main lobe to side lobe peaks, and for rejection correlation this is flat to evenly minimize all cross correlation.

This problem is a strictly convex problem, it contains a linear function that is applied through an absolute square function, and multiplied by a constant function. This function, in standard form, can now be solved for filter coefficients using semi-definite programming (SDP) tools.

CVX is a toolbox that enables convex optimization of convex problems presented in standard form [18, 19]. The standard form implies specific rules, which once met allows for a useful tool that can quickly find an optimal solution to a problem. Using the toolbox, solving convex problems is relatively simple as it requires only the objective function as well as defined parameters. Implementation of the solution to the mis-matched filter using CVX can be found in Appendix A.1.4.

CVX provides a simple yet powerful method for solving convex problems, especially useful for finding optimal filter coefficients for the mismatched filter. However there are some limitations in using CVX for sonar. In particular, the memory usage for solving complex convex functions is significant. The correlation matrices for each code are already of significant size, in some cases of the order of 5000 by 10000 and to use them in the CVX toolbox the real and imaginary parts are separated in memory, as part of the operation. This doubles the number of matrices, and as there is a squaring operation the size is doubled yet again. For large code lengths, this quickly becomes a memory intensive solution.

When generating codes for sonar, it was found that for most cases CVX was able to be used, but for the longest length codes memory limitations were reached. Because of this, another solution was needed, one that did not have the same memory limitations as CVX or other SDP tools.

3.3.2 Gradient Descent Method for Mis-Matched Filter

Gradient descent is a memory efficient algorithm, that only requires searching for maximal points and an instantaneous gradient at the point of error to step. It is not as solution efficient as CVX, however the reduced complexity makes it ideal for longer length codes since it requires less computational resources to solve.

Implementation of the gradient descent method for mis-matched filters is similar for matched filters, however when optimizing the transmit codes are kept static and not optimized, and the transducer characteristics do not need to be included as they are not present in the digital system. An optimization implementing the gradient descent method for mis-matched filters is included in Appendix A.1.5.

The series of methods from this chapter detail solutions for generating codes that target several requirements for sonar, in matched filtering, digital transmitting and in mis-matched filtering. To ensure the codes meet requirements, and that they are generated as per design, several codes are generated and evaluated for performance in Chapter 4.

Chapter 4

Performance Considerations

Methods have been developed in Chpater 3 to optimize codes for various code properties, targeting real world applications in sonar. These methods aim to generate a set of optimal codes, however there are various parameters driving the optimization which vary the balance of output properties. For implementation, it is useful to understand the relationship between input parameters and output properties, so that codes can be designed for specific properties. It is also important to verify the codes meet the designed properties, to develop a robust system.

This chapter observes generated codes and their properties, for relationships between the input and output of the optimization. An example set of parameters are used to produce a set of codes, to ensure codes meet requirements for an example sonar system. Furthermore, comparisons are made to existing codes, as well as overall bounds for codes, to investigate the effectiveness of generated codes and to see how close codes are to an optimal solution.

4.1 Tradeoffs

The optimization for sonar codes minimizes the error for each property specified by the objective function. This minimization is dependant on the

input parameters, which are specified before optimization begins. Perturbing the various input parameters showed several changes to the output properties, suggesting relationships between input and output parameters. It is useful to investigate these relationships, as they provide information that can be used to achieve specific output properties once optimized, instead of reaching arbitrary minimization for a particular set of codes.

Multiple parameters have a significant effect on the output code and its properties. The weighting factors for optimization is one set of these parameters. The weighting factors affect the significance of error for each property, and so affects how each property is prioritized in optimization. The higher the priority, the lower the error relative to other properties. In practise this prioritization typically is at the cost of other properties controlled by the weighting, and so weighting one property higher causes that property to improve, but other properties to deteriorate in terms of error. It is expected that this trade-off between weightings is linear, as the weightings are combined in the objective function linearly.

Another significant parameter is code length. A lower settling point has been observed for longer codes, similar to how standard codes are limited by the Welch bound. This suggests a similar bound may exist for these multi-detection codes. For implementation, a shorter code is ideal as it allows for a shorter burst between pings and also is easier to store in the system. The relation between side lobe and code length provides useful information on code lengths for desired performance in a sonar system.

The size of the main lobe in autocorrelation also has a significant effect on code properties. The main lobe corresponds to the range for fish to fish resolution occurs, where detection can occur only if there is a small difference in the power of reflected signals. It was found through empirical testing that the size of this region affected side lobes significantly, and so is useful to know the relationship for achieving specific requirements. An increase in main lobe size would correspond to a reduced resolution in time for resolving differing power reflections while being able to resolve

signals of a larger difference in power.

From the above, the input parameters for the code are primarily based on the following parameters. Included are the default values per parameter, used in each of the subsequent subsections for testing and displaying the relation between parameters.

- Code Length = 500
- Weighting factors [Sidelobe, Power, Main Lobe] = [1 0 $10^{\frac{-40+3}{20}}$]
- Main Lobe Size = 10

For sake of completion, the following also affect the optimization to a lesser extent:

- Power scaling function
- Iteration step size
- Transducer Transfer Function
- Code start points

4.1.1 Code Length

The length of the code for sonar systems determines the duration of a signal. Signals in turn are limited by how long they can transmit in a sonar system. If the signals are too long, the reflections interfere with reception. Because of this, a longer code limits the rate that a sonar system can ping, and so limits what ranges can be detected as well as how often information is updated. For ideal systems, the ping rate would be high, and so the code length should be short.

Conversely, the Welch bound has indicated that longer code lengths provide better bounds on correlation, suggesting that for better properties a longer code should be used. This contrast in requirements and in bounds

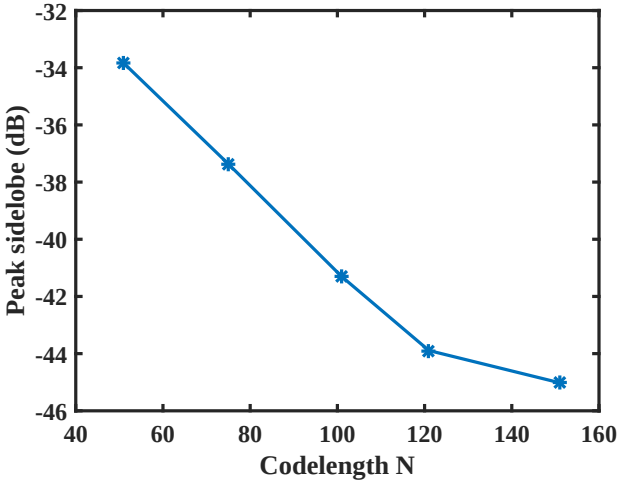


Figure 4.1: Relationship between length of code vs side lobe level in the matched filter.

leaves strict limitations on what can be implemented in a sonar system. Using an alternative detection system, with multi-level detection, separates the codes from the Welch bound, however there are still limitations on what code lengths can be used. It has been observed that longer code lengths still improve side lobe performance in a multi-level detection code, and so it is worthwhile investigating this relationship.

Empirical testing has shown that for longer code lengths, a lower side lobe level can be achieved, as shown in Figure 4.1. Here it can be seen that the codes follow an approximately inverse proportional relation, similar to the Welch bound. The codes were tested for a range of code lengths, and for a fixed main lobe size of 9 samples wide, from sample -4 to +4 relative to the central peak. The weighting factors applied were for weighting the main lobe and side lobe as standard, and zero weighting on power characteristics to observe the bounds on the code length.

From this relationship it can be seen that similar constraints apply to the multi-level detection codes as for standard codes for implementation in a sonar system. For the target of -40 dB for fish to bottom detection, it can be seen it can be achieved in this case for codes of length 200 samples. For

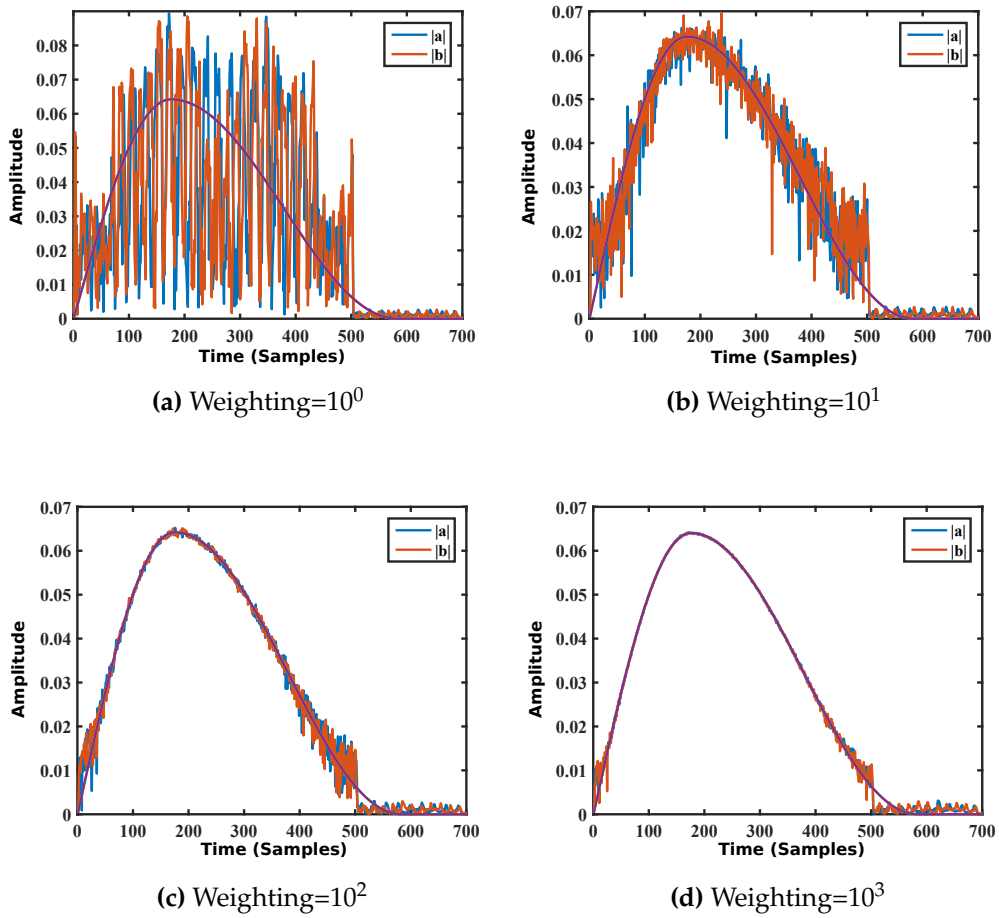
cases of shorter code length, the required side lobe characteristics cannot be met with the default parameters by varying only code length. Another approach must be taken to improve the side lobe properties.

4.1.2 Weighting

Weightings in optimization are arbitrary prioritizations in an objective function. The weightings prioritize the amount a function or sub-function should be optimized relative to other functions. Because of this arbitrary prioritization, it can be difficult to observe the effect a weighting has on the optimization other than improving or worsening a particular property. In some cases this is acceptable, as a simple attempt at a set of weightings can provide a good result which can then be adjusted to adjust the error of each property. In the case of sonar, there are a few points that can be noted to improve this weighting.

The weighting for side lobe level is the fixed point for the weightings. For design, the side lobe levels are targeted for -40 dB, to resolve fish from the sea floor. This fixed point removes some of the ambiguity with the relative weightings, allowing for a more quantitative look at the effect of other weightings. Conversely, weighting of the main lobe is based on the Rayleigh Criterion, requiring points outside the detection point to be below -3 dB. This is another fixed point for weighing. The relative ratio between the main lobe and side lobe is -3 dB : -40 dB, and so to optimize to these levels the functions should be prioritized based on this ratio. The weightings are applied as -3 dB for side lobes and -40 dB for the main lobe, to have a high weighing for side lobes and a lower weighting for the main lobe.

The weighting for power shaping is not based on a fixed weighting relative to the weightings for the side lobe level and the main lobe level. The weighing for power shaping controls the optimization of the magnitude of the code to match a target pulse shape, and this optimization is not related



(a) Weighting=10^0 **(b)** Weighting=10^1

(c) Weighting=10^2 **(d)** Weighting=10^3

Figure 4.2: Relationship between power weighting in optimization vs power shaping.

directly related to other optimizations. This allows the power shaping weighting to be varied separately.

Figure 4.2 shows the relation between the power weighting factor and shaping of the magnitude of samples. For weightings less than 10^0, the magnitude of the code was not significantly shaped. By increasing the power weighing, the magnitude of samples in the code deviate less from the target power shape. This suggests that a code can be shaped to a desired level of deviation from a target pulse shape, by adjusting the power weighting in design.

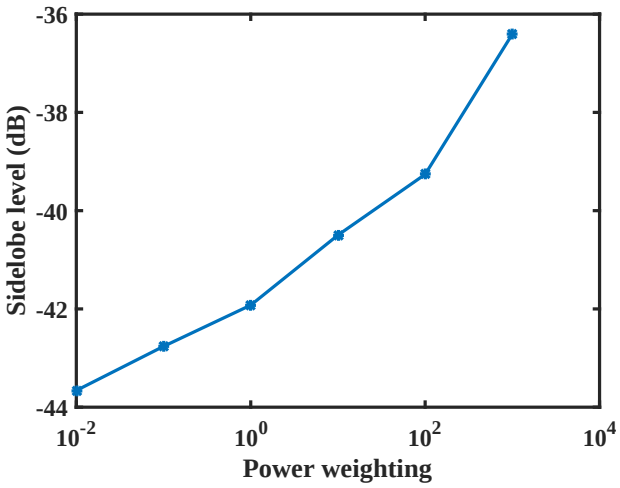


Figure 4.3: Relationship between power weighting in optimization vs side lobe level in matched filter.

One point to note is the change in deviation of magnitudes to the target shape, along the sequence. For relatively low values in the pulse shape, the deviation in magnitude is higher, and for relatively high values in pulse shape, the deviation is less. This suggests the power weighting is applied more heavily on higher values in the shape than lower. Although variable deviation is not ideal, the results show that the magnitude is still well shaped to a target shape, and so the variation is considered negligible.

As power weighting increases, the relative weighting for matched filter optimization decreases, which implies a tradeoff between the two properties. Figure 4.3 shows a consequence of this tradeoff, between power weighting and sidelobe level. This shows that as the power weighting increases, the side lobe also increases. Lower side lobes are desired, for fish bottom detection, so this tradeoff suggests that as the power weighting is increased to reduce deviation from a target pulse shape, the side lobe characteristics will worsen, and so a balance must be found to meet both properties.

Side lobe characteristics are a requirement set to be below -40 dB, yet

the power weightings are qualitatively set to be as good as possible. Requirements can be met by setting a fixed requirement at -40 dB for side lobes, then choosing the largest power weighting to meet this result will provide an optimal code. Observing the relation for this example, Figure 4.3, shows that this can be met for these particular parameters using a power weighting of about 10^1. The actual weighting can vary between parameters, however provides a methodology for designing codes with optimal power properties.

4.1.3 Main Lobe Size

The main lobe corresponds to a wide detection region, which enables better side lobe characteristics to be obtained in optimization of codes, as discussed throughout Chapter 2. Typical matched filtered systems target delta-like detection in autocorrelation, effectively a main lobe size of zero. By widening this main lobe, it was observed that by sacrificing correlation properties in the main lobe, the side lobe characteristics were improved.

In implementation, this main lobe can be compared with fish detection. For the detection of pings reflecting off two fish, where the reflected pings are of similar power, the main lobe does not affect detection as points are optimized to meet the Rayleigh Criterion, discussed in Section 1.1.2. For reflection of pings where one is from a fish and another the sea floor, the reflection off the floor will be up to 40 dB higher than the reflection off the fish and so any overlap in signals can only be detected in regions where autocorrelation of the stronger signal is below -40 dB, in other words regions outside the main lobe. To maximize the resolution of this detection, the width of the main lobe should be minimized.

Figure 4.4 shows the relation between main lobe width in samples vs the achievable side lobe level after optimization. Each point is calculated for a constant code length of 500, and with constant weighting factors. A random vector initialized each code, and optimization automatically

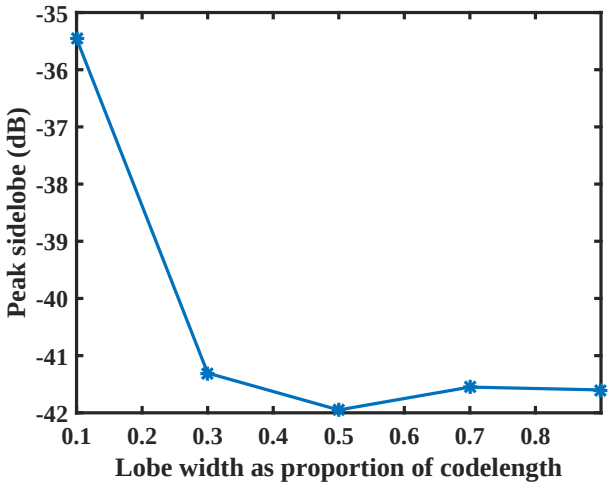


Figure 4.4: Relationship between main lobe size vs side lobe level in matched filter.

stopped when the change in error dropped below 0.01 dB per 10,000 steps. This was repeated for several random codes. Here it can be seen that, overall, increasing the main lobe width reduces the side lobe level, improving performance.

Compared to the sonar requirements, which for improved resolution aim to minimize the width of this main lobe, this tradeoff result shows a conflicting property that codes optimal for side lobes have larger main lobes. To design for ideal properties, a trade off must be made between increasing the main lobe width to improve side lobe performance, versus decreasing the main lobe to improve time resolution for pings of significant power difference.

The bounds and relationships of the various input parameters provide a scope for codes that are able to be produced. However this does not directly inform how codes perform in a real system. To get a better idea of realistic properties, it is useful to parameterized the codes for a real life system, and observe output properties.

4.2 Parametrization for a Sonar System

With the scope of parameters known, codes can now be produced to meet requirements for a real life sonar system. These requirements are based on target ranges desired for a fish detection system, however the applications are broad enough that the codes can be adapted as needed. The previous analysis on how parameters affect the output properties show what is feasible in implementing a sonar system, and so guide the efforts on what properties should be targeted, as well as the potential for improvement.

4.2.1 Requirements (including calculations)

The requirements of a sonar system are based off the desired detection range of a sonar system for recreational and industrial fishing vessels. These systems need to detect fish at both close range of the order of metres, as well as long range to the order of 1-2 kilometres. A table listing example ranges and requirements are shown in Figure 4.5. Here it can be seen that for various ranges of detection, there are different constraints on the rate of pings. This limits the time interval between pings, requiring shorter pings for pings of a higher rate. The rate of pings limits the length of the signal in each ping, and so puts constraints on the code and its properties in the matched filter. However from Section 4.1 it has been shown that there are limitations for the properties that can be achieved using optimization. Based on these varying requirements and limitations, it is difficult to produce a single code set that applies to all ranges, and so the sonar system has been designed to use multiple code sets for each set of ranges.

The example ranges highlight a few interesting requirements for the sonar codes. Note the requirements include the time for a signal to transmit from the boat to the target, and back, calculated using the following equation.

$$d_{\text{fish}} = \frac{ct}{2}$$

Depth	Ping rate	Pulse length	Min depth	Max depth	Range resolution	Range resolution	Range resolution	Range resolution
(m)	(pps)	(ms)	(m)	(m)	(two fish) (m)	(two fish) (ms)	(fish bottom) (m)	(fish bottom) (ms)
5	46	0.1	0.2	16.3	0.01	0.013	1	1.333
5	46	0.2	0.3	16.3	0.01	0.013	1	1.333
7.5	33	0.2	0.3	22.7	0.01	0.013	1	1.333
10	26	1	1.5	28.8	0.01	0.013	1	1.333
15	17.4	1	1.5	43.1	0.01	0.013	1	1.333
17.5	15	1	1.5	50	0.01	0.013	1	1.333
20	13.1	2	3	57.3	0.01	0.013	1	1.333
25	10.6	2	3	70.8	0.01	0.013	1	1.333
30	8.8	5	7.5	85.2	0.01	0.013	1	1.333
40	6.8	5	7.5	110.3	0.01	0.013	1	1.333
50	5.5	10	15	136.4	0.01	0.013	1	1.333
60	4.5	10	15	166.7	0.01	0.013	1	1.333
70	3.9	10	15	192.3	0.01	0.013	1	1.333
80	3.4	20	30	220.6	0.01	0.013	1	1.333
90	3.1	20	30	241.9	0.01	0.013	1	1.333
100	2.8	20	30	267.9	0.01	0.013	1	1.333
150	1.9	40	60	394.7	0.01	0.013	1	1.333
540	1.3	40	60	576.9	0.01	0.013	1	1.333
1000	0.7	40	60	1071.4	0.01	0.013	1	1.333

Figure 4.5: Table of ranges and requirements for a sonar system. This system aims to detect across both shallow and deep water, however can have separate modes for each set of ranges.

Firstly, for fish to fish resolution, the requirement is to resolve signals 0.013 ms apart, equivalent to about 0.97 cm apart in distance. Assuming the code is sampled at roughly 100 kHz, this corresponds to a sample resolution of about 0.013ms $\times$ 100kHz $=$ 1.3 samples. To meet the Rayleigh Criterion, to ideally resolve the two signals the overlap points should not exceed -3 dB. Assuming similarly shaped signals in the receiving filter, since the signals are based on the same transmit signal, the overlap point is likely to be equidistant from the two peaks, roughly 0.65 samples away. This is difficult to design in a code, as the signal is optimized and adjusted per sample, not per fraction of a sample. Instead, the requirements can simply be exceeded at the rounded requirement, at 1 sample out. This ensures that two fish can still be resolved with high accuracy.

To resolve fish from the sea floor, the signal must be able to be resolved while in the side lobe of a stronger signal. This requires side lobes to be below -40 dB to the peak. The requirement for this resolution is for 1m between targets, corresponding to a time resolution of 1.333 ms between signal peaks. At 100 kHz, this corresponds to 1.333ms $\times$ 100kHz $=$ 133.3 samples. In design of the matched filter, this corresponds to the main lobe between -133 to $+133$ sample offsets.

This resolution is less stringent than for fish to fish resolution, so much that it is larger than some of the required pulse lengths. This means that for shorter pulse lengths, lengths less than the time resolution of fish to bottom, the requirement is automatically met as matched filters outside an input signal are defined to be zero, or negative infinity dB. In reality it is based on the noise level, however this makes no difference to the required design as the matched filter is designed for rejecting noise. Short pulse lengths require no additional consideration for designing fish to bottom resolution, while longer pulses do have this fish to bottom requirement.

Pulse lengths significantly vary based on the required resolution. For short distances a high ping rate is needed, limiting the maximum pulse length available for design. For a depth of 5 m, a pulse length of 0.1 ms is

available. At 100 kHz, this allows for a code length of 0.1ms $\times$ 100kHz $= 10$ samples. Earlier analysis showed that a short pulse length allows for only a limited side lobe control, however after analyzing requirements on fish to bottom resolution the side lobe levels are only needed for code lengths above 133 samples when sampled at 100 kHz. For points above 133 samples, the side lobe is considered as the points outside the main lobe, and so for short code lengths above 133 samples there are only a few sample points needed for side lobe optimization, still providing a simple requirement in optimization. These characteristics in the requirements simplifies design requirements for shorter codes, while still allowing for longer codes to meet resolution requirements.

The previous analysis shows how requirements convert into parameters for code optimization, and so can be used to produce an optimal code for sonar. The process of producing the codes is outlined below.

4.2.2 Optimization output

With a set of requirements laid out, and an indication of property tradeoffs illustrated, a strong foundation has been set for analysis of the sonar codes. Before observing the final output it is useful to look at the stages of producing the codes, observing how the code's properties are set and improved at each stage. This gives an idea of how well each stage can optimize, as well as any weaknesses that need to be overcome.

The set of sonar codes are produced in 3 stages. These are:

- Matched Filter Optimization
- Digital Transmitter Mapping
- Mis-Matched Filter Correction

The process will use the example codes for detecting at a depth of 50-70m, for a pulse length of 10 ms. At 100 kHz, this corresponds to a code length of 1000 samples. The Rayleigh criterion needs to resolve signals

separated by 1.3 samples, and the main lobe sits at roughly -133 to +133 sample offsets from the main matched filter peak.

Matched Filtered Optimization

Optimization begins by inputting parameters. These parameters are a code length, using the pulse length times the sample rate. This example sets the code length as 10ms $\times$ 100kHz $=$ 1000 samples. It was found that the main lobe size could be reduced further than the requirement, achieving better resolution. The parameter for main lobe was modified to 65 samples out from the central peak, to achieve 0.666 ms fish bottom range resolution, or 0.5 m resolution for fish to bottom. Weighting factors are [1 λ_{PAPR} $10^{\frac{-40+3}{20}}$], where λ_{PAPR} is adjusted to provide power weighting without permitting side lobes above -40 dB.

Optimization using iterative methods step to reduce error of a vector and so, in some cases, not guaranteed to achieve a perfect solution. Instead, the error is reduced each step until the error is considered non-significant. This suggests that optimal codes may be optimized forever, however in reality a good solution is produced within a few steps, and so an arbitrary stop point can be set by observing the error not changing significantly. A more robust method in place is to set this stop point in the optimization, which stops the algorithm once the net improvement of error drops below a threshold. This threshold stepping is implemented by running the algorithm for roughly 10,000 steps, then comparing the error from before the steps to the error after, and if the improvement in error is above the threshold the optimization repeats for another set of steps. If below the threshold, it is unlikely that further significant optimization is possible, so the optimization stops. The large set of steps between comparisons allows for any potential overshoot and self correction, to prevent the algorithm from ending prematurely. This allows for multiple optimizations to be queued at once, so that a set of codes for each requirement can be generated with one optimization set up.

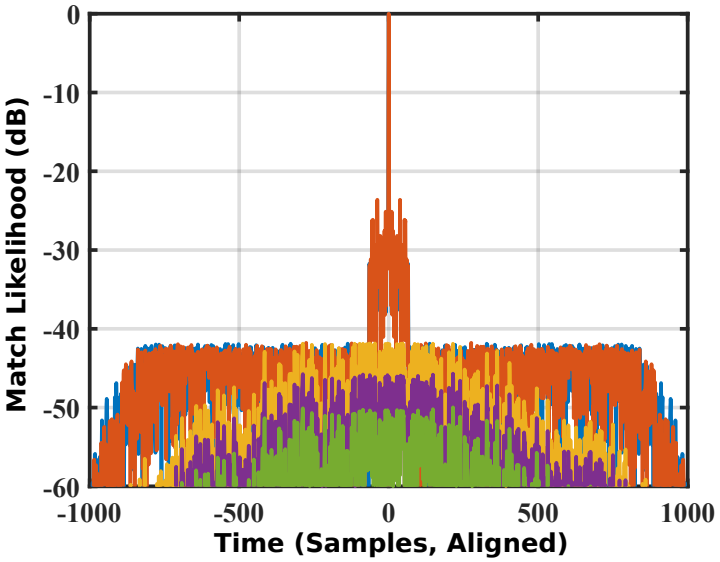


Figure 4.6: Optimization Output for the Matched Filter optimizer, observing the code's matched filter output.

The optimization was applied to the example parameters outlined earlier, optimized until there is no significant change in error. Properties observed in the optimized code are based on the matched filter, and the magnitude of the code. The matched filter provides observation of resolution requirements, while the magnitude provides observation of power requirements of the code. For the sonar code in this example, the properties are shown in Figures 4.6, 4.7 and 4.8.

Figure 4.6 shows the properties of both codes in the matched filter. This plot effectively compares a single received pulse compared in the matched filter, scaled based on the magnitude of the code. Note the side lobes sitting below -40 dB. Also, any cross correlation between codes at edge angles of the transducer is below -40 dB as well. A zoomed version of the same plot is shown in Figure 4.7, to show further details of the main lobe. Here it can be seen that there is a distinct main lobe between -65 and +65 sample offsets. Zooming in on the central peak, the offsets outside of the central peak sit

below -10 dB, which is well below -3 dB meeting the Rayleigh Criterion. For resolution, the code meets the expected properties in the matched filter, and so these codes are expected to perform as needed in the sonar system by correctly resolving any interfering signals.

Although not a direct objective for design, spikes in the main lobe are non-ideal. For clear resolution in the main lobe, any overlapping signals must be stronger than the highest point in the region, to avoid confusion with the main signal. Spikes in Figure 4.7 are sporadic, preventing overlapping weaker signals in the main lobe being resolved when below -30 dB. The main lobe is primarily optimized to meet the Rayleigh criterion, and using peak optimization continually targets the highest points next to the central peak. To reduce spikes in the main lobe, an additional region could be implemented for peak error minimization, targeting the main lobe excluding points near the Rayleigh Criterion. This would prevent the algorithm locking optimization near the central peak, smoothing out the main lobe region.

The magnitude of each signal after passing through a transducer is shown in Figure 4.8, and the pulse shaping function is shown as well. In this case an arbitrary shaping function has been used as a test function, however any function can be used to shape the pulse. This function used a sine wave to ramp up, then a cosine wave to ramp down followed by zeros to pad out the trailing transients of the transducer. This combination of functions provides a smooth waveform with a continuous gradient, while being simple to generate.

The signals' magnitudes in Figure 4.8 are shown to clearly follow the shaping function, ideally meeting the power requirements for the set of sonar codes. Based on the matched filter properties, the side lobes could be sacrificed to further improve magnitude shaping. The side lobes are targeted to be below -40 dB, and any improvement beyond this is beneficial, however does not provide any direct improvement from the requirements. One alternative for improvement is to limit the optimization to -40 dB then

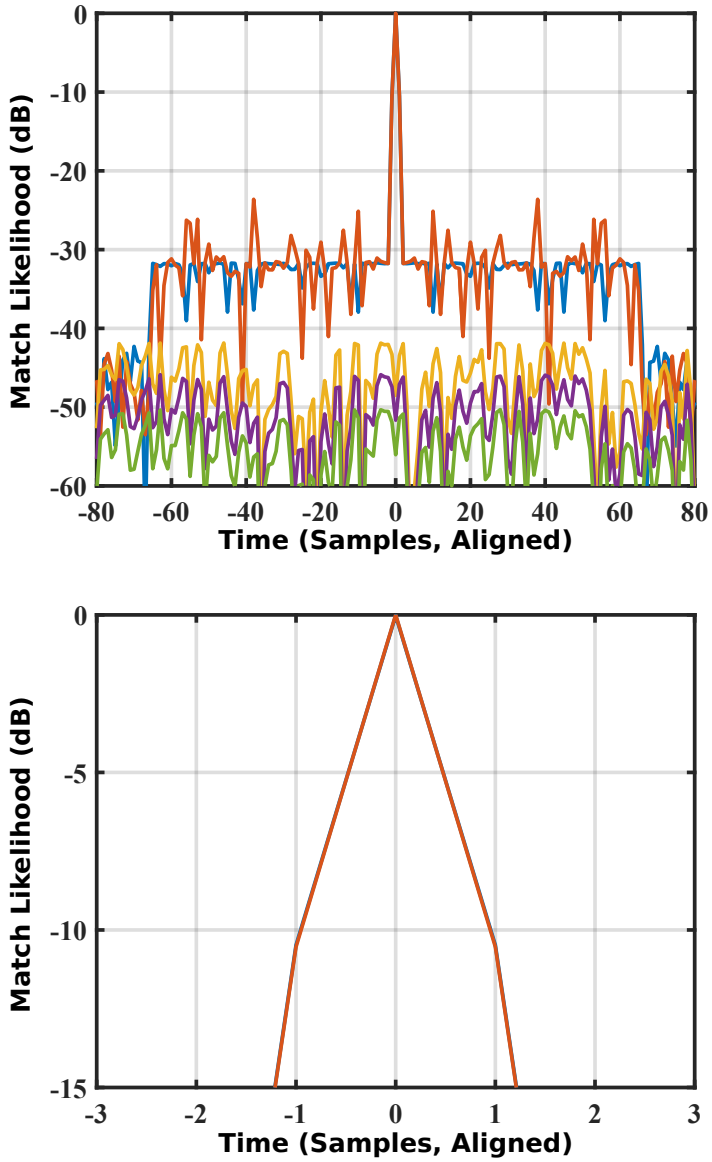


Figure 4.7: Optimization Output for the Matched Filter optimizer, observing the code's matched filter output. Zoomed to show details of main lobe.

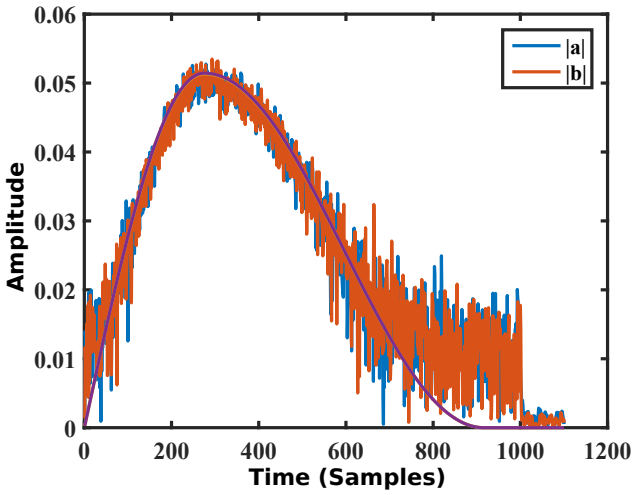


Figure 4.8: Optimization Output for the Matched Filter optimizer, observing each signal's magnitude properties after passing through a transducer. Blue and red show the magnitude of each code, and purple shows the shaping function used to shape the magnitude of each code.

using further optimization for magnitude shaping, which can potentially improve the shaping even further. At this stage, results have proven more than satisfactory, so further improvements to the algorithm are optional.

These example results show the output of matched filter optimization meet the requirements sufficiently, and are a good indicator for the final codes and the properties expected after transmitting. The next stage looks at how this matched filtered code translates to a digital transmitter, to apply to a sonar system.

Digital Transmitter Mapping

The digital transmitter mapping takes a set of samples and maps it to a waveform that the digital transmitter can transmit, using coefficients to control the waveform. This mapping aims to reproduce the input waveform as accurately as possible, such that samples match the design. The accuracy of the samples is based on the phase properties and the amplitude properties

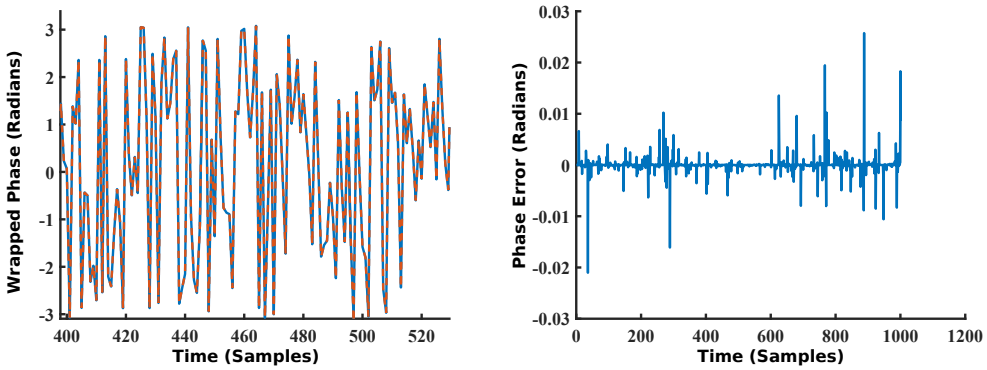


Figure 4.9: Optimization Output for the Digital Transmitter mapping, observing the phase properties of each code. Left shows a subsection of ideal phase (blue) and mapped phase (red). Right shows phase error of each sample in radians, for the digital transmitter output and the target output.

of the samples. For this example one digitally transmitted sequence is observed, however similar properties are present in the second code.

The set of sonar codes for 50-70m is mapped and evaluated based off the output from the matched filter optimization. For illustrative purposes a subset of the phase characteristics of both the original matched filter optimizer output and of the digital transmitter output are included in Figure 4.9. Here it can be seen that the mapped phase is very similar to the original phase. Observing the whole sequence, the error does not go beyond 0.03 radians. This suggests the phase is mapped on the digital transmitter accurately. Further accuracy can be obtained by optimising for more iterations, and with smaller step sizes.

The amplitude of the code is directly compared in Figure 4.10. This plot compares the ratio of original sequence amplitude to the amplitude of the digitally transmitted signal. An accurate mapping should show a flat, constant ratio across samples to show the code scaled equally across the whole sequence. For most of the sequence the ratio of amplitudes is constant near 8.7, however there are peak errors at points that can be as large as 50% of the amplitude. This is not ideal, and occurs due to cycles

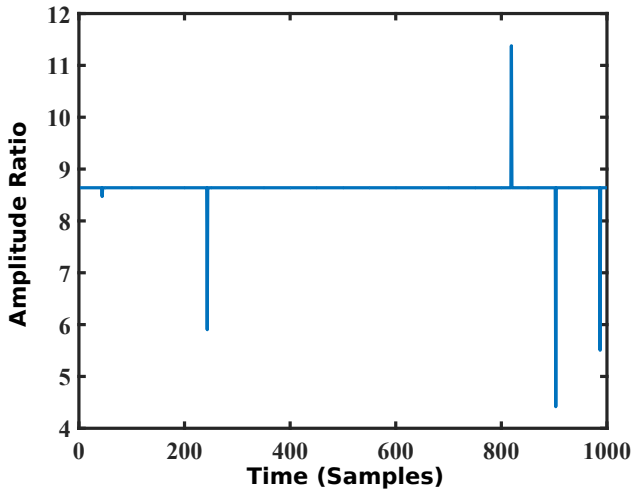


Figure 4.10: Optimization Output for the Digital Transmitter mapping, observing the amplitude properties of each code. Amplitude properties represented as a ratio, each sample's amplitude for ideal sonar divided by each sample's amplitude in the digital transmitter.

spanning several sample periods. A single cycle has a constant amplitude, and so when it spans multiple periods the constant amplitude is spread across these periods. If these samples are designed for different amplitudes, then there will be some error across the samples. This error is always present if cycles span multiple samples, however later results will show this does not have a significant impact on results.

For a sanity check, it is useful to also look at the distribution of coefficients for both cycle length and for amplitude control. Figure 4.11 shows the distribution of cycle lengths. Note the sequence is followed by a constant set of cycles, these are at the centre frequency of the digital transmitter and are effectively zero padding the sequence.

To get a frame of reference for the coefficients, it is useful to look at how the coefficients compare to the transmit and sample rates of the system. The clock rate of the transmitter is about 102.5 MHz, which controls the lengths of each digitally transmitted cycle, in relative sample lengths. This can be

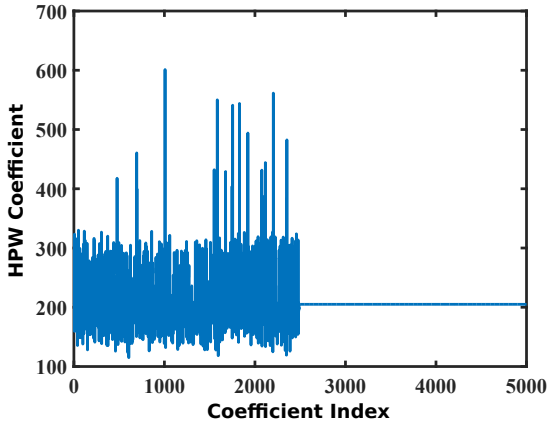


Figure 4.11: Optimization Output for the Digital Transmitter mapping, observing the distribution of coefficients for half cycle lengths.

used to calculated a ratio of the clock rate over the rate in question. For centre frequency, this corresponds to 102.5 MHz / 250 kHz = 410 samples, and for sampling frequency about 102.5 MHz / 100 kHz = 1025 samples. Converted to half pulse widths, which are control coefficients in the digital transmitter, these are 205 samples for centre frequency, and about 512 for sampling frequency.

Observing the generated coefficients in Figure 4.11 , the sequence shows most of the half cycle lengths distributed around the centre frequency, about 205 samples. There are also a few spikes in long lengths, above 400 samples. Several of these spikes in lengths are longer than the number of cycles per sample, which is at about 512 samples, and so correspond to cycles spanning multiple samples. Since cycles are a fixed sinusoidal shape, the samples are coupled in amplitude and phase, potentially contributing to error. Identifying these points indicates areas of potential improvement for the algorithm, and in the digital transmitter, which can be investigated further in future work.

The amplitude coefficients are shown in Figure 4.12. The amplitude control is similar to the amplitude of the signal itself, however takes into ac-

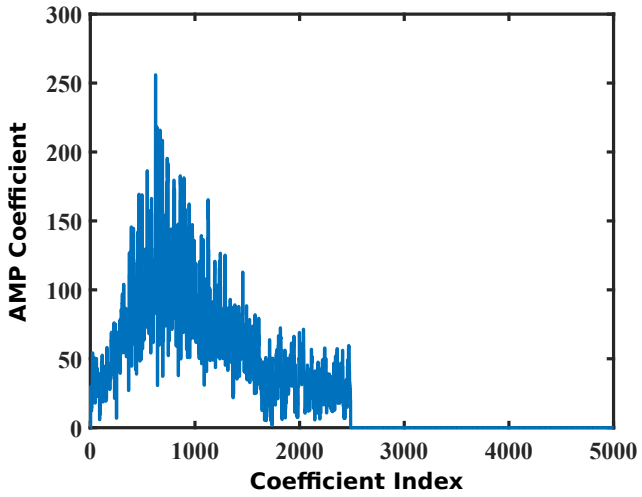


Figure 4.12: Optimization Output for the Digital Transmitter mapping, observing the distribution of coefficients for amplitude control.

count harmonic loss. This relationship is non-linear, however the mapping is monotonic increasing and one to one, as discussed in Section 2.3, so it is expected the amplitude coefficients follow a similar shape to the amplitude itself. This is evident in the figure, where the amplitude coefficients ramp up then down, similar to the original matched filter optimized sequence and in the shaping function.

This evaluation of the digitally transmitted signal shows signal characteristics very similar to the original match filter optimized signal. As a final comparision for this stage, the matched filter of the digital transmitted signal is shown in Figure 4.13. Here the matched filter characteristics are shown to be very similar, with side lobes below -40 dB, however a new set of spikes have been introduced outside the main lobe, close to the centre. This set of results is still usable, however can still be further improved. To realise this improvement, the receiving coefficients can be optimized using a mis-matched filter.

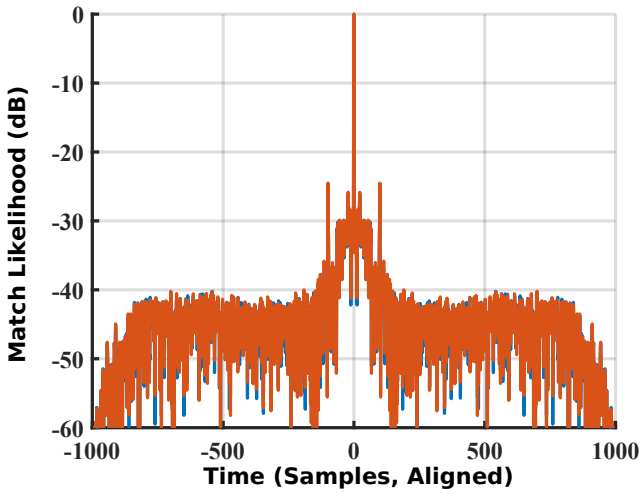


Figure 4.13: Matched filter of one code, comparing autocorrelation from matched filter optimization (blue) and digital transmitter mapping (red). Note the spikes either side of the main lobe, not originally present in the first stage of matched filter optimization.

Mis-Matched Filter

The digital transmitter was shown to transmit a designed code, however with few errors based on the constraints of cycles. This suggests that matching a transmitted waveform will not produce ideal properties as designed. However by using a mis-matched filter the filter properties can be improved. By mis-matching the digitally transmitted signal, the filter coefficients are obtained by a similar optimization as in the matched filter, but in this stage against a static code.

The results of this optimization are shown in Figure 4.14. This figure shows the matched filter properties meeting the ideal properties for resolving signals, similar to the matched filter properties in the first stage. The major differences are that this is now receiving a signal that can be transmitted in a high power system, using the digital transmitter, and that the filter coefficients are no longer constrained by the power scaling properties as in the transmitting code. The reduced constraints allow for further optimization and improvement, as evident in the mis-matched filter

response.

From this example it is shown that a code for a particular range is designed and optimized, in both transmitting and receiving. The process is repeated for other ranges. It should be noted that the digital transmitter is aimed for high power systems, for transmitting the larger distances in the requirements. This additional design constraint is not needed for short ranges, and so short ranges only require design of a matched filter.

4.2.3 Comparison against existing codes

Comparison of existing codes provide a good benchmark for the properties of the generated sonar codes. Existing codes such as Gold and Zadoff-Chu codes have useful properties with regard to autocorrelation and cross correlation, making them useful for the matched filter. Newer codes utilise these codes and more, providing additional properties that are useful for not only sonar but for communication and detection systems in general.

Chirp signals are a common signal used for sonar detection [6, 26, 45]. A chirp signal is a sinusoidal waveform that changes frequency over time. This has a fixed bandwidth of signal, and has typically good properties for auto-correlation and cross correlation. A linearly changing waveform makes it simple to generate, and fixed bandwidth is a useful benefit for designing on frequency dependant systems, such as with sonar transducers.

A comparison of a chirp sequence of similar length is shown in Figure 4.15a and 4.16. The signals tested were linear up-chirps, with parameters chosen to target as good performance as possible to compare against the optimized codes from Section 4.2.2. Each chirp was designed using a sampling rate of 100kHz, each given 10 kHz bandwidth and given frequency separation to allow for ideal cross correlation properties.

The auto-correlation and cross correlation of these two chirp signals are shown to indicate the potential performance in a matched filter. Overall the properties initially seem good, however the side lobes are still too

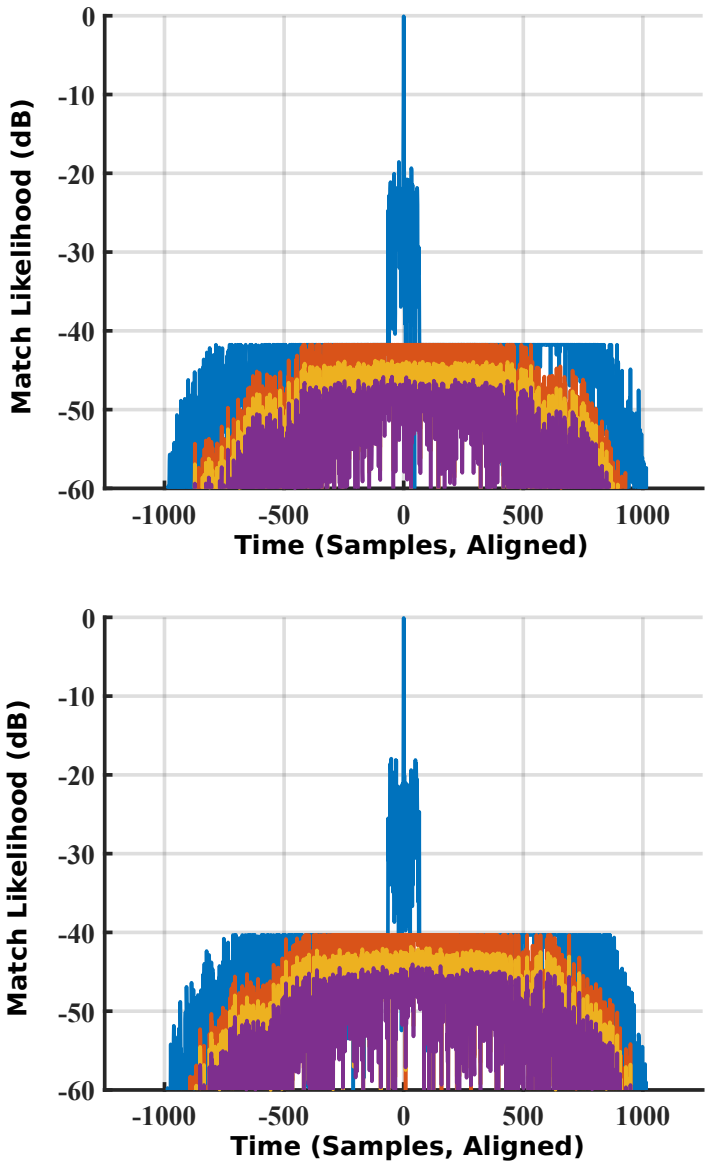


Figure 4.14: Optimization Output for the Mis-Matched Filter optmization, optimizing filter coefficients for receiving a digitally transmitted signal. Shown is the receiving filter characteristics for filter coefficients of code A receiving signal A and signal B (top), and code B receiving both signals A and B (bottom).

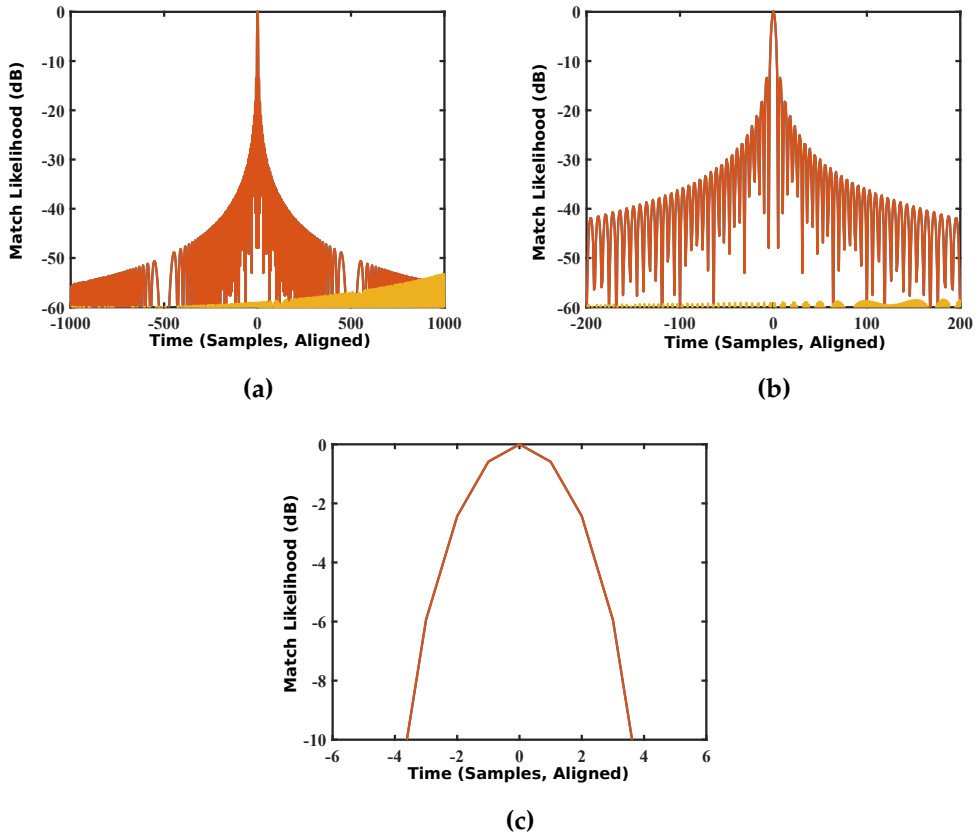


Figure 4.15: Observing the autocorrelation and cross correlation of two chirp signals. (a) and (b) indicate main lobe properties, with side lobes dropping below -40 dB beyond ± 170 sample offsets. Also note the wide peak (c) for the Rayleigh Criterion.

high. For the requirement of -40 dB, the side lobes meet these requirements outside offsets ± 170, which for a sampling rate of 100kHz is beyond the required resolution. Cross correlation properties are satisfactory in this case. For Rayleigh Criterion signal resolution, the main peak has points above -3 dB outside the 0 lag offset, making similar strength signal resolution difficult. Although the signal shows promising properties, it doesn't meet the requirements and so is not as good to implement as the optimized signal.

Another useful comparison is to introduce power scaling. Scaling the magnitude of samples affects correlation, and so it is useful to note if any additional considerations could make the chirp signal viable. A plot of correlation properties introducing this power scaling into the chirp is included in Figure 4.16. Here the signal shows much better side lobe properties, however the main lobe is now much worse for similar strength signal resolution. Although the chirp signal is easier to generate based on a simple function, the target properties are not met and so is not as good a sonar code as the optimized code.

Zadoff-Chu sequences are another class of sequences that provide useful correlation properties for code design [27]. The Zadoff-Chu sequence is based on a complex exponential sequence oscillating at a rate based on a prime number [4]. The use of this prime number ensures that values are not exactly the same as the phase wraps around, and so provides good correlation properties for every shift of the sequence. These sequences primarily have good periodic correlation properties, however the aperiodic properties are also notable.

Figure 4.17 shows the correlation properties of a pair of Zadoff-Chu codes, each with relatively prime coeffcients to generate the code. The relatively prime coefficients ensure good cross correlation properties. The figure shows initially that side lobe levels are worse for Zadoff-Chu codes than for chirp codes, however the main peak has a much sharper descent for non-zero offsets, making similar strength detection better. These codes

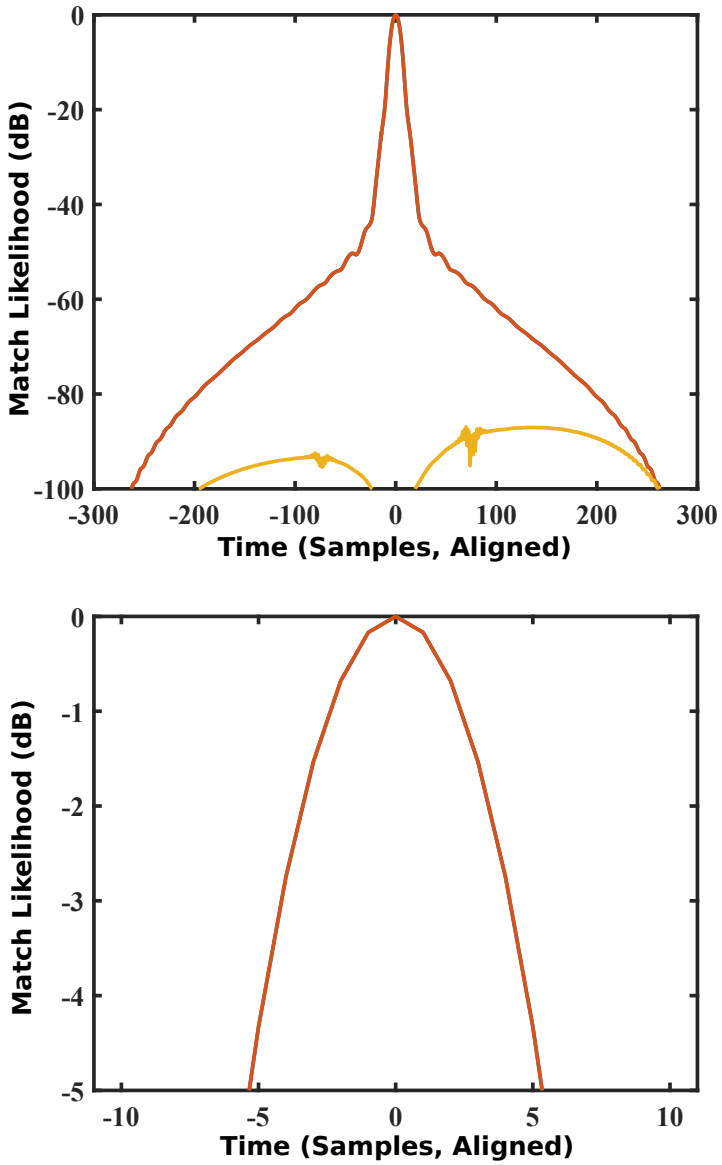


Figure 4.16: Autocorrelation and cross correlation of two chirp signals, frequency separated and power scaled. Note the wide main lobe, making resolution of similar power signals difficult for small offsets.

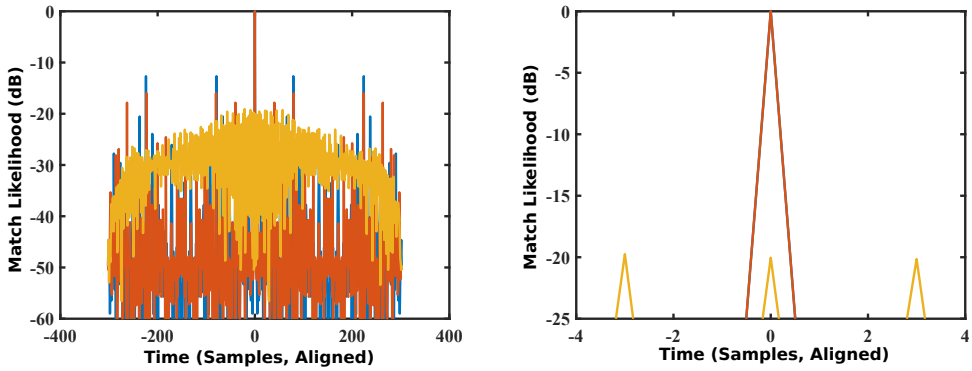


Figure 4.17: Autocorrelation and cross correlation of two Zadoff-Chu sequences. Note the spikes in correlation, and also the sharp peak detection around the central peak, good for similar strength signal resolution.

provide a better property in one regard, however at the sacrifice of another property.

Note that although the correlation properties appear worse than chirp signals, the errors are spikes. The Gradient Descent algorithm uses peak minimization to reduce error, and so if there are only a few peaks then there is minimal optimization effort to improve the code. This property suggests that Zadoff-Chu codes could be a good start point for optimization of codes using the gradient descent method. Empirical evidence supports this, showing that these codes take less iterations to optimize than chirp codes, or random codes. Because of this property, implementation has defaulted to use Zadoff-Chu codes as a start point, then optimize to produce ideal codes.

4.2.4 Comparison to bounds

The codes generated have been optimized for correlation and power properties in a sonar systems. Compared to existing codes, these optimized codes have been shown to perform better overall, in terms of resolution for sonar. However although the codes are relatively better, it is more powerful

to ensure codes are optimal for all possible codes. To evaluate this, and to further validate the performance of these codes, it is useful to look at the bounds.

The Welch bound, analysed in Section 2.4, shows the overall bounds on correlation for a set of codes. This bound, repeated below in Equation 4.1, shows the minimum peak correlation for autocorrelation excluding the zero lag peak, and for all cross correlation for all codes in a set. This provides an indicator of optimal properties that can be achieved in sonar.

$$c_{max} \geq \sqrt{\frac{M-1}{M(2N-1)-1}} \tag{4.1}$$

Observing the Welch bound, for $M = 2$ codes of length $N = 1000$, the bound on correlation is $c_{max} \geq -36.0173$ dB. The optimized codes have shown to be below these codes, achieving side lobes of -40 dB. Initially this would seem to violate the Welch bounds, however the main lobe in autocorrelation is not based solely on the zero lag offset, as required by the Welch bound, but is instead a region. Because of this, the Welch bound does not directly apply. Despite this, it is useful to know that the codes have properties better than other codes which rely on a peak only at the zero lag offset.

Further analysis of the bounds showed the modified Welch bound, as analysed earlier in Section 2.4.5. This shows a bound that allows for a main lobe region, providing a more useful metric for evaluating the performance of codes generated by this optimization. Note that this modified Welch bound is for a fixed level of correlation in the main lobe, however results of optimization have shown that the main lobe level is not fixed in the matched filter. As shown earlier in Figure 4.7, points near the zero lag peak are near -10 dB, while points further out are closer to -30 dB. Because of this non-fixed property, the main lobe is estimated to be between -20 and -30 dB on average, to enable a rough comparison.

Parametrizing the modified Welch bound using the optimization inputs

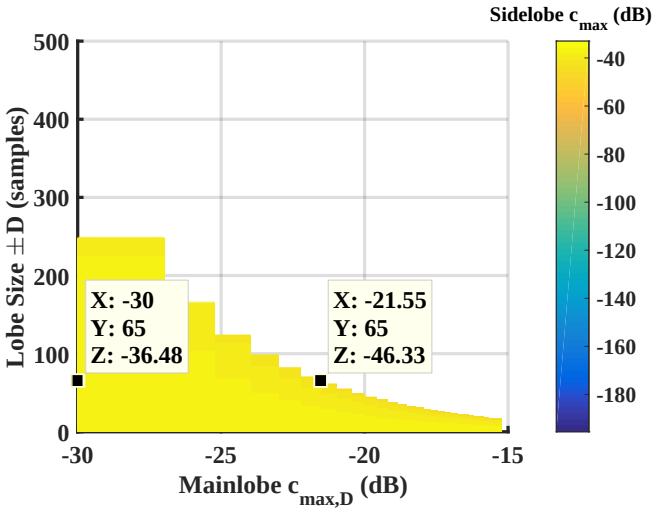


Figure 4.18: Modified Welch bound for 2 codes, each code length 1000. Points indicated show the lobe size of points ±65, and the various side lobe correlation bounds that can be set.

in Section 4.2.2, a relation of bounds relative to main lobe can be observed. This is illustrated in Figure 4.18. Here it can be seen that for a fixed lobe size, as the main lobe level increases the bound on the side lobe level decreases, making it possible to achieve better side lobe performance. Also note the white space towards the top right of the plot, this region indicates negative infinity dB, suggesting potentially zero side lobes and cross correlation.

Labelled on this figure is the main lobe size of ±65, used in the optimization. Here it can be seen the side lobe properties are between -36.48 dB to -46.33 dB, based on the assumption the main lobe correlation level is between -30 dB and -20 dB. A plot of correlation for a fixed lobe size of ±65 shows this relation in more detail, as shown in Figure 4.19. This plot shows that it is possible to achieve side lobes of below -40 dB using a main lobe of ±65, by setting the main lobe correlation high enough.

These bounds agree with the optimization output shown in 4.6. This earlier plot showed correlation side lobes and cross correlation settling at roughly 43 dB, which sits within the expected range of -36.48 dB to -46.33

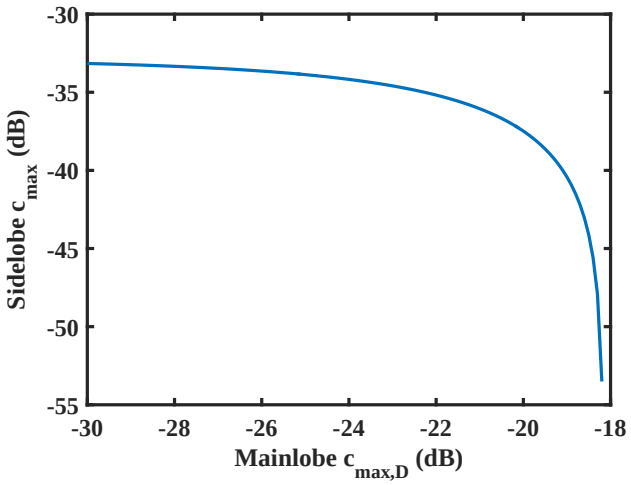


Figure 4.19: Modified Welch bound for $M = 2$ codes, each code length $N = 1000$, main lobe of $D = 65$ corresponding to ± 65 sample offsets.

dB from the modified Welch bound. This suggests the codes are close to optimal.

A noteworthy characteristic of the plot in 4.19 is the asymptote as the main lobe correlation level is increased towards -18 dB. In reality this is level in main lobe correlation is not ideal, as the main lobe corresponds to potential interference for fish to fish detection. Allowing a lower main lobe allows for detection of small signals in the main lobe of a strong signal, however If allowed to rise this main lobe detection will be limited. This introduces a new kind of trade off, however is more appropriate to analyse in requirements design rather than signal design, which is beyond the scope of this thesis.

Overall, codes have been shown to go beyond the performance of codes bound by the Welch bound, and have also been shown to be limited by the modified Welch bounds. The performance of these codes close to the bounds suggest they are close to optimal for implementation in a sonar system. To confirm these properties, an empirical test was set up to evaluate codes in a real life situation. This is explored further in Chapter 5.

Chapter 5

Evaluation - Empirical Testing (Water Tank)

A methodology for design of sonar codes has been developed for multiple level detection, and the codes have been evaluated in simulation for various properties which assume ideal conditions. These conditions are good for comparison from a theoretical perspective, allowing for a fair check against other codes and against theoretical bounds. It has been shown that they meet / exceed requirements, and have significantly better properties than existing codes that are used in sonar. From a theoretical perspective, these codes are ideal for use in sonar.

To provide further insight to the properties of these codes, it is useful to also look at codes from a practical perspective. Physical systems for testing sonar give an opportunity to observe how the codes perform in a real life scenario, providing confirmation to the accuracy of code design and providing checks for any additional considerations that may have been missed during the design phase. The valuable insight and checks of a practical system is worthwhile to set up.

For this evaluation, a water tank was set up with a transducer which had been previously characterised for frequency response across angles. With the characteristics known, sonar codes could be generated to transmit

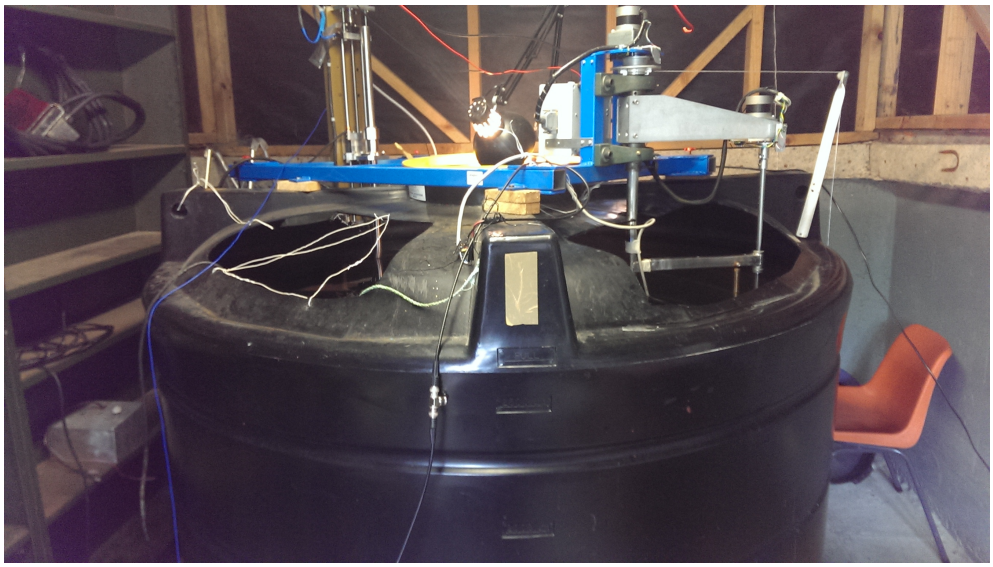


Figure 5.1: Photos of the water tank, where sonar codes were tested.

through this transducer with ideal properties. The details of this set up are elaborated in Section 5.1.

5.1 Setup

A water tank was set up with a transducer transmitting pressure waveforms, and a hydrophone receiving these waveforms at the opposite end. This water tank can be seen in Figure 5.1. The transducer was free to rotate horizontally, and was controlled via software. This allowed the relative angle between transducer and hydrophone to be set with high accuracy, which allowed for testing a variety of situations for the sonar codes. Triggering the sonar code pulses, recording the received waveform and control of the transducer angle was all handled by software, which allowed for automated testing once parameters were set.

The signal was generated using the Agilent 33522A Arbitrary Waveform Generator [23], connected through a pre amplifier in turn connected to the transducer, shown in Figure 5.2 and 5.3. The arbitrary waveform generator

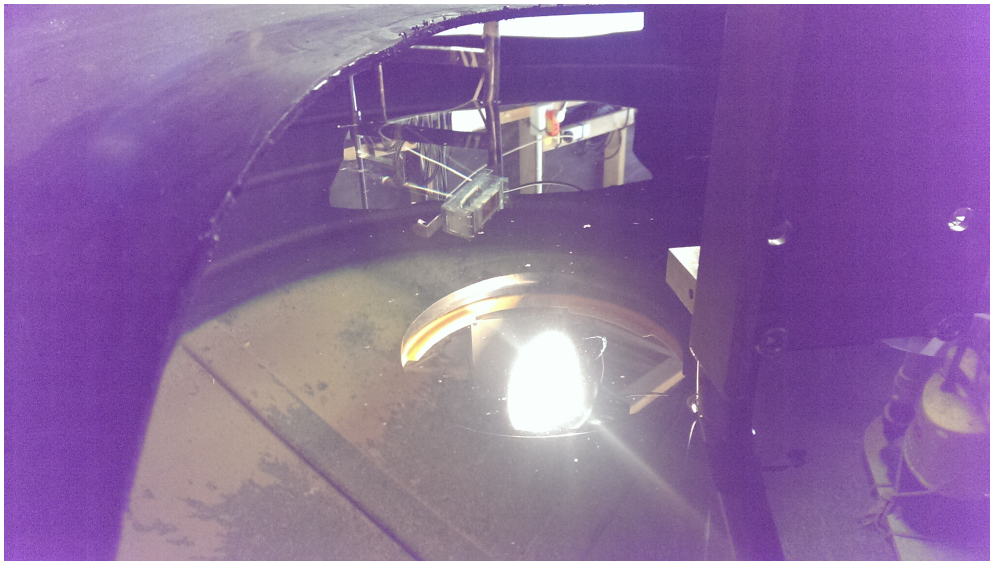


Figure 5.2: Internals of the water tank, showing the transducer transmitting codes.

Figure 5.3: Internals of the water tank, showing the hydrophone receiving pulses.

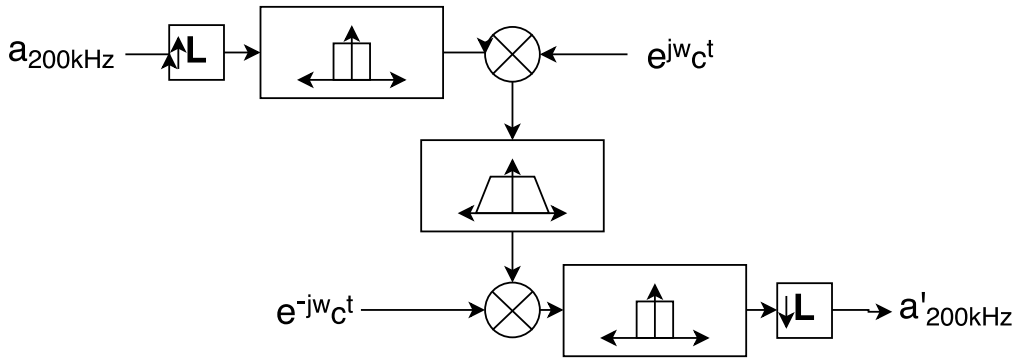


Figure 5.4: Model of the signal path for testing in the water tank system. Quadrature modulation used to shift the code up to pass band.

is able to produce a waveform of real valued samples with high accuracy, and so a real valued waveform at passband was needed, as data input to the arbitrary waveform generator. This signal was designed in MATLAB, and was implemented using an upsampled baseband signal which was then quadrature modulated to the pass band. A diagram illustrating the signal model is illustrated in Figure 5.4. This model shows the process of the signal, and is useful for identifying potential sources of error.

The signal travels from the transducer, through the water and is picked up by a hydrophone, which has a relatively flat frequency response for the frequency range being analysed. The hydrophone, shown in Figure 5.2, picks up the sonar signal from the water, and converts it to an electrical signal. This electrical signal is then analysed through an oscilloscope, which records an output in a file to be later analysed digitally. For this experiment, a matched filter was simulated in MATLAB to confirm the response. The matched filter compares the originally designed sonar codes against the received waveforms, to check if the received waveform is made up of the designed codes. If transmitted in the water tank correctly, and if the codes are designed correctly, the matched filter will have an output similar to the autocorrelation for matching codes, and low cross correlation when detecting non matched codes.

5.2 Differences and Considerations

It is worth noting the differences between this practical evaluation versus evaluating in the idealized simulated environment. One such difference is the signal channel model as seen earlier in Figure 5.4. This channel shows the signal being up converted to the pass band, passed through the channel, received then down sampled to be processed by the matched filter. This model highlights several additions to the channel, which are explored further in the following sections.

5.2.1 Noise

In simulation, noise was assumed to be ideally minimized by the matched filter and so noise was not included in simulation analysis. The simulation of the up conversion and down conversion process yields a perfect reconstruction, as there are no additional sources of error that could perturb the code. However in the water tank experiment there are several potential sources of error.

A major error source is the noise of the channel itself, in the water. Although this noise source should be theoretically minimized by the matched filter, it should still be considered when analysing the results. Averaging the measurements of several pulses mitigates the error from noise, assuming the noise has a mean of 0. This is a viable method to reduce noise as it is likely to be present in the sonar system itself, in the form of repeated pings.

5.2.2 Reflections

Another difference from simulation is the limited duration for codes in the water tank. The water tank is of limited size, and so codes have a limited duration when transmitted to be picked up by a hydrophone before reflections are picked up. The diagram in Figure 5.5 shows a few expected

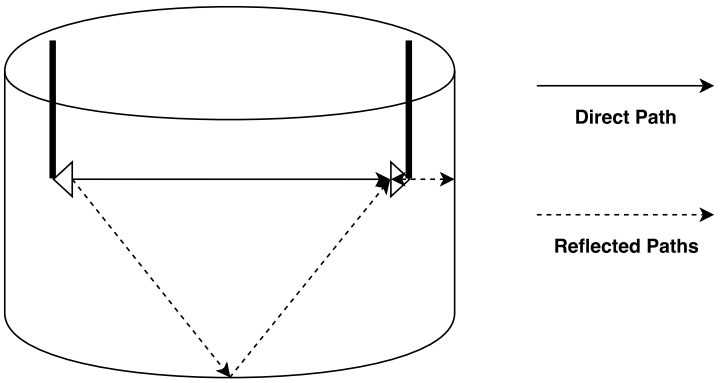


Figure 5.5: Illustration of some expected reflections of a transmitted sonar pulse in a water tank. Left is a transmitter, right a receiver. Two reflections shown, one off the right of the tank, and one off the bottom of the tank.

reflections in the water tank. As the signal travels from left to right, the direct path should be the first received signal. A reflection is expected as the signal continues past the hydrophone, reflecting off the back wall of the water tank then returning to the hydrophone. Another reflection is expected from bouncing off the bottom of the tank.

In the tank used for the experiment, the receiving hydrophone was positioned roughly 7.5 cm away from the side of the tank, and placed significantly high off the bottom. The first reflections were from the side of the tank, and for a code sampled at 200 kHz the first reflection would be picked up roughly 0.1 ms after the first pulse was picked up. This is calculated based on the speed of water, as shown in Equation (5.5). For a code sampled at 200 kHz, this corresponds to a maximum code length of 20 samples.

$$c = \frac{d}{t_{pulse}} \tag{5.1}$$

$$c = 1481\text{m/s, for water at } 20^{\circ}\text{C} \tag{5.2}$$

$$d = 2 \times 0.075\text{m} \tag{5.3}$$

$$t_{\text{pulse}} = d/c = \frac{0.15\text{m}}{1481\text{m/s}} = 0.1\text{ms} \tag{5.4}$$

$$t_{\text{pulse}} = N * t_s \tag{5.5}$$

$$= N/f_s \tag{5.6}$$

$$f_s = 200\text{kHz} \tag{5.7}$$

$$N = t_{\text{pulse}} * f_s \tag{5.8}$$

$$= 0.1\text{ms} \times 200\text{kHz} \tag{5.9}$$

$$= 20 \text{ samples} \tag{5.10}$$

From these measurements we can calculate expected time intervals that the hydrophone can receive a signal, before the next reflection begins to interfere. To measure codes ideally, in other words without interference, the codes must be limited to these time intervals. Compared to a simulated signal, which has no limitations with reflections or overlapping signals, the space of codes to test are significantly limited.

5.2.3 Equipment

A typical sonar system transmits using a transducer, then receives on either the same transducer, or on a hydrophone. This means the signal passes through two transfer functions, through the transmitting transducer and through the receiving hydrophone. The water tank uses a single transducer, then receives on a hydrophone that has a flat frequency response over the frequency range. Effectively, the signal passes through only one transfer function, highlighting a difference between the test and a real life sonar scenario. Transmitting over a single transfer function allows for more

accurate characterization of the response of the transducer, important for optimizing the codes for the transducer, and for testing the accuracy of a system.

Another point to note is that the codes are optimized per transducer, so the codes used in the tank test are unique to the transducer. The codes tested are for the transducer introduced earlier, in Figure 3.1. The reduced gain at wide angles of $\pm 60, \pm 70$ degrees is beneficial for minimizing cross correlation, as these regions are regions which will interfere, and will have a reduced gain relative to the main beam axis along the 0 angle.

5.3 Results from testing

The limitations in the water tank place limits on what can be tested for sonar codes. The most significant of these limitations is the code length. Codes longer than 20 samples will be received with interference from reflections, so to minimize this interference a relatively short code length must be used. This limitation prevents the testing of sonar codes for implementation requirements, however tests can still be done to compare against simulations. This checks for consistency, and allows for observation of how noise may perturb the filter response of the sonar codes.

A set of codes were generated for length 100 samples using matched filter optimization, then upsampled to roughly 1 MHz. The short code length provided spacing to minimize code overlap, and the code was oversampled sufficiently such that when the waveform was band shifted, the waveform could be clearly represented. The waveform was then modulated using quadrature phase modulation, as a simple method to generate and accurately modulate the waveform digitally. This waveform was transmitted through the system, and received on the oscilloscope to be analyzed digitally offline. The output was demodulated using quadrature demodulation, and filtered to obtain the base band response. This base band signal was then passed through a matched filter, matching against the designed code

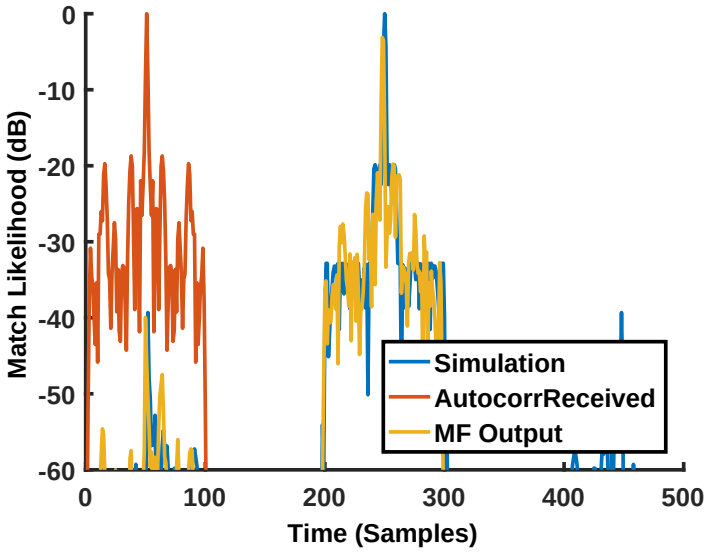


Figure 5.6: Water tank experiment 1 result, Code A (Not aligned). Note peaks in side lobes

from the original optimization.

The output of the matched filter is shown in Figure 5.6 for code A, and in Figure 5.7 for code B. In both cases there is a match in the matched filter, and a significant main peak detected, however there are also significant peaks in the side lobes, not matching the design of the original codes. These side lobe peaks may be incorrectly identified as lower power reflections if the threshold is set according to the original design, and deviate significantly from the design. This is not ideal.

To determine the cause of this deviation from design, it is useful to look at the channel model from Figure 5.4. Here there are several processes at work, and parts that have already been compensated for. Particularly noteworthy are the upsampling and downsampling processes.

Assuming perfect upsampling and downsampling, it is expected that a signal that was upsampled then downsampled would be perfectly reconstructed, and mirror the original signal. However it should be noted that the experiment deals with a real signal, which is bound in time limiting the

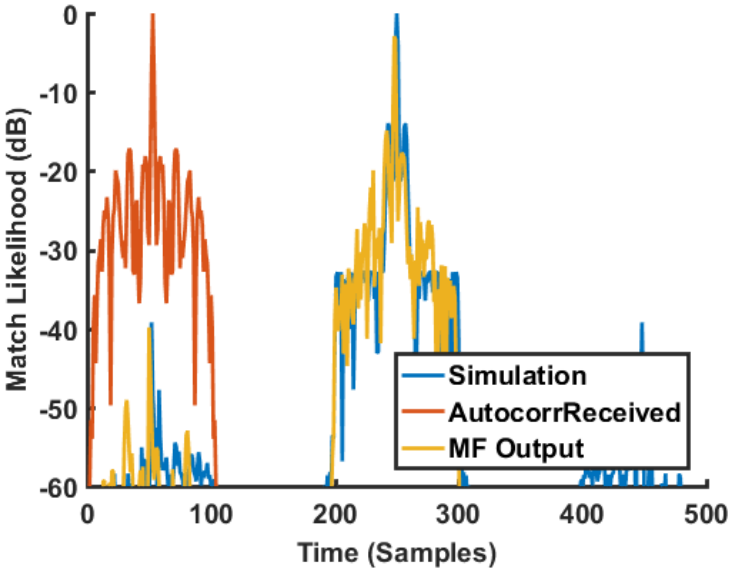


Figure 5.7: Water tank experiment 1 result, Code B (Not aligned). Note peaks in side lobes

length of samples. Another limitation is the code itself, which is defined for a limited time window and has samples for all points in the window. Limited windows are not represented in perfect upsampling and downsampling processes, and if unaccounted for lead to distortion at the edge samples. This can be accounted for by zero padding the edges of the code, as well as allowing for preceeding and trailing zeros when recording the received signal. This reduces the windowing effect in sampling, reducing distortion caused by windowing the sequence and receiving signal.

Another limitation in the channel is filtering the upsampled signal and the downsampled signal, during modulation. Modulation should ideally shift a waveform up to a frequency band, however some modulation schemes have not ideal, leaking signals outside the transmit frequency band [37]. This is typically handled by filtering the band of the signal, after the signal is shifted to the pass band. Ideally this filtering would be perfect, to only pass the signal at the pass band without any change to the signal. Offline filtering can approach this ideal with little distortion to the signal,

but in real world usage filtering is typically causal and of finite impulse response, to ensure stability and reduce complexity. For this experiment the MATLAB command 'filtfilt' was used, which applies a filter then reverses the output and passes it again to prevent any phase distortion. This 'filtfilt' command, in combination with a relatively high order butterworth filter, was used to filter the signal for the pass band. Filtering using this zero-phase distortion filter should have little effect on the output filter response, and is unlikely to cause any changes from the expected simulated response.

Noise is another characteristic that may have a significant effect on the output matched filter. Noise is present in any real world system, and can enter at every stage of signal transfer. In particular, the cables running from the signal generator to the transducer, and from the hydrophone to the oscilloscope were long due to space limitations, and could potentially couple significant noise into the system from nearby electrical sources, or even to each other. Averaging the result should significantly reduce any white noise, and overall is considered an inherent part of the system. This characteristic is significant, but not significant enough to cause specific points to peak as we saw in the first set of results. Random noise would be expected to apply equally to the matched filter response, without any preference to particular offsets.

The transducer response is a significant part of the channel, and has been compensated for in the matched filter optimization. The transducer response was measured by applying a single cycle pulsed sine wave through the input, followed by a flat zero signal of equal length. This pulse provides a wide frequency response near the frequency band, and so allows for an input of various frequencies with a signal pulse. The output can then be measured, and compared to the input, to form a frequency response. This can be converted into a transfer function, which can then be optimized using the processes outlined earlier. The accuracy of the transducer's frequency response is dependant on the accuracy of sampling the received signal. A signal that is oversampled significantly has high resolution in

the frequency domain, and so can resolve more detailed characteristics in the frequency domain. This leads to a more accurate impulse response. From empirical measurements it was assumed the transducer response was accurate enough for testing, and further measurements confirmed the frequency response was accurate.

From these considerations it was noted the upsampling and downsampling process would have the biggest impact on results, and so zero padding was introduced to compensate for distortion at the beginning and end of the sequence. This was implemented in the experiment, testing the code by taking the optimization output, adding preceding and trailing zeros, upsampling then modulating and filtering. The signal was generated and passed through the water tank system, then received on the oscilloscope. The output was recorded with padding at the beginning and end of the received waveform, and the signal demodulated and downsampled. The base band sequence was finally passed through a matched filter based on the original optimization output, and is shown in Figures 5.8 and 5.9 for codes A and B respectively.

These figures show a significant decrease of peaks in the sidelobes, and also show the signal is matched closer to 0 dB in the main peak. Characteristics match that of the simulation, showing spikes and shallows in the empirical filter plot as with the simulated filter, which confirms the accuracy of the model and design. There are small increases in side lobe characteristics compared to the simulated characteristics from code design, however these can be attributed to perturbations from noise. Overall the codes provide significant improvement in matched filter characteristics. It should be noted that with the additional zero padding that the code lengths are increased, and so affect the requirements. A longer code length means that for a fixed ping rate, the number of samples available for design are reduced. This additional consideration is needed for all future designs.

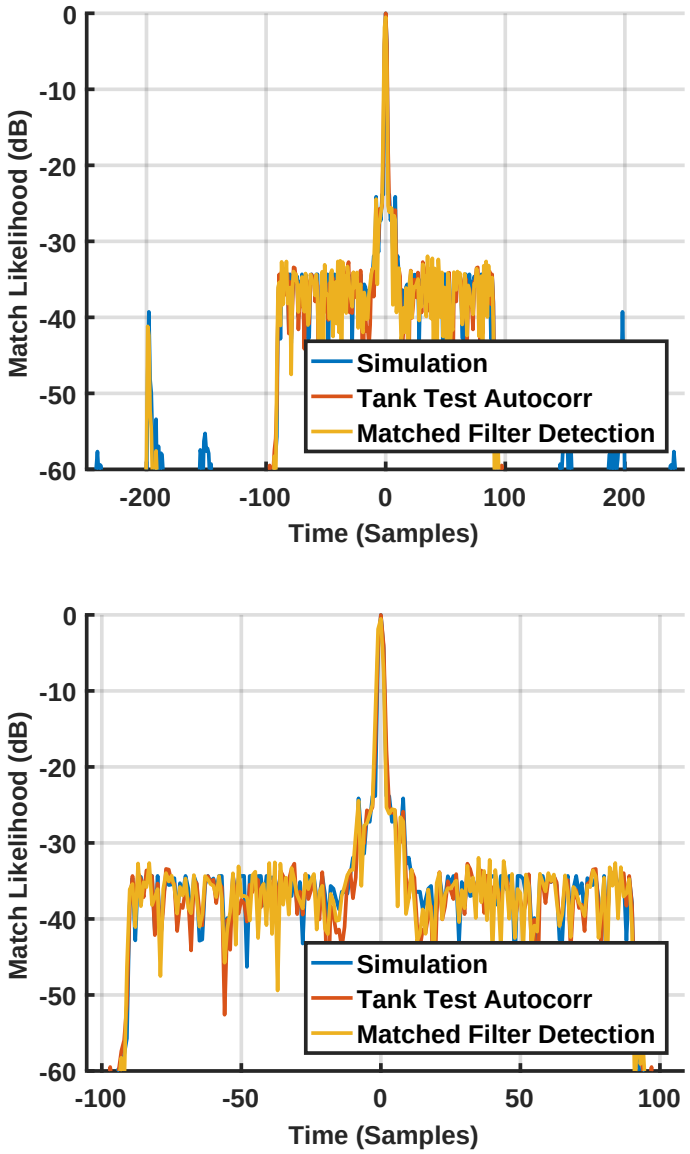


Figure 5.8: Water tank experiment 2 result, Code A (Aligned). Note significant reduction in side lobes matching design, and similar characteristics.

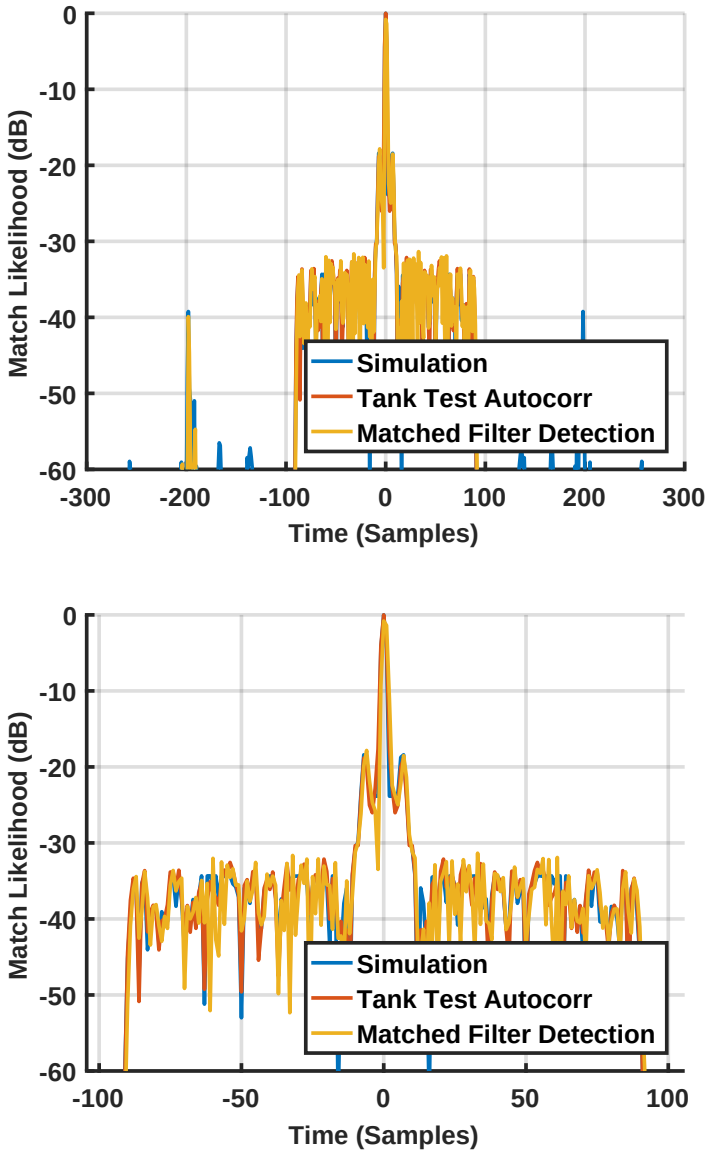


Figure 5.9: Water tank experiment 2 result, Code B (Aligned). Note significant reduction in side lobes matching design, and similar characteristics.

Chapter 6

Conclusion

Advanced sonar systems can potentially offer greater precision and resolving power than existing systems. A key component of the possible improvement is the design of the transmitted signal. To permit the simultaneous use of multiple transmitters, multiple signals having low cross correlation must be designed.

A set of codes have been developed and optimized for sonar detection. To meet object resolvability requirements, the codes were designed using multiple threshold criteria in their correlations. This is a novel approach to detection. These codes were produced using numerical optimization techniques, another novel approach for the design of both the codes and matched filters simultaneously. Bounds were also developed, extending Welch bounds to include the new class of sequences for multi-level detection.

A digital transmitter was to be used in the sonar system, and so a method was developed to map the optimal match filtered sequence onto this digital transmitter. The digital transmitter, a design imposed on the project prior to my work, was a non-linear system that changes the length and frequency of cycles to meet phase and amplitude properties at sample points. This added complications to design, as the variable length cycles made some sample points impossible to map. Additionally the integrating

effect of phase when changing a cycle length affected the efficiency of optimization. A sliding window optimization was used to map the match filtered sequence onto the digital transmitter, to mitigate the integrating effect on phase as well as to skip any points of fixed or oscillating error. The mapping was confirmed to be accurate by observing errors in phase and amplitude, as well as observing a simulated match filtered output from a digitally transmitted sequence.

Small errors were identified in the digital transmitter sequences, and so a mis-match filter was implemented to mitigate the effect of these errors. The use of a mis-match filter vastly simplified the problem, which became a strictly convex problem. This problem was characterized, and input into convex optimization tools to produce an optimal solution for filter coefficients. The output mis-matched filter was observed to correct imperfections from the digital transmitter, producing ideal filter outputs for resolution and detection in sonar.

An empirical test was conducted to confirm the accuracy of sequence design, using a water tank. The codes were transmitted from one side of the tank to the other, and processed digitally through a matched filter. The process was used to confirm the accuracy of upscaling a sequence, and that transducer characteristics were accurately modeled and compensated for in sequence design. Limitations in the dimension of the tank limited the length of sequences, and so the tank was used as a consistency check to confirm codes worked as designed. The codes were designed for side lobes of about -30 dB, and empirical results confirmed these same results of -30 dB. Furthermore the characteristics of the matched filter outputs were very similar, showing spikes and shallows in the same places as in simulation. This suggests the simulation modeled the empirical experiment conditions accurately, confirming consistency between simulation and in physical experiments. An important result from the tests highlighted that it is important to zero pad the sequence before upscaling, to prevent distortion at the edge of the sequences.

The overall design for sonar codes uses a matched filter optimizer to produce ideal codes, a digital transmitter mapping to map the codes to a usable system, and a mis-matched filter optimization to compensate for limitations in the digital transmitter. The design was confirmed to be accurate for several simulations at each stage, and with empirical experiments. Requirements in resolution and pulse shaping were met, and input parameters provide further options for producing further codes.

Appendix A

A.1 Source Files

A.1.1 Script to call optimization

```
%% Script to Call optimizer
% Script to parameterize and call optimizer, and to periodically display / save results.
% Rajiv Pratap


clear all, a_1 = NaN; b_1 = NaN;
steps = 1e4;

N=1000;
% a_1=randn(N,1)+1j*randn(N,1); a_1=a_1./norm(a_1);
% b_1=randn(N,1)+1j*randn(N,1); b_1=b_1./norm(b_1);

error_margin = 0.01;
ramp_fun = [sin(2*pi/4 *(1:300)/300), cos(2*pi/2*(1:700)/700)*.5+.5, zeros(1,200)].^2.';

k=0;
prevEE=1;
for k=1:Inf
  k
  [v,v2,EE] = optimizer('figure',100,...
     'N',N ...
```

```
    ,'l', [1 1e1 10^(-(40-3)/20)] ...
    ,'ramp', ramp_fun ...
    ,'fixedlobe',65 ...
    ,'a',a_1,'b',b_1 ...
    ,'fSample',50e3 ...
    ,'mu',5e3 ...
    ,'steps',steps);

 if (20*log10(EE) - 20*log10(prevEE) > -error_margin)
    break
 end

 N = length(v)/4;
 M = 50;
 N2 = N+M-1;
 lobe = 65;

 a_1 = v(1:N)+1j*v(N+1:2*N);
 b_1 = v(2*N+1:3*N)+1j*v(3*N+1:4*N);


 % Plotting Correlation properties
 h1 = sfigure(1);
 h2 = sfigure(2);

 drawnow;

 % Save plots for viewing afterwards
 saveas(h1,fullfile('Figures009',sprintf('CorrProperties_phase')),'fig');
 saveas(h1,fullfile('Figures009',sprintf('CorrProperties_phase')),'png');

 saveas(h2,fullfile('Figures009',sprintf('MagProperties_phase')),'fig');
 saveas(h2,fullfile('Figures009',sprintf('MagProperties_phase')),'png');
 save(fullfile('Figures009','v_set'),'v','v2');

 prevEE=EE;
end
```

A.1.2 optimizer.m

```
function [v,v2, EE, output_args ] = optimizer( varargin )
%OPTIMIZER Optimizes codes for sonar application.
% Rajiv Pratap

%% Initial and default variables

N = 201;
Nt = 201;

fClock=204.8E6; R1 = 410;
fCentre = fClock/2/R1;%(min(freq)+max(freq))/2;
fSample = 100e3;

% Isolate N initially, so it can initialise other variables.
for i=1:2:length(varargin)
  switch upper(varargin{i})

      case 'N'
        N = varargin{i+1};
      case 'FSAMPLE'
        fSample = varargin{i+1};
  end
end

n = [1:N]';
k = 2;
N2 = N+Nt-1;

steps = Inf;
figure_toggle = 1;

lobe = floor(.3*N); %210;
stepdown_width = 5;

ramp_fun = .9*(tukeywin(N2,.5)+.1);
```

```
ramp_fun(floor(N2/2):end) = (.9*(N2:-1:floor(N2/2))/N2+.1);

% Scaling vector to scale each gradient for optimizer weighting.
% Ordered by [G_c G_p2a G_rc];
l = [1 1e-12 10^(-(40-3)/20)];                                          40
l = [0 1e-1 0];

[~,~,freq] = impulsetransducer1(Nt,[0 0],[fSample fCentre]);
[txrx_nadir,TxRxC1_nadir_raw] = impulsetransducer1(Nt,[0 0],[fSample fCentre]);
txrx_nadir = txrx_nadir./norm(TxRxC1_nadir_raw,1);
[txrx2_nadir,TxRxC2_nadir_raw] = impulsetransducer1(Nt,[0 0],[fSample fCentre]);
txrx2_nadir = txrx2_nadir./norm(TxRxC2_nadir_raw,1);

angle_range = [60:5:70];
M_angle = length(angle_range);                                          50
for i=1:length(angle_range)
   [txrx_angles(:,:,i)] = impulsetransducer1(Nt,[angle_range(i) 0],[fSample fCentre])./norm(T
   [txrx2_angles(:,:,i)] = impulsetransducer1(Nt,[-angle_range(i) 0],[fSample fCentre])./norm
end
Nt = length(txrx_nadir); N2 = N+Nt-1;

% % Could initialise with Zadoff-Chu sequence instead.
u1 = 23; u2 = 29;
a = exp(-1j*(pi*u1*n.*(n+1))/N);
b = exp(-1j*(pi*u2*n.*(n+1))/N);                                        60

a = a/norm(conv(txrx2_nadir,a));
b = b/norm(conv(txrx_nadir,b));

mu=50;

%% Switch to implement function options

 for i=1:2:length(varargin) %iterate length varargin in case other arguments exist for functior
  switch upper(varargin{i})                                             70

     case 'N'
       %N = varargin{i+1};
```

```
    case 'FSAMPLE'

    case 'MU'
      mu = varargin{i+1};
    case 'P2A'
      l = varargin{i+1};
    case 'LOBE'                                                          80
      % Lobe as a fraction of N
      lobe = floor(N*varargin{i+1});
    case 'FIXEDLOBE'
      lobe = floor(varargin{i+1});
    case 'TAPER'
      % Distance from centre autocorrelation to design Rayleigh stepdown point
      stepdown_width = floor(varargin{i+1});
    case 'RAMP'
      ramp_fun = varargin{i+1};
      ramp_fun = resample(ramp_fun,N2,length(ramp_fun));                 90
    case 'L'
      l = varargin{i+1};
    case 'A'
      if ~isnan(varargin{i+1})
        a = varargin{i+1};
      end
    case 'B'
      if ~isnan(varargin{i+1})
        b = varargin{i+1};
      end                                                               100
    case 'FIGURE'
      figure_toggle = varargin{i+1};
    case 'STEPS'
      steps = varargin{i+1};
    otherwise
      error(['Unexpected option:' varargin{i}])
  end
 end

%% More options, currently static                                      110
```

```
v = [real(a);imag(a);real(b);imag(b)];

% Set up H matrix for easy calculation of gradient
% Sets each row to shift relative to the previous row, creating a band
% matrix.
txrx_op_angles = repmat([(conj((txrx_angles)));zeros((N2)-length(txrx_angles),1,M_angle)],1,N
txrx_op_angles = permute(txrx_op_angles,[2 1 3]);
[m, n, o] = size(txrx_op_angles);                                                    120
D = 0:n-1;
B = zeros(m, n, o);
for i = (1 : m)
   B(i,:,:) = [txrx_op_angles(i,(n - D(i) + 1 : n),:), txrx_op_angles(i,(1 : n - D(i) ),:)];
end
txrx_op_angles = B;

txrx_op2_angles = repmat([(conj((txrx2_angles)));zeros((N2)-length(txrx2_angles),1,M_angle)],
txrx_op2_angles = permute(txrx_op2_angles,[2 1 3]);
[m, n, o] = size(txrx_op2_angles);                                                   130
D = 0:n-1;
B = zeros(m, n, o);
for i = (1 : m)
   B(i,:,:) = [txrx_op2_angles(i,(n - D(i) + 1 : n),:), txrx_op2_angles(i,(1 : n - D(i) ),:)];
end
txrx_op2_angles = B;

% Set up txrx_op matrices for nadir calculations
txrx_op_nadir = repmat([(conj((txrx_nadir)));zeros((N2)-length(txrx_nadir),1)],1,N)';
[m, n] = size(txrx_op_nadir);                                                        140
D = 0:n-1;
B = zeros(m, n);
for i = (1 : m)
   B(i,:) = [txrx_op_nadir(i,(n - D(i) + 1 : n)), txrx_op_nadir(i,(1 : n - D(i) ))];
end
txrx_op_nadir = B;

txrx_op2_nadir = repmat([(conj((txrx2_nadir)));zeros((N2)-length(txrx2_nadir),1)],1,N)';
[m, n] = size(txrx_op2_nadir);
```

```
D = 0:n-1;
B = zeros(m, n);
for i = (1 : m)
   B(i,:) = [txrx_op2_nadir(i,(n - D(i) + 1 : n)), txrx_op2_nadir(i,(1 : n - D(i) ))];
end
txrx_op2_nadir = B;
clear('B');

%% Loop for error / gradient calculation

prevEE = 0;
for i=1:steps
  % Calculating Correlation gradient
  a_1 = v(1:N) + 1j*v(N+(1:N));
  b_1 = v(2*N+(1:N)) + 1j*v(3*N+(1:N));

  a_2_set = zeros(N2,M_angle); b_2_angle = zeros(N2,M_angle);
  for k=1:M_angle
     a_2_set(:,k) = conv(txrx2_angles(:,:,k),a_1);
     b_2_set(:,k) = conv(txrx_angles(:,:,k),b_1);
  end
%   a_2 = conv(txrx2,a_1); % results in 201x1 matrix
%   b_2 = conv(txrx,b_1);

  a_2_nadir = conv(txrx2_nadir,a_1); % results in 201x1 matrix
  b_2_nadir = conv(txrx_nadir,b_1);

  max_error = 0;
  max_error_position_c = 0;
  %min_error = realmax; min_error_position = 0;

  % Correlation error calculation

  % Looking at each angle for the transducer, and picking the corr
  % characteristics resulting in the largest error.
  Eab_set = zeros(N2*2-1,1,M_angle);
  for k=1:M_angle
     Eab_set(:,:,k) = xcorr(a_2_set(:,k),b_2_set(:,k));
```

```
  end
  Eab_max = max(Eab_set);
  [~, Eab_pos] = max(Eab_max,[],3);
  Eab = Eab_set(:,:,Eab_pos);
  a_2_angle = conv(txrx2_angles(:,:,Eab_pos),a_1);
  b_2_angle = conv(txrx_angles(:,:,Eab_pos),b_1);
  txrx_op = txrx_op_angles(:,:,Eab_pos);
  txrx_op2 = txrx_op2_angles(:,:,Eab_pos);

%   txrx_op = fft(txrx_op).*fft(chebwin(Nt).');
%    txrx_op2 = fft(txrx_op2).*fft(chebwin(Nt).');

  %Eab = (xcorr(a_2,b_2));
  Eaa = (xcorr(a_2_nadir,a_2_nadir));
  Ebb = (xcorr(b_2_nadir,b_2_nadir));

  % Recalculate a and b for the filtered transducer function
  a_2_nadir = conv(txrx2_nadir,a_1);
  b_2_nadir = conv(txrx_nadir,b_1);

  % Set requirements for correlation
  Eaa_c = Eaa;
  Eaa_c(N2-lobe:N2-1) = zeros(size(Eaa(N2-lobe:N2-1)));
  Eaa_c(N2+1:N2+lobe) = zeros(size(Eaa(N2+1:N2+lobe)));
  Eaa_c(N2) = Eaa(N2)-1;

  Ebb_c = Ebb;
  Ebb_c(N2-lobe:N2-1) = zeros(size(Eaa(N2-lobe:N2-1)));
  Ebb_c(N2+1:N2+lobe) = zeros(size(Eaa(N2+1:N2+lobe)));
  Ebb_c(N2) = Ebb(N2)-1;

  % Set requirements for Rayleigh Criterion
  Eaa_rc = abs(Eaa); % Set abs here so abs compared with -3dB
%   Eaa_rc(N2-lobe:N2-1) = [zeros(lobe-stepdown_width,1); Eaa_rc(N2-stepdown_width:N2-1)-10^(
%   Eaa_rc(N2+1:N2+lobe) = [Eaa_rc(N2+1:N2+stepdown_width)-10^(-3/20); zeros(lobe-stepdown_wi
  Eaa_rc(N2-lobe:N2-1) = [Eaa_rc(N2-lobe:N2-stepdown_width-1); zeros(stepdown_width,1)];
  Eaa_rc(N2+1:N2+lobe) = [zeros(stepdown_width,1);Eaa_rc(N2+stepdown_width+1:N2+lobe)];
  Eaa_rc(N2) = Eaa(N2)-1;
```

```
  Ebb_rc = abs(Ebb); % Set abs here so abs compared with -3dB
%   Ebb_rc(N2-lobe:N2-1) = [zeros(lobe-stepdown_width,1); Ebb_rc(N2-stepdown_width:N2-1)-10^(-
%   Ebb_rc(N2+1:N2+lobe) = [Ebb_rc(N2+1:N2+stepdown_width)-10^(-3/20); zeros(lobe-stepdown_wid
  Ebb_rc(N2-lobe:N2-1) = [Eaa_rc(N2-lobe:N2-stepdown_width-1); zeros(stepdown_width,1)];
  Ebb_rc(N2+1:N2+lobe) = [zeros(stepdown_width,1);Eaa_rc(N2+stepdown_width+1:N2+lobe)];
  Ebb_rc(N2) = Ebb(N2)-1;


  Etotal_Vector_c = max(abs([Eab,Eaa_c,Ebb_c]),[],2);
  %Etotal_Vector_c2 = arrayfun(@(x,y,z) max([x,y,z]),Eab,Eaa_c,Ebb_c);  %
  Etotal_Vector_rc = max(abs([Eab,Eaa_rc,Ebb_rc]),[],2);
  %Etotal_Vector_rc2 = arrayfun(@(x,y,z) max([x,y,z]),Eab,Eaa_rc,Ebb_rc);  %
  [EE, max_error_position_c] = max(Etotal_Vector_c);
  [~, max_error_position_rc] = max(Etotal_Vector_rc);

  % Calculating the actual gradient at the max error positions
  offset_range = [max_error_position_c, max_error_position_rc]-N2;
  for k=1:length(offset_range)
     offset = offset_range(k);

     if offset<0
        a0 = [zeros(-offset,1); a_2_angle(1:(N2+offset))];
        b0 = [b_2_angle((1-offset):N2); zeros(-offset,1)];
        a0_nadir = [zeros(-offset,1); a_2_nadir(1:(N2+offset))];
        b0_nadir = [b_2_nadir((1-offset):N2); zeros(-offset,1)];

        h01a = [txrx_op2(:,(1-offset):N2),zeros(N,-offset)]; %h01 goes in front of eaa^*
        h02a = [zeros(N,-offset),txrx_op2(:,1:(N2+offset))]; %h02 goes in front of eaa
        h01b = [txrx_op(:,(1-offset):N2),zeros(N,-offset)]; %h01 goes in front of ebb^*
        h02b = [zeros(N,-offset),txrx_op(:,1:(N2+offset))]; %h02 goes in front of ebb
     elseif offset>0
        a0 = [a_2_angle((1+offset):N2); zeros(offset,1)];
        b0 = [zeros(offset,1); b_2_angle(1:(N2-offset))];
        a0_nadir = [a_2_nadir((1+offset):N2); zeros(offset,1)];
        b0_nadir = [zeros(offset,1); b_2_nadir(1:(N2-offset))];

        h01a = [zeros(N,offset),txrx_op2(:,1:N2-offset)]; %h01 goes in front of eaa^*
```

```
        h02a = [txrx_op2(:,1+offset:N2),zeros(N,offset)]; %h02 goes in front of eaa
        h01b = [zeros(N,offset),txrx_op(:,1:N2-offset)]; %h01 goes in front of ebb^*
        h02b = [txrx_op(:,1+offset:N2),zeros(N,offset)]; %h02 goes in front of ebb
    else
        a0 = a_2_angle;
        b0 = b_2_angle;
        a0_nadir = a_2_nadir;                                                    270
        b0_nadir = b_2_nadir;

        h01a = txrx_op2; %h01 goes in front of eaa^*
        h02a = txrx_op2; %h02 goes in front of eaa
        h01b = txrx_op; %h01 goes in front of ebb^*
        h02b = txrx_op; %h02 goes in front of ebb
    end

    Eab = sum(a_2_angle .* conj(b0));
    Eaa = sum(a_2_nadir .* conj(a0_nadir));                                      280
    Ebb = sum(b_2_nadir .* conj(b0_nadir));


    % Find max error for this offset
    if(offset == 0)
      [Etotal, Eindex] = max(abs([Eab Eaa-1 Ebb-1]));
      Eindex_rc = Eindex;
%     elseif abs(offset)>=stepdown_width && abs(offset)<=20%abs(offset)==stepdown_width || a
%       [Etotal, Eindex] = max(abs([Eab (abs(Eaa)-stepdown) (abs(Ebb)-stepdown)]));
    elseif (offset <= lobe) && (offset >= -lobe)                                 290
      [Etotal, Eindex] = max(abs([Eab 0 0]));
      if abs(offset)>= stepdown_width
        [~, Eindex_rc] = max(abs([Eab Eaa Ebb]));
      else
        Eindex_rc = 0;
      end
    else
      [Etotal, Eindex] = max(abs([Eab Eaa Ebb]));
      Eindex_rc = Eindex;
    end                                                                          300
```

```
if (k==2 && (Eindex_rc == 2 || Eindex_rc == 3)) || (k==1 && (Eindex == 2 || Eindex == 3))
if offset < 0
   h01a = [txrx_op2_nadir(:,(1-offset):N2),zeros(N,-offset)]; %h01 goes in front of eaa^*
   h02a = [zeros(N,-offset),txrx_op2_nadir(:,1:(N2+offset))]; %h02 goes in front of eaa
   h01b = [txrx_op_nadir(:,(1-offset):N2),zeros(N,-offset)]; %h01 goes in front of ebb^*
   h02b = [zeros(N,-offset),txrx_op_nadir(:,1:(N2+offset))]; %h02 goes in front of ebb
elseif offset>0
   h01a = [zeros(N,offset),txrx_op2_nadir(:,1:N2-offset)]; %h01 goes in front of eaa^*
   h02a = [txrx_op2_nadir(:,1+offset:N2),zeros(N,offset)]; %h02 goes in front of eaa
   h01b = [zeros(N,offset),txrx_op_nadir(:,1:N2-offset)]; %h01 goes in front of ebb^*
   h02b = [txrx_op_nadir(:,1+offset:N2),zeros(N,offset)]; %h02 goes in front of ebb
else
   h01a = txrx_op2_nadir; %h01 goes in front of eaa^*
   h02a = txrx_op2_nadir; %h02 goes in front of eaa
   h01b = txrx_op_nadir; %h01 goes in front of ebb^*
   h02b = txrx_op_nadir; %h02 goes in front of ebb
end
end
% Check which loop, then calculate gradient
if k==2
   if Eindex_rc == 1
    m1 = conj(h02a)*b_2_angle;
    m2 = conj(h01b)*a_2_angle;
    G_ab = [  real((Eab).*(m1));...
             imag((Eab).*(m1));...
             real(conj(Eab).*(m2));...
             imag(conj(Eab).*(m2))  ];
    G1_rc = G_ab;

   elseif Eindex_rc == 2
    if offset == 0
     m1 = conj(h02a)*a_2_nadir;
     G_aa = [real(2.*(Eaa-1).*(m1));...
            imag(2.*(Eaa-1).*(m1))
            zeros(size(b_1));...
            zeros(size(b_1)) ];

    elseif abs(offset) <= lobe
```

```
 m1 = conj(h02a)*a_2_nadir;
 m2 = conj(h01a)*a_2_nadir;
 G_aa = [real(conj(Eaa).*(m1) + (Eaa).*(m2));...
        imag(conj(Eaa).*(m1) + (Eaa).*(m2));...
        zeros(size(b_1));...
        zeros(size(b_1)) ];
else
 m1 = conj(h02a)*a_2_nadir;
 m2 = conj(h01a)*a_2_nadir;
 G_aa = [real(conj(Eaa).*(m1) + Eaa.*(m2));...
        imag(conj(Eaa).*(m1) + Eaa.*(m2));...
        zeros(size(b_1));...
        zeros(size(b_1)) ];
end
G1_rc = G_aa;

elseif Eindex_rc == 3
 if offset == 0
  m1 = conj(h01b)*b_2_nadir;
  G_bb = [zeros(size(a_1));...
        zeros(size(a_1)); ...
        real(2.*(Ebb-1).*(m1));...
        imag(2.*(Ebb-1).*(m1)) ];
 elseif abs(offset) <= lobe
  m1 = conj(h01b)*b_2_nadir;
  m2 = conj(h02b)*b_2_nadir;
  G_bb = [zeros(size(a_1));...
        zeros(size(a_1)); ...
        real(conj(Ebb).*(m1) + (Ebb).*(m2));...
        imag(conj(Ebb).*(m1) + (Ebb).*(m2))  ];
 else
  m1 = conj(h01b)*b_2_nadir;
  m2 = conj(h02b)*b_2_nadir;
  G_bb = [zeros(size(a_1));...
        zeros(size(a_1)); ...
        real(conj(Ebb).*(m1) + Ebb.*(m2));...
        imag(conj(Ebb).*(m1) + Ebb.*(m2))  ];
 end
```

```
      G1_rc = G_bb;
    else
      G1_rc = 0;                                                    380
    end
  else %(k == 1)
    if Eindex == 1
     m1 = conj(h02a)*b_2_angle;
     m2 = conj(h01b)*a_2_angle;
     G_ab = [  real((Eab).*(m1));. . .
              imag((Eab).*(m1));. . .
              real(conj(Eab).*(m2));. . .
              imag(conj(Eab).*(m2))  ];
     G1_c = G_ab;                                                   390

    elseif Eindex == 2
     if offset == 0
      m1 = conj(h02a)*a_2_nadir;
      G_aa = [real(2.*(Eaa-1).*(m1));. . .
             imag(2.*(Eaa-1).*(m1))
             zeros(size(b_1));. . .
             zeros(size(b_1)) ];
     else
      m1 = conj(h02a)*a_2_nadir;                                    400
      m2 = conj(h01a)*a_2_nadir;
      G_aa = [real(conj(Eaa).*(m1) + Eaa.*(m2));. . .
             imag(conj(Eaa).*(m1) + Eaa.*(m2));. . .
             zeros(size(b_1));. . .
             zeros(size(b_1)) ];
     end
     G1_c = G_aa;

    elseif Eindex == 3
     if offset == 0                                                 410
      m1 = conj(h01b)*b_2_nadir;
      G_bb = [zeros(size(a_1));. . .
             zeros(size(a_1)); . . .
             real(2.*(Ebb-1).*(m1));. . .
             imag(2.*(Ebb-1).*(m1)) ];
```

```
        else
         m1 = conj(h01b)*b_2_nadir;
         m2 = conj(h02b)*b_2_nadir;
         G_bb = [zeros(size(a_1));...
                zeros(size(a_1)); ...
                real(conj(Ebb).*(m1) + Ebb.*(m2));...
                imag(conj(Ebb).*(m1) + Ebb.*(m2))  ];
        end
        G1_c = G_bb;
       else
        G1_c = 0;
       end
    end
 end
 clear h0*;

 % P2A Gradient Calculation
 ramp_fun = ramp_fun ./ (norm(ramp_fun,1)./N2);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 G1_p2a_a = zeros(N,1);
 G1_p2a_b = zeros(N,1);

 a_2 = a_2_nadir;
 Ha=txrx_op2_nadir.';

 b_2 = b_2_nadir;
 Hb=txrx_op_nadir.'; % Redefinitions to make code shorter

%  b_2_r = repmat(b_2,1,N);
%  b_2_r2 = repmat(b_2,1,N2);


%  % For ratio error adjustment (error range dependant on ramp function)
%  e_p2a = conj(a_2).*a_2 ./((a_2'*a_2)/N2*ramp_fun_short)-1;
%   [~,e_p2a_ind] = max(e_p2a.*conj(e_p2a));
%   e_p2a_val = e_p2a(e_p2a_ind);
%   ndx = e_p2a_ind;
%
```

```
%   de_dx_p2a = (a_2'*a_2/N2*ramp_fun(ndx)* Ha(ndx,:)'*a_2(ndx) - ...
%        Ha'*a_2/N2*ramp_fun(ndx)*a_2(ndx)'*a_2(ndx)) ...
%     ./ (a_2'*a_2/N2*ramp_fun(ndx)).^2;
%  G1_p2a_a = 2*e_p2a_val * de_dx_p2a;
%
%  % Full gradient set, to do more than max adjustment
%
%  de_dx_p2a_2 = ((a_2'*a_2)/N2*ramp_fun_r.'.*Ha(:,:)'.*a_2_r.' - ...
%       (Ha'*a_2_r2)/N2.*ramp_fun_r.'.*(a_2_r'.*a_2_r.')) ...
%     ./ (a_2'*a_2/N2*ramp_fun_r.').^2;
%  G1_p2a_a_set = 2* bsxfun(@times,e_p2a.',de_dx_p2a_2);
%   G1_p2a_a = (sum(G1_p2a_a_set,2))/N2;%de_dx_p2a_2(:,ndx);%

  % For difference error adjustment (const error range)
  e_p2a = conj(a_2).*a_2 - a_2'*a_2/N2*ramp_fun;
  [~,ndx] = max(e_p2a);
  %dedx_p2a = Ha'.*a_2_r.' - (Ha'*a_2_r2.')/N2.*ramp_fun_r.';
  dedx_p2a = bsxfun(@times,Ha',a_2.') - bsxfun(@times,Ha'*a_2,ramp_fun.')/N2;
%   clear Ha;
  G1_p2a_a_set = 2*bsxfun(@times,e_p2a.',dedx_p2a);
%  clear dedx_p2a;
  G1_p2a_a = (sum(G1_p2a_a_set,2))/N2^.5;% Reduce p-1 norm error
  %G1_p2a_a = G1_p2a_a_set(:,ndx);% Reduce positive peak error

  e_p2a_b = conj(b_2).*b_2 - b_2'*b_2/N2*ramp_fun;
  [~,ndx] = max(e_p2a_b);
  %dedx_p2a_b = Hb'.*b_2_r.' - (Hb'*b_2_r2.')/N2.*ramp_fun_r.';
  dedx_p2a_b = bsxfun(@times,Hb',b_2.') - bsxfun(@times,Hb'*b_2,ramp_fun.')/N2;
%  clear Hb;
  G1_p2a_b_set = 2*bsxfun(@times,e_p2a_b.',dedx_p2a_b);
%   clear dedx_p2a_b;
  G1_p2a_b = (sum(G1_p2a_b_set,2))/N2^.5; % Reduce p-1 norm error
  %G1_p2a_b = G1_p2a_b_set(:,ndx);% Reduce positive peak error

%   dxdx_p2a_b = bsxfun(@times,Hb',b_2.') - bsxfun(@times,bsxfun(@mtimes,Hb',b_2),ramp_fun.')/

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  G1_p2a = [real(G1_p2a_a);imag(G1_p2a_a);real(G1_p2a_b);imag(G1_p2a_b)];
```

```
  % Find Net Error Gradient
  G1 = l(1)*G1_c + l(2)*G1_p2a + l(3)*G1_rc;

  v2 = [real(a_2_nadir);imag(a_2_nadir);real(b_2_nadir);imag(b_2_nadir)];
  v = v - mu.*G1;

%   if EE > prevEE
%       mu=mu/10;                                                          500
%   else
%       mu=mu*1.5;
%   end
%   prevEE = EE;
%
  if (mod(i,figure_toggle ) == 0)
      template = zeros(2*N2-1,1); template(N2) = 1;
      aa = xcorr(a_2_nadir,a_2_nadir);
      bb = xcorr(b_2_nadir,b_2_nadir);
      for k=1:M_angle                                                    510
        a_2_plot = conv(txrx2_angles(:,:,k),a_1);
        b_2_plot = conv(txrx_angles(:,:,k),b_1);
        ab(:,k) = xcorr(a_2_plot,b_2_plot);
      end

      ee = (abs([aa-template;bb-template;reshape(ab,[M_angle*length(aa),1])]));
      %ee((N-lobe):(N+lobe)) = zeros(2*lobe+1,1);
      ee((0*N2+N2-lobe-1):(0*N2+N2+lobe+1)) = zeros(2*lobe+3,1);
      ee((2*N2+N2-lobe-1):(2*N2+N2+lobe+1)) = zeros(2*lobe+3,1);
      %ee((4*N2-2+N2-lobe):(4*N2-2+N2+lobe)) = zeros(2*lobe+1,1);          520
      EE = max(ee);

%       if EE > prevEE
%           mu=mu/8;
%       else
%           mu=mu*2;
%       end
%       prevEE = EE;
```

```
% if (mod(i,figure_toggle ) == 0)                                        530
    %p2a_a = max(abs(a_1))./(sum(abs(a_1))/N);
    %p2a_b = max(abs(b_1))./(sum(abs(b_1))/N);
    %evaluate ramping function
    p2a_a = max((conj(a_2_angle).*a_2_angle)./(a_2_angle'*a_2_angle/N2*ramp_fun));%max(abs(a_1).
    p2a_b = max((conj(b_2_angle).*b_2_angle)./(b_2_angle'*b_2_angle/N2*ramp_fun));%max(abs(b

%       if(20*log10(EE) > 0)
%           pause;
%       end
                                                                        540
    % Plotting Correlation properties


        h1=sfigure(1);
          x = -N2+1:N2-1;
          plot(x,20*log10(abs([aa bb ab])));%ab])));
          title(sprintf('error:%1.3f dB, p2a:%1.3f, step:%1.f',20*log10(EE),max(p2a_a,p2a_b),i));
          %zoom on;
          grid on;
          ylim([-60 0]);                                                550

        h2=sfigure(2); plot(abs([a_2_nadir b_2_nadir ramp_fun.^.5.*(a_2_nadir'*a_2_nadir/N2)^.5 ramp
          title('Magnitude of codes'),
          legend('|a|','|b|','Location','NorthEast')

%           h3=sfigure(3); plot(([G1 l(1)*G1_c l(2)*G1_p2a])),
%             title('Error Gradients and Gradient components'),
%             legend('G_{total}','G_{correlation}','G_{peak-to-average}','Location','NorthEast')

%           h4=sfigure(4); plot([(phase(a_1)) (phase(b_1))]),             560
%             title('Phase of signals')
%             legend('a','b','Location','NorthEast')

    %       h=figure(5); plot(real([a_1 b_1])),
    %         title('Real component of signals')
    %         legend('a','b','Location','Best')
```

```
%              h=sfigure(6); plot((e_p2a)),
%                title('p2a Error')
              h=sfigure(7); plot(linspace(min(freq),max(freq),length(a_2)),abs(fftshift([fft(a_2) fft(b_2)])))
%                h=sfigure(7); plot(linspace(min(window_range),max(window_range),length(a_2)),abs(fftshift
               title('FFT of signals')

          drawnow;   %pause(0.5);

  end


%    if figure_toggle == 1
%      close([h1 h2 h3 h4]);
%    end
end

end
```

A.1.3 Baseband to Digital Transmitter Mapping Optimization

```
%% BBtoDTx
% Sliding window optimization to map a baseband sequence on to a digital transmitter.
%  Rajiv Pratap

clear all

% Using angle to match phase
%% Set up variables

% Random signal
% N = 100;
% bb = randn(N,1) + 1j*randn(N,1);
% bb = rand(N,1).*exp(1j* 2*(2*rand(N,1)-1));
% bb = rand(N,1).* exp(1j* 2*cumsum(2*rand(N,1)-1.5));
%bb = bb./norm(abs(bb));

% or, sonar designed sequence
load('v_set');
% load('/u/students/pratapraji/Masters/matlab/rajiv/StrictMiniMaxOptimization/Figures003/v_se
N = length(v)/4;
Nt = 50;
a_1 = v(1:N)+1j*v(N+1:2*N);
b_1 = v(2*N+1:3*N)+1j*v(3*N+1:4*N);
bb = [0; a_1];


% Tx spec
fClock = 204.8E6;
num_hwp_bits = 12;
num_amp_bits = 8;
TransmitterParams = SpecifyTransmitterParams( fClock, num_hwp_bits, num_amp_bits);

TxampMAX = 2^TransmitterParams.num_amp_bits;
```

```
% Rx spec
fSample = 100E3;
% sonar centre frequency
R1=410;
fCentre = fClock/2/R1;%200E3;
ReceiverParams = SpecifyReceiverParams( fSample, fCentre);

%% Loop to fit HPW to each phase value

AMP = zeros(1,5*N);%255*ones(1,2*N);

HPW_zero = fClock/2 /fCentre /2;
HPW2_zero = fClock/2 /fSample /2;
HPW = HPW_zero*ones(1,5*N);
HPWtotal = 0;
HPW0sum = cumsum([HPW2_zero*ones(1,5*N)]);

flipbit = zeros(1,2*N);

% Resample, find the upsampled phase, then decimate the phase to get better
% unwrapping
phase_bb = phase(bb);
phase_bb = angle(bb);

% Begin optimizing
window = 3;
walkFun = cumsum(upsample(diff(1:length(bb)+1),100*window));
walkFun = walkFun+mod(0:length(walkFun)-1,window) -window+2;
walkFun = min(walkFun,length(bb));
walkFun = max(1,walkFun);

k=1;
for i1 = 1:length(bb)
    deltaHPW = ones(1,window);
    mu = .1;
    i3=1;
```

```
    while i3 <=100 && max(abs(deltaHPW))>= 1
        i3=i3+1;
        for i2 = 1:window
            i = min(max(i1+i2-1,1),length(bb));
            HPWsum = cumsum([0, HPW]);
%               HPW_phasediff = mod(HPWsum(i)-160,320)+160;

            % While using valid k, and
            % while the current HPW cumsum is less than the RX timing cumsum   80
            % % Use a 50% offset in RX timing to ensure HPW is sampled at a
            % % reasonable point within the waveform, so that changing length
            % % significantly changes phase
            k=1;
            while k<length(HPWsum)-1 && HPWsum(k+1) < HPW0sum(i)-HPW_zero/4
                k=k+1;
            end
            k2=find(HPWsum < HPW0sum(i)-HPW_zero/4 & HPWsum > HPW0sum(max(i-1,1)));

            m(i) = HPW0sum(i) - HPWsum(k);                                    90


            % Recalculate values in the plot
            bb_DTx = DTxtoBB(HPW(1:length(AMP)),AMP,TransmitterParams, ReceiverParams);%, flipbit
            phase_bbDTx = phase(bb_DTx);
            phase_bbDTx = angle(bb_DTx);


            mu = .1;
            deltaHPW(i2) = -(phase_bb(i)-phase_bbDTx(i))/(2*pi+(phase_bb(i)-phase_bbDTx(i)))*(m(i)

            HPW(k2) = HPW(k2) + mu*deltaHPW(i2);

            if HPW(k) > HPW2_zero*2 || HPW(k) < HPW2_zero/4
                HPW(k) = HPW2_zero - HPW(k);
                HPW(k) = min(max(HPW(k),HPW2_zero/4),HPW2_zero*2);
            end

            % Amplitude adjustment - choose k without halfpulse offset
```

```
            k=1;                                                                    110
            while k<length(HPWsum)-1 && HPWsum(k+1) <= HPW0sum(i)
                k=k+1;
            end
            AMP(k2) = asin(abs(bb(i))./max(abs(bb))) ./ (pi/2/TxampMAX);

            if mod(i3,30) == 0 && i2 == 1
                % Recalculate values in the plot
                [bb_DTx,t ]= DTxtoBB(HPW(1:length(AMP)),AMP,TransmitterParams, ReceiverParams);
                phase_bbDTx = angle(bb_DTx);
                                                                                    120
                sfig(12); clf; hold on
%                   plot(([phase_bb]))
%                   plot(([phase_bbDTx]))
                plot(wrapToPi([phase(bb)]))
                plot(wrapToPi([phase(bb_DTx)]),'-.')
                drawnow

                sfig(13); clf; hold on
                plot(([phase_bb]))
                plot(([phase_bbDTx]),'-.')                                          130
                xlim([max(1,i1-5) min(length(bb),i1+5)])
                drawnow

                sfig(11); plot(HPW); drawnow
                drawnow
            end
        end
    end
end
%HPW = round(HPW);                                                                  140


TxampMAX = 2^TransmitterParams.num_amp_bits;
bb_hpwalign = spline(HPW0sum(1:length(bb)), abs(bb), HPWsum(1:length(bb)));

% Produce the DTx signal for comparision
bb_DTx = DTxtoBB(HPW(1:length(AMP)),AMP,TransmitterParams, ReceiverParams);%, flipbit);
```

```
%% Plotting characteristics
t_rx = (1:length(bb_DTx))/ReceiverParams.fSample;
sfig(1); clf; plot(t_rx, .5*real(bb_DTx))
sfig(10); clf; plot(AMP)
sfig(11); clf; plot(HPW)

sfig(12); clf; hold on
plot(([phase_bb]))
plot(([phase(bb_DTx)]))

sfig(13); clf; hold on
plot(wrapToPi([phase_bb]))
plot(wrapToPi([phase(bb_DTx)]),'--')

sfig(14); clf; hold on
plot(wrapToPi([phase_bb])-wrapToPi([phase(bb_DTx(1:length(bb)).')]))

sfig(4);
plot(abs(bb)./abs(bb_DTx(1:length(bb)).'))

%% Corr properties - transmitting after the transducer
N2 = min([length(bb) length(bb_DTx)])-1;
bbcorr = xcorr(bb(1:N2));
bbdtxcorr = xcorr(bb_DTx(1:N2));

sfig(20); clf; hold on
plot(-N2+1:N2-1,20*log10(abs(bbcorr)))
plot(-N2+1:N2-1,20*log10(abs(bbdtxcorr) ./max(bbdtxcorr) *max(bbcorr)))

% plot(20*log10(abs(bbcorr)))
% plot(20*log10(abs(bbdtxcorr) ./max(bbdtxcorr) *max(bbcorr)))

%% Check against transducer - corr properties DTx before transducer

range = -length(a2)+1:length(a2)-1;
```

```
Nt = 50;
[TxRxC1_nadir_raw, ~,  freq] = transducer3(Nt,[0 0]);
[TxRxC2_nadir_raw, ~,  ~] = transducer3(Nt,[0 0]);
TxRxC1_nadir = TxRxC1_nadir_raw./norm(TxRxC1_nadir_raw,1);
TxRxC2_nadir = TxRxC2_nadir_raw./norm(TxRxC2_nadir_raw,1);

txrx_nadir = ifft(ifftshift(TxRxC1_nadir))';
txrx2_nadir = ifft(ifftshift(TxRxC2_nadir))';

% fSample = 100e3;
% fCentre = 250e3;
[txrx_nadir,TxRxC2_nadir_raw] = impulsetransducer1(Nt,[0 0],[fSample/2 fCentre]);
[txrx2_nadir,TxRxC2_nadir_raw] = impulsetransducer1(Nt,[0 0],[fSample/2 fCentre]);

a2_hpw = conv(bb_DTx(1:length(bb)),txrx2_nadir);
a2_hpw = conv(bb_DTx(1:length(bb)),txrx2_nadir);
a2 = conv(bb,txrx2_nadir);
aa_hpw = abs(xcorr(a2_hpw));
aa = abs(xcorr(a2));
figure(21); sfig(21); clf; hold on;
plot(range, 20*log10((aa)./max(aa)))
plot(range, 20*log10((aa_hpw)./max(aa_hpw)))
ylim([-60 0])
grid on

% save a_BBtoDTx_004 HPW AMP bb bb_DTx fSample fCentre
```

A.1.4 Mis-matched Filter Optimization

```
%% Mis−matched filter optimization
% Optimize MF for an incoming sonar code, using the CVX toolbox.
%  Rajiv Pratap
% run '/am/rialto/home1/pratapraji/MATLAB/cvx/cvx_setup.m'
% addpath(genpath('~/Masters/matlab/rajiv/transducer/'))
clear all;
% Load Digital Transmitter codes, processing their baseband signals
load('a_BBtoDTx_005');
b_1 = bb_DTx(1:length(bb)).';
load('b_BBtoDTx_005');
a_1 = bb_DTx(1:length(bb)).';
N = length(a_1);
fClock=204.8E6; R1 = 410;
fCentre = fClock/2/R1;
% transducer characteristics − nadir
Nt2 = 100;
[txrx_nadir,TxRxC1_nadir_raw] = impulsetransducer1(Nt2,[0 0],[fSample/2 fCentre]);
[txrx2_nadir,TxRxC2_nadir_raw] = impulsetransducer1(Nt2,[0 0],[fSample/2 fCentre]);
Nt = length(txrx_nadir); N2 = N+Nt−1;

a = conv(a_1,txrx2_nadir);
b = conv(b_1,txrx_nadir);

% transducer characteristics − angled
angle_range = [60:5:70];
M_angle = length(angle_range);
A_angle = []; B_angle = [];
for i = 1:length(angle_range)
    [txrx_angles(:,:,i),TxRxC2_nadir_raw] = impulsetransducer1(Nt2,[angle_range(i) 0],[fSample/2 fCentre]
    [txrx2_angles(:,:,i),TxRxC2_nadir_raw] = impulsetransducer1(Nt2,[−angle_range(i) 0],[fSample/2 fCent
```

```
    b_angle(:,i) = conv(b_1,txrx_angles(:,:,i));%M̂
    M̂
    B_angle_temp = convmtx(flip(conj(b_angle(:,i))),N2);M̂
    M̂
    B_angle = [B_angle; B_angle_temp];M̂                                    40
endM̂
M̂
clear 'TxRxC1' 'TxRxC2' 'A_angle_temp' 'B_angle_temp';M̂
M̂
M̂
lobe40 = 65;M̂
template = ones(2*N2-1,1);M̂
template(N2-lobe40:N2-1) = 10^(-40+3)*ones(length(N2-lobe40:N2-1),1);M̂
template(N2+1:N2+lobe40) = 10^(-40+3)*ones(length(N2+1:N2+lobe40),1);M̂
M̂                                                                      50
template2 = zeros(2*N2-1,1);M̂
template2(N2) = 1;M̂
M̂
A = convmtx(flip(conj(a)),N2);M̂
B = [imag(A).^2+real(A).^2];M̂
M̂
% Begin optimizationM̂
cvx_beginM̂
    variable x(N2) complexM̂
    minimize( ...M̂                                                     60
        max([ ...M̂
            pow_abs(A*x-template2,2) .*template; ...M̂
            pow_abs(B_angle*x,2) ...M̂
        ]))         M̂
cvx_endM̂
M̂
% Display results and saveM̂
h(1) = sfigure(1); clf; hold on;M̂
plot(-(N2-1):N2-1,20*log10(abs(xcorr(x,a))))M̂
for i=1:M_angleM̂                                                       70
    plot(-(N2-1):N2-1,20*log10(abs(xcorr(x,b_angle(:,i)))))M̂
endM̂
title('CVX MF properties for code b, and its detection against x')M̂
```

```
zoom on; grid on; ylim([−60 0])M̂
M̂
%save cvx_optimized_MF_bx_cross_BBtoDTx005 xM̂
%saveas(h(1),fullfile(sprintf('CorrProperties_CVX_MF_bx_cross_BBtoDTx005')),'fig');M̂
%saveas(h(1),fullfile(sprintf('CorrProperties_CVX_MF_bx_cross_BBtoDTx005')),'png');M̂
M̂
```

A.1.5 Mis-Matched Filter Gradient Optimization

```
% Optimization of Matched Filter coefficients for detecting received code a

clear all;

% Can swap b and a to choose which to optimize
% load('/am/rialto/home1/pratapraji/Masters/matlab/marco/a_BBtoDTx_004.mat');
% b_1 = bb_DTx(1:1600).';
% load('/am/rialto/home1/pratapraji/Masters/matlab/marco/b_BBtoDTx_004.mat');
% a_1 = bb_DTx(1:1600).';

% load('/u/students/pratapraji/Masters/matlab/rajiv/TransducerAngleRange/Figures004/v_set.mat
% N = length(v)/4;
% Nt = 50;
% N2 = N+Nt-1;
% a_1 = v(1:N)+1j*v(N+1:2*N);
% b_1 = v(2*N+1:3*N)+1j*v(3*N+1:4*N);

N = 200; Nt = 50; N2 = N+Nt-1; % Max appears to be N=2000
a_1 = randn(N,1) + 1j*randn(N,1);
b_1 = randn(N,1) + 1j*randn(N,1);
powerSkew = ((1:N)/(.1*N).*exp(-(1:N)/(.1*N))).'; powerSkew=powerSkew./norm(powerSkew);
a_1 = a_1./abs(a_1).*powerSkew;
b_1 = b_1./abs(b_1).*powerSkew;

N = length(a_1);
Nt = 50; N2 = N+Nt-1;
lobe40 = 105;

% Transducer characterised
Nt = 50;
[TxRxC1_nadir_raw, ~,  freq] = transducer3(Nt,[0 0]);
[TxRxC2_nadir_raw, ~,  ~] = transducer3(Nt,[0 0]);
TxRxC1_nadir = TxRxC1_nadir_raw./norm(TxRxC1_nadir_raw,1);
TxRxC2_nadir = TxRxC2_nadir_raw./norm(TxRxC2_nadir_raw,1);
txrx_nadir = ifft(ifftshift(TxRxC1_nadir))';
```

txrx2_nadir = ifft(ifftshift(TxRxC2_nadir))';

```
a = conv(a_1,txrx2_nadir);
b = conv(b_1,txrx_nadir);
                                                                    40
% Angled transducers characterised
angle_range = [60:5:70];
M_angle = length(angle_range);
for i = 1:length(angle_range)
    [TxRxC1(:,:,i), ~, ~] = transducer3(Nt,[angle_range(i) 0]);
    [TxRxC2(:,:,i), ~, ~] = transducer3(Nt,[-angle_range(i) 0]);
%     [TxRxC2(:,:,i), ~, ~] = transducer3(Nt,[angle_range(length(angle_range)-i+1) 0]);
    TxRxC1(:,:,i) = TxRxC1(:,:,i)./norm(TxRxC1_nadir_raw,1);
    TxRxC2(:,:,i) = TxRxC2(:,:,i)./norm(TxRxC2_nadir_raw,1);
    Nt = 50;                                                        50
    txrx_angles(:,:,i) = ifft(ifftshift(TxRxC1(:,:,i)))';
    txrx2_angles(:,:,i) = ifft(ifftshift(TxRxC2(:,:,i)))';

    a_angle(:,i) = conv(a_1,txrx2_angles(:,:,i));
    b_angle(:,i) = conv(b_1,txrx_angles(:,:,i));

    a_angle(:,i) = a_angle(:,i)./norm(a);
    b_angle(:,i) = b_angle(:,i)./norm(b);
end
                                                                    60
a=a./norm(a);
b=b./norm(b);

% Start point of optimization variable x as the optimial MF coefficients
% for SNR detection (autocorrelation)
x = a;


%% Loop for optimizing x
                                                                    70
mu=.05;
steps = Inf;%1e6;
```

```
template1 = zeros(2*N2-1,1);
template1(N2) = 1;

template2 = ones(2*N2-1,1);
template2(N2-lobe40:N2+lobe40) = 10^((-40+3)/20);
template2(N2)=1;
                                                                    80

for i=1:steps
   % Defining error, and correcting error regions
   xa1 = xcorr(x,a);
   xa = (xa1-template1).*template2;
   for i1=1:size(b_angle,2)
      xb_angle(:,i1) = xcorr(x,b_angle(:,i1));
   end

   xcorr_vector = [xa;reshape(xb_angle,[],1)];                      90
   [~,e_corr_ndx] = max(abs(xcorr_vector));
   e_corr = xcorr_vector(e_corr_ndx);

   G_ax = zeros(N2,1); % Gradient due to correlation in A
   G_bx = zeros(N2,1); % Gradient due to correlation in B
   if e_corr_ndx == N2  %Centre autocorr error
      G_ax = e_corr*a;
   elseif e_corr_ndx <= 2*N2-1  %Non-centre autocorr error
      shift = e_corr_ndx-N2;
      if shift < 0                                                  100
         G_ax = (e_corr)*([a(1-shift:end); zeros(-shift,1)]);
      else %shift > 0
         G_ax = (e_corr)*([zeros( shift,1);a(1:end-shift)]);
      end
      G_bx = zeros(N2,1);
   else % cross corr error
      shift = mod(e_corr_ndx-1,2*N2-1)+1-N2;
      i1 = 1+floor( e_corr_ndx./(2*N2-1) );

      if shift < 0                                                  110
         G_bx = (e_corr)*([b_angle(1-shift:end,i1); zeros(-shift,1)]);
```

```
        else %shift > 0
            G_bx = (e_corr)*([zeros( shift,1);b_angle(1:end-shift,i1)]);
        end
    end

    x = x - mu*(G_ax+G_bx);

    % Plotting during optimization
    if mod(i,1000) == 0                                                              120
        sfig(1); plot(-N2+1:N2-1,20*log10(abs([xa1 xb_angle]))); ylim([-60 0]); title(sprintf('
        sfig(2); plot(abs([a b x]))
        drawnow
    end
end
```

Bibliography

[1] BREHMER, P., DO CHI, T., AND MOUILLOT, D. Amphidromous fish school migration revealed by combining fixed sonar monitoring (horizontal beaming) with fishing data. *Journal of Experimental Marine Biology and Ecology 334*, 1 (2006), 139–150.

[2] CANDAN, C. On the design of mismatched filters with an adjustable matched filtering loss. In *Radar Conference, 2010 IEEE* (May 2010), pp. 1311–1316.

[3] CAPOCELLI, R., DESANTIS, A., AND VACCARO, U. *Sequences II: Methods in Communication, Security, and Computer Science*. Springer Science & Business Media, 2012.

[4] CHU, D. Polyphase codes with good periodic correlation properties (corresp.). *Information Theory, IEEE Transactions on 18*, 4 (Jul 1972), 531–532.

[5] CLAY, C. S., AND HORNE, J. K. Acoustic models of fish: the atlantic cod (gadus morhua). *The Journal of the Acoustical Society of America 96*, 3 (1994), 1661–1668.

[6] COLLINS, T., AND ATKINS, P. Nonlinear frequency modulation chirps for active sonar. *IEE Proceedings - Radar, Sonar and Navigation 146*, 6 (Dec 1999), 312–316.

[7] CROCKER, M. J. *Handbook of acoustics*. John Wiley & Sons, 1998.

[8] DAS, A., KUMAR, A., AND BAHL, R. Realistic ambient noise analysis for passive surveillance algorithm design. In *Underwater Technology (UT), 2011 IEEE Symposium on and 2011 Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC)* (April 2011), pp. 1–8.

[9] ECKART, C. Optimal rectifier systems for the detection of steady signals. *Scripps Institution of Oceanography* (1952).

[10] ECKART, C., COMMITTEE, N. D. R., ET AL. *Principles of Underwater Sound*. reprinted and distributed by the Research Analysis Group, Committee on Undersea Warfare, National Research Council, 1946.

[11] FOOTE, K. G. Importance of the swimbladder in acoustic scattering by fish: a comparison of gadoid and mackerel target strengths. *The Journal of the Acoustical Society of America 67*, 6 (1980), 2084–2089.

[12] FOOTE, K. G. Fish target strengths for use in echo integrator surveys. *The Journal of the Acoustical Society of America 82*, 3 (1987), 981–987.

[13] FRANK, R. L. Polyphase codes with good nonperiodic correlation properties. *Information Theory, IEEE Transactions on 9*, 1 (Jan 1963), 43–45.

[14] F.R.S., L. R. Xxxi. investigations in optics, with special reference to the spectroscope. *Philosophical Magazine Series 5 8*, 49 (1879), 261–274.

[15] FUCHS, E., AND MASOUM, M. A. *Power quality in power systems and electrical machines*. Academic press, 2011, ch. 2.5 - Ferroresonance of power transformers.

[16] GIANNAKIS, G., AND TSATSANIS, M. Signal detection and classification using matched filtering and higher order statistics. *Acoustics, Speech and Signal Processing, IEEE Transactions on 38*, 7 (Jul 1990), 1284–1296.

[17] GOLD, R. Optimal binary sequences for spread spectrum multiplexing (corresp.). *Information Theory, IEEE Transactions on 13*, 4 (October 1967), 619–621.

[18] GRANT, M., AND BOYD, S. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura, Eds., Lecture Notes in Control and Information Sciences. Springer-Verlag Limited, 2008, pp. 95–110. `http://stanford.edu/~boyd/graph_dcp.html`.

[19] GRANT, M., AND BOYD, S. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[20] GRONEMEYER, S., AND MCBRIDE, A. Msk and offset qpsk modulation. *IEEE Transactions on Communications 24*, 8 (Aug 1976), 809–820.

[21] JENSEN, J. L. W. V. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica 30*, 1 (1906), 175–193.

[22] KAY, S., AND SALISBURY, J. Improved active sonar detection using autoregressive prewhiteners. *The Journal of the Acoustical Society of America 87*, 4 (1990), 1603–1611.

[23] KEYSIGHT TECHNOLOGIES. *30 MHz Function/Arbitrary Waveform Generators*, 08 2014. Data Sheet.

[24] KNIGHT, W., PRIDHAM, R. G., AND KAY, S. Digital signal processing for sonar. *Proceedings of the IEEE 69*, 11 (Nov 1981), 1474.

[25] KRANTZ, S. G. Jensen's inequality. In *Handbook of complex variables*. Springer Science & Business Media, 2012, ch. 9.1.3.

[26] LEBLANC, L. R., MAYER, L., RUFINO, M., SCHOCK, S. G., AND KING, J. Marine sediment classification using the chirp sonar. *The Journal of the Acoustical Society of America 91*, 1 (1992), 107–115.

[27] LEVANON, N., AND MOZESON, E. *Radar signals*. John Wiley & Sons, 2004.

[28] LOVE, R. H. Measurements of fish target strength: a review. *Fish. Bull 69*, 4 (1971), 703–715.

[29] LUCCHETTI, A., SALA, A., AND JECH, J. M. Impact and performance of mediterranean fishing gear by side-scan sonar technology. *Canadian journal of fisheries and aquatic sciences 69*, 11 (2012), 1806–1816.

[30] MCAULAY, R., JOHNSON, J., AND LAB., M. I. O. T. L. L. *Optimal Mismatched Filter Design for Radar Ranging and Resolution*. Technical note. Defense Technical Information Center, 1969.

[31] MELVIN, G., LI, Y., MAYER, L., AND CLAY, A. Commercial fishing vessels, automatic acoustic logging systems and 3d data visualization. *ICES Journal of Marine Science: Journal du Conseil 59*, 1 (2002), 179–189.

[32] MISUND, O. A. Underwater acoustics in marine fisheries and fisheries research. *Reviews in Fish Biology and Fisheries 7*, 1 (1997), 1–34.

[33] MOHAN, N., AND UNDELAND, T. M. *Power electronics: converters, applications, and design*. John Wiley & Sons, 2007, ch. 23 - Power Supplies.

[34] MORI, J.-L., AND GOUNON, P. The use of stochastic matched filter in active sonar. In *Signal Processing Conference, 2000 10th European* (Sept 2000), pp. 1–4.

[35] MORSE, P., AND FESHBACH, H. *Methods of theoretical physics*. No. v. 2 in International series in pure and applied physics. McGraw-Hill, 1953.

[36] POPOVIC, B. M. Generalized chirp-like polyphase sequences with optimum correlation properties. *IEEE Transactions on Information Theory 38*, 4 (Jul 1992), 1406–1409.

[37] PROAKIS, J. G., SALEHI, M., ZHOU, N., AND LI, X. *Communication systems engineering*, vol. 94. Prentice Hall New Jersey, 1994, pp. 70–131.

[38] RABBANI, S. Derivation of the Matched Filter as the Highest SNR Linear Estimator. `http://srabbani.com/matched_filter.pdf`, 2006.

[39] RAHMATI, M., PANDEY, P., AND POMPILI, D. Separation and classification of underwater acoustic sources. In *Underwater Communications and Networking (UComms), 2014* (Sept 2014), pp. 1–5.

[40] ROHLING, H. Mismatched filter design for pulse compression. In *Radar Conference, 1990., Record of the IEEE 1990 International* (May 1990), pp. 253–257.

[41] ROSE, C. S., STONER, A. W., AND MATTESON, K. Use of high-frequency imaging sonar to observe fish behaviour near baited fishing gears. *Fisheries research 76*, 2 (2005), 291–304.

[42] ROSS, D. *Mechanics of underwater noise*. Elsevier, 2013.

[43] ROUPHAEL, T. J. *RF and digital signal processing for software-defined radio: a multi-standard multi-mode approach*. Newnes, 2009, ch. 7.3.4.

[44] ROYDEN, H. L., AND FITZPATRICK, P. *Real analysis*, vol. 198. Macmillan New York, 1988.

[45] SCHOCK, S. G. A method for estimating the physical and acoustic properties of the sea bed using chirp sonar data. *IEEE Journal of Oceanic Engineering 29*, 4 (Oct 2004), 1200–1217.

[46] SCHOLNIK, D. P. Optimal filters for range-time sidelobe suppression. In *Signal Processing Conference, 2000 10th European* (2000), IEEE, pp. 1–4.

[47] SIMIC, S., ZEJAK, A., AND GOLUBICIC, Z. Range sidelobe reduction in the portable battlefield surveillance radar. In *Telecommunication in Modern Satellite Cable and Broadcasting Services (TELSIKS), 2011 10th International Conference on* (Oct 2011), vol. 2, pp. 571–574.

[48] SNYMAN, J. A. Standard methods for constrained optimization. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms* (2005), 57–96.

[49] SUNARDI, S., DIN, J., YUDHANA, A., AND HASSAN, R. B. R. Target strength for fish identification using echo sounder. *Applied Physics Research 1*, 2 (2009), 92.

[50] TREES, H. L. V. *Detection, Estimation, and Modulation Theory: Radar-Sonar Signal Processing and Gaussian Signals in Noise*. Krieger Publishing Co., Inc., 1992.

[51] TREVORROW, M. V. Boundary scattering limitations to fish detection in shallow waters. *Fisheries Research 35*, 12 (1998), 127 – 135.

[52] VACCARO, R. The past, present, and the future of underwater acoustic signal processing. *Signal Processing Magazine, IEEE 15*, 4 (Jul 1998), 21–51.

[53] WALDRON, S. Generalized welch bound equality sequences are tight frames. *Information Theory, IEEE Transactions on 49*, 9 (Sept 2003), 2307–2309.

[54] WELCH, L. Lower bounds on the maximum cross correlation of signals (corresp.). *Information Theory, IEEE Transactions on 20*, 3 (May 1974), 397–399.

[55] WENZ, G. M. Review of underwater acoustics research: Noise. *The Journal of the Acoustical Society of America 51*, 3B (1972), 1010–1024.

[56] WILSON, O. *Introduction to the Theory and Design of Sonar Transducers Peninsula*. Los Altos, CA, USA, 1988, p. 1.

[57] WINDER, A. Ii. sonar system technology. *Sonics and Ultrasonics, IEEE Transactions on 22*, 5 (Sept 1975), 291–332.

[58] WIRTH, W.-D. *Radar techniques using array antennas (FEE radar, sonar, navigation & avionics series)*, vol. 10. IET, 2001, pp. 125–155.