Intelligent Detection of Guided Scrambling Coded Sequences

by

Edward Carle

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Communications

Department of Electrical and Computer Engineering

University of Alberta

Abstract

This document presents a case study in the as-yet unexplored avenue of exploiting running digital sum (RDS) statistics to achieve reduced error rates in the detection of balanced guided scrambling (GS) coded sequences. As demonstrated in this research, balanced GS codes are a good candidate for this approach as the properties of GS coded sequences that produce desirable spectral results translate well to desirable RDS statistics for detection. Through software simulation of GS coded sequences, it is demonstrated that the RDS can be accurately approximated as a cyclostationary Gaussian Markov process. Using an intelligent detection technique developed in this work that takes these RDS statistics into account, error rates over additive white Gaussian noise (AWGN) channels are shown to be significantly reduced. While this research focused on GS coded sequences selected using the minimum square weight (MSW) criteria, the document proposes the development of lower rate codes with high-order spectral-shaping properties that might lend themselves well to this intelligent detection technique achieving even greater reduction in error rates.

Acknowledgements

Although this research work officially falls within the scope of an MSc degree, it has been a work in progress since my undergraduate years. Because of this, acknowledgements are due over many years of influence. To my wife Sophie: there are few in your position that would be so tolerant and supportive of ones academic pursuits in the midst of building a family; rarely a day passes that I don't count myself incredibly fortunate to have met you. My friend and fellow engineering graduate student James Maldanar: you have been an invaluable second brain when I've gotten myself stuck on seemingly trivial problems. Scott Joly — my oldest friend: having a non-engineer who is both willing and capable of discussing my research has been a constant source of insight that one just cannot acquire talking to other engineers. Professors Witold Krzymień, Chintha Tellambura and Masoud Ardakani: many of the ideas and insights I've relied upon for this work have come to me while listening to your lectures. Lastly my supervisor, Professor Ivan Fair: you have been an unparallelled mentor in this realm for many years now. This is not the first time I have acknowledged your guidance and inspirations and I'm quite confident it will not be the last.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document explores exploiting the statistical properties of balanced guided scrambling (GS) codes to achieve reduced error rates when detecting GS coded sequences across signal corrupting channels. To accomplish this an approximate model is created to statistically characterize GS coded sequences and an intelligent detection technique is applied that takes this model into consideration to reduce the probability of erroneous sequence detection. Both the GS coding technique and the aforementioned intelligent detection technique are implemented in software, therefore the model and results are generated largely through computational intensive simulations. The software and all associated data has been made available to the reader as part of a free and open-source software suite.[1]

1.1 Overview

The reader should first familiarize themselves with the notation and conventions outlined in section 1.2 on the following page. Chapter 2 on page 6 covers much of the background material with which the reader should be familiar,

[1] Guided Scrambling GNU Radio Module: `https://github.com/eddic/gr-gs`.

touching on both constrained sequence coding and signal detection while focusing on guided scrambling. Chapter 3 on page 41 explores various methods and their justifications for building approximate statistical models of GS coded sequences and proposes an algorithmic approach to intelligently detect these sequences. A rudimentary explanation of the techniques used to implement the approach in software is also included in this chapter. Chapter 4 on page 62 itemizes the results, highlighting the reduced sequence detection error rates, for two specific code configuration cases, and outlines additional observations of note made while generating these results. From these results chapter 5 on page 70 draws conclusions as to the value this work might hold while exploring a possible future research avenue that was observed.

1.2 Notation & Conventions

Throughout this document the following notation and conventions are used.

Normalized frequency, denoted $\hat{f}$, is a unitless measure of frequency that is derived from the actual frequency f normalized by the signal centre frequency f_0 and baud rate $f_s = 1/T_s$ according to equation (1.1).

$$\hat{f} = \frac{f - f_0}{f_s} = (f - f_0)T_s \tag{1.1}$$

For baseband as opposed to passband signals, $f_0 = 0$ and $\hat{f}$ is simply $\frac{f}{f_s}$.

Normalized spectral density, $\hat{Y}(\hat{f})$, is also unitless and is a measure of spectral density derived from the actual spectral density $Y(f)$ as a function of normalized frequency $\hat{f}$. It is modified by centre frequency f_0 and baud rate f_s as well as the symbol energy E_s.

$$\hat{Y}\left(\hat{f}\right) = \frac{Y(f)}{\sqrt{f_s E_s}} = \frac{Y\left(\hat{f} f_s + f_0\right)}{\sqrt{f_s E_s}} \tag{1.2}$$

This normalization process ensures that the total energy per symbol of the

normalized signal is always equal to 1.

$$\int_{-\infty}^{\infty} \left|\hat{Y}\left(\hat{f}\right)\right|^2 d\hat{f} = 1$$

The normalized *power* spectral density (PSD) is simply the square of the magnitude of the normalized spectral density:

$$\left|\hat{Y}\left(\hat{f}\right)\right|^2 = \frac{|Y(f)|^2}{\sqrt{f_s E_s}} = \frac{\left|Y\left(\hat{f} f_s + f_0\right)\right|^2}{f_s E_s} \tag{1.3}$$

Noise power, n_0, refers to the average amount of noise energy per symbol *per complex axis*. Given an additive white Gaussian noise source with standard deviation σ:

$$n_0 = \sigma^2 \tag{1.4}$$

Thus for real valued signals n_0 equals the total noise energy per symbol whereas for complex values signals the total noise energy per symbol is $2n_0$.

Identifiers for sequences, vectors and matrices are expressed in bold font. For example $\mathbf{X} = \{x_k\}_{k=1}^{K}$ identifies a sequence $\mathbf{X}$ with elements x_k defined for $k \in [1, k]$.

Random variables (RV) are always capitalized (X) but not all capitalized variables imply a random variable. Random processes (RP) contain a subscript k (X_k). The use of the subscript n (X_n) refers specifically to a cyclostationary random process. All probabilistic functions are expressed using blackboard fonts. Table 1.1 on the following page lists all such functions used in this document.

Throughout this document probability mass functions (PMF) will be constructed by sampling continuously defined functions. The symbol ζ will be used exclusively to denote a continuity correction ensuring a valid PMF. Assuming a continuously defined function f and a discrete random variable X,

$\mathbb{P}[X = x]$	Probability that RV $X = x$ or RP $X_k = x$ over all k
$\mathbb{P}[X_k = x]$	Probability that RP $X_k = x$ at index k
$\mathbb{P}[X_n = x]$	Probability that the cyclostationary RP $X_n = x$ at position n within the cyclic period N.
$\mathbb{E}[X]$	Expected value of RV X or RP X_k over all k
$\mathbb{E}[X_k]$	Expected value of RP X_k at index k
$\mathbb{E}[X_n]$	Expected value of cyclostationary RP X_n at position n within the cyclic period N.
$\mathbb{V}[X]$	Variance of RV X or RP X_k over all k
$\mathbb{V}[X_k]$	Variance of RP X_k at index k
$\mathbb{V}[X_n]$	Variance of cyclostationary RP X_n at position n within the cyclic period N.
$\mathbb{C}_{XX}[\tau]$	Autocovariance of weakly stationary RP X_k
$\mathbb{C}_{XX}[n, \tau]$	Autocovariance of weakly cyclostationary RP X_k. $n \in [1, N]$ where N is the cyclic period.
$\mathbb{C}_{XX}[k, \tau]$	Autocovariance of non-stationary RP X_k.
$\mathbb{R}_{XX}[\tau]$	Autocorrelation of weakly stationary RP X_k
$\mathbb{R}_{XX}[n, \tau]$	Autocorrelation of weakly cyclostationary RP X_k. $n \in [1, N]$ where N is the cyclic period.
$\mathbb{R}_{XX}[k, \tau]$	Autocorrelation of non-stationary RP X_k.

Table 1.1: Probabilistic functions used in document

the PMF is defined as:

$$\mathbb{P}[X = x] = \zeta f(x)$$

where

$$\zeta = \frac{1}{\sum_{\forall x} f(x)}$$

thereby ensuring that

$$\sum_{\forall x} \mathbb{P}[X = x] = 1$$

Digital symbols can be expressed in two states. A symbol x will represent a value from the field $\mathrm{GF}(2^M)$ while $\tilde{x}$ with a tilde will represent a value from the signalling constellation. For example, the binary non-return-to-zero case would be:

$$x \in \mathrm{GF}(2^1) \in \{0, 1\}$$

$$\tilde{x} \in \mathbb{R} \in \{-1, 1\} = \begin{cases} -1, & x = 0 \\ 1, & x = 1 \end{cases}$$

With the 4-state quadrature amplitude modulated case this might be:

$$x \in \mathrm{GF}(2^2) \in \{0, 1, 2, 3\}$$

$$\tilde{x} \in \mathbb{C} \in \{1 + j, -1 + j, 1 - j, -1 - j\} = \begin{cases} 1 + j, & x = 0 \\ -1 + j, & x = 1 \\ 1 - j, & x = 2 \\ -1 - j, & x = 3 \end{cases}$$

The caligraphic symbol $\mathcal{F}$ is used exclusively to denote a Fourier transform while subscripting indicates which variables are tranformed. For example $\mathcal{F}_{\tau \to \nu}\{g(\tau)\}(\nu)$ denotes a Fourier transform of $g(\tau)$ with the time parameter τ transformed to the frequency parameter ν.

Chapter 2

Background

The research topics covered by this document straddle two sub-fields in the realm of communications. For this reason and in an effort to maintain brevity, the background information in this chapter outlines a limited number of relevant details from the areas of channel coding and signal detection. This treatment assumes a background in both probability theory [1] and information theory [2].

2.1 Channel Coding

Channel coding in general, and constrained sequence coding in particular, seeks to encode a sequence of digital symbols so as to ensure that the resulting sequence satisfies a set of constraints [3].

One of the simplest examples of a constraint often placed upon a sequence involves the *run length*. Should a sequence of digital symbols contain a long subsequence of identical symbol values, recovering timing information for detection may become unreliable [4, p. 51]. Thus one might design a code to ensure that, regardless of the source data, runs of identical symbols are limited to a length at most $k + 1$. Figure 2.1 on the next page presents an

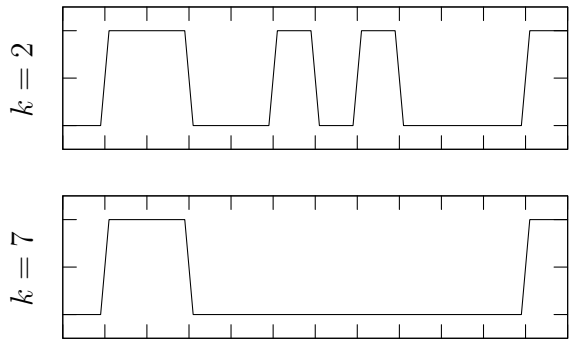

Figure 2.1: Run length limited sequence examples. The top sequence satisfies a $k = 2$ constraint while the bottom has $k = 7$.

example of run-length limited binary sequences. Since the pool of acceptable symbol sequences is now limited, the only way to maintain the same flow of information is to allow for redundancy in the channel while increasing the baud rate. The concept of *capacity* is introduced in [3] as the maximum amount of information that can flow through a constrained channel per encoded symbol or per second. As shown in [4, p. 60], the capacity for run-length limited channels with $k \gg 1$ is:

$$C(k) \approx 1 - \frac{2^{-k}}{4 \ln 2} \tag{2.1}$$

with units of bits of information per coded symbol.

Although often considered a framing strategy, the method proposed by Gordon Bell [5] for use in universal asynchronous receiver-transmitter (UART) devices is one of the simplest and most ubiquitous run length limited codes. The idea is to encapsulate a source word of data in start and stop symbols. The example in figure 2.2 on the following page is the most common configuration used in RS-232/485/422 systems [6, p. 1038].

The *code rate* R is a measure of how much redundancy is introduced by

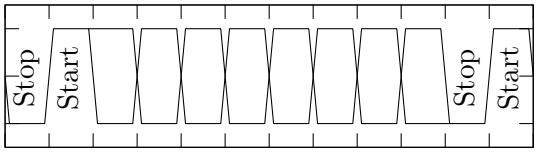

Figure 2.2: UART 8N1 style bit sequence with binary source words of length 8 preceded by a single start bit (1) and terminated by a single stop bit (0).

the code in order to meet the constraints. When both source words and codewords have fixed length, the code rate is defined as:

$$R \equiv \frac{\text{source word length}}{\text{codeword length}} \tag{2.2}$$

The example in figure 2.2 contains 2 bits of redundancy in the start/stop section with source words of length 8, therefore it has a code rate of:

$$R = \frac{8}{8+2} = \frac{4}{5} = 0.8$$

Due to the start/stop bits, the run length of 1s or 0s in the 8N1 configuration is always bounded at 9, thus $k = 8$. However from equation (2.1) on the preceding page, the capacity of a run length limited code with $k = 8$ is

$$C(k=8) \approx 1 - \left.\frac{2^{-k}}{4\ln 2}\right|_{k=8} \approx 0.998$$

A capacity of 0.998 implies that a code should exist for which only 1 out of every 500 bits need be redundant. From this one can conclude that although the UART strategy is simple to implement, it is not very efficient in terms of maximizing the flow of information across the channel. To compare codes the *code efficiency*, η, indicates how close a code is to capacity [4, p. 96]:

$$\eta = \frac{R}{C} \tag{2.3}$$

Source Word	Codeword
0	$x0$
1	01

Table 2.1: MFM coding rules. The x shall always be the complement of the last bit of the previous codeword.

Based on this expression, the efficiency of this UART configuration is $\frac{0.8}{0.998} = 80\%$ while the maximum code efficiency is $\eta = 100\%$.

Often, constrained sequence encoders are implemented with a state dependent table that maps source words directly to code words. To understand their operation it is helpful to examine them as both mapping tables and finite state machines. The modified frequency modulation (MFM) code is a popular run length limited code with $k = 2$.[1] See table 2.1 for the direct

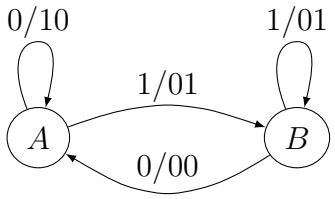

Figure 2.3: MFM state diagram. The input/output mapping is on the state transition.

source word to codeword mapping and figure 2.3 for the state diagram. A codeword ending in 0 puts the system into state A while a codeword ending in 1 puts it into state B.

Input: `1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1`

Output: `0100100101010100100100101001000 1`

[1] This analysis of MFM neglects *precoding* and its effects. See [4, p. 98] for clarification.

Note that although the encoder is state dependent, the decode process is entirely state independent: the second bit of the codeword is decoded as the source bit. Effectively all that is happening is a *clock bit* is being inserted between source bits. The ability of a code to be decoded in a state independent fashion is desired as it ensures detection errors do not propagate into subsequent codewords [4, p. 104].

2.2 Balanced Codes

Another constraint often placed on a sequence, and it is this constraint that is relevant to this research, is that the sequence is *balanced*. Balanced, in this context, means that the sum of digital symbols over time is bounded. This sum, as a sequence progresses, is referred to as the *running digital sum* (RDS). Assuming a source symbol sequence $\mathbf{X} = \{x_k\}_{k=1}^{K}, x_k \in \mathrm{GF}(2^M), \tilde{x}_k \in \mathbb{C}$, the RDS sequence ($\mathbf{\Phi} = \{\phi_k\}_{k=0}^{K}, \phi_k \in \mathbb{C}$) is defined as:

$$\phi_k = \phi_{k-1} + \tilde{x}_k = \phi_0 + \sum_{n=1}^{k} \tilde{x}_n \tag{2.4}$$

Figure 2.4 on the next page illustrates an RDS sequence with source symbols $x_k \in \mathrm{GF}(2^1), \tilde{x}_k \in \{-1, 1\}, \phi_k \in \mathbb{Z}$.

The RDS of an uncoded sequence can be described by its statistical characteristics as a random process (RP) Φ_k from uncorrelated symbols X_k where the probability of 1 and 0 are p and $1-p$, respectively. This is clearly a random walk [1, p. 317] but to characterize this, X_k will be first redefined in terms of a Bernoulli distributed RP X'_k:

$$\mathbb{P}[X_k = x] = \begin{cases} 1-p & x = 0 \\ p & x = 1 \end{cases}$$

$$X'_k \sim \mathrm{Bernoulli}(p)$$

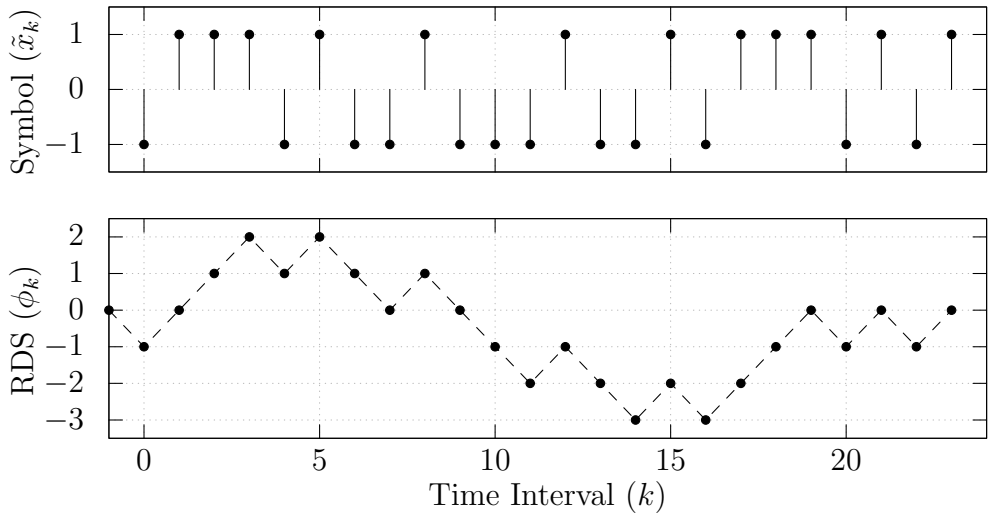

Figure 2.4: $x_k \in \mathrm{GF}(2^1), \tilde{x}_k \in \{-1, 1\}$ symbol sequence and its corresponding RDS sequence.

$$X_k = 2X'_k - 1$$
$$\Phi_k = \phi_0 + \sum_{n=1}^{k} \widetilde{X}_k = \phi_0 + \sum_{n=1}^{k} \left(2\widetilde{X}'_k - 1\right) = \phi_0 + 2\sum_{n=1}^{k} \widetilde{X}'_k - k$$

In [1, p. 54] it is shown that a sum of independent and identical Bernoulli distributed random variables gives a binomial distributed random variable.

$$Y_k = \sum_{n=1}^{k} \widetilde{X}'_k \sim B(k, p)$$
$$\Phi_k = \phi_0 + 2Y_k - k$$
$$\mathbb{E}[\Phi_k] = \phi_0 + 2\mathbb{E}[Y_k] - k = \phi_0 + 2kp - k$$
$$\mathbb{E}[\Phi] = \text{Undefined} \Leftrightarrow k \to \infty$$
$$\mathbb{V}[\Phi_k] = \mathbb{V}\left[\Phi_0 + 2Y_k - k\right] = 4\mathbb{V}\left[Y_k\right] = 4kp(1-p)$$
$$\mathbb{V}[\Phi] = \text{Undefined} \Leftrightarrow k \to \infty$$

It can therefore be seen that the first order statistics of the RDS are effectively unbounded if k is also unbounded. Although the mean will collapse to ϕ_0 in the case of a symmetric[2] channel, the variance will continue to diverge with k so long as $p \neq 0$ or $p \neq 1$.

A balanced code will therefore seek to ensure that the mean $\mathbb{E}[\Phi]$ and the variance $\mathbb{V}[\Phi]$ are minimized regardless of k. To increase predictability regardless of the source data, it is also preferred that these statistics be independent of the source data statistics. More specifically the balanced code seeks to minimize the *sum* of the square of the mean and variance, $\mathbb{E}[\Phi]^2 + \mathbb{V}[\Phi] = \mathbb{E}[\Phi^2]$.

A simple and commonly used balanced code is the Manchester code. See table 2.2 for the coding rules and figure 2.5 on the next page for an example. Note that the RDS at the end of each codeword is always zero. From this it is obvious that the RDS can never surpass 1 or -1, but a more rigorous

Source Word	GF(2^1) Codeword	Mapped Codeword
0	01	$\{-1, 1\}$
1	10	$\{1, -1\}$

Table 2.2: Manchester Coding Rules.

analysis of its statistics is still warranted.

To calculate the mean RDS $\mathbb{E}[\Phi]$ of a Manchester encoded RP, each codeword can be treated as as its own symbol. This derivation assumes the same source statistics discussed above where the probability of 1 and 0 are p and $1 - p$, respectively.

$$\begin{aligned}\mathbb{E}[\Phi] &= \mathbb{P}[X = 0]\frac{-1+0}{2} + \mathbb{P}[X = 1]\frac{1+0}{2} \\ &= \frac{p - (1-p)}{2} = p - \frac{1}{2}\end{aligned}$$

[2]In a binary symmetric channel, $p = 0.5$.

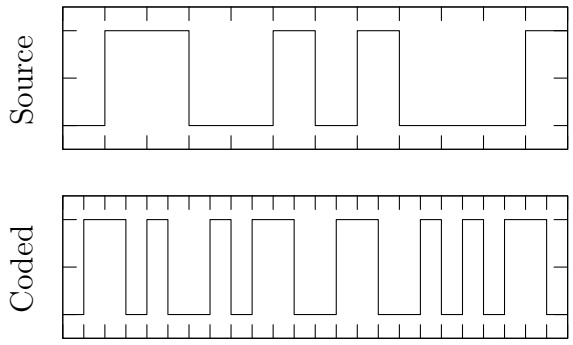

Figure 2.5: Sample Manchester coded sequence

Thus the mean RDS is completely independent of k, but is dependent on the source statistics. Changes in p will cause the mean RDS value to vary in the range $(-1/2, 1/2)$.

The variance can be calculated using similar means.

$$\begin{aligned}\mathbb{V}[\Phi] &= \mathbb{E}[\Phi^2] - \mathbb{E}[\Phi]^2 \\ &= \mathbb{P}[X=0]\frac{(-1)^2+0^2}{2} + \mathbb{P}[X=1]\frac{1^2+0^2}{2} - \left(p-\frac{1}{2}\right)^2 \\ &= \frac{(1-p)+p}{2} - \left(p-\frac{1}{2}\right)^2 = \frac{1}{2} - \left(p-\frac{1}{2}\right)^2\end{aligned}$$

Like the mean, the variance is independent of k but has a dependency on the source statistics. It will vary in the range $(1/4, 1/2)$. Figure 2.6 on the following page plots the dependencies.

Although the mean and variance of the RDS are dependent on the source statistics p, $\mathbb{E}[\Phi^2] = \mathbb{V}[\Phi] + \mathbb{E}[\Phi]^2$ is constant at $1/2$. Thus Manchester coding sets $\mathbb{E}[\Phi^2] = 1/2$ regardless of the source statistics p and the position in the sequence k.

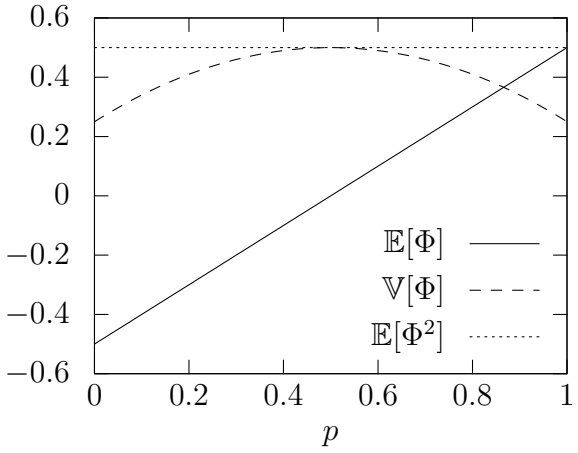

Figure 2.6: Manchester coding RDS (Φ) first-order statistics.

2.2.1 Spectral Shaping

Now consider why one might seek to control the RDS statistics. At first glance the benefits seem to be similar to that of the run length limited codes discussed in section 2.1 on page 6: the Manchester code discussed in section 2.2 is clearly limited to run lengths of 2. It turns out that this property of balanced codes, although beneficial, is secondary to the larger goal of *spectral shaping* [4, p. 195].

The raised cosine pulse, as seen in figure 2.7 on the following page, is one of the more commonly implemented pulse shapes in digital systems. It satisfies the Nyquist criteria for a pulse with zero inter-symbol interference (ISI)[3] and is band-limited [7, p. 134]. Equation (2.5) on the next page defines

[3]Assuming a channel free of distortion.

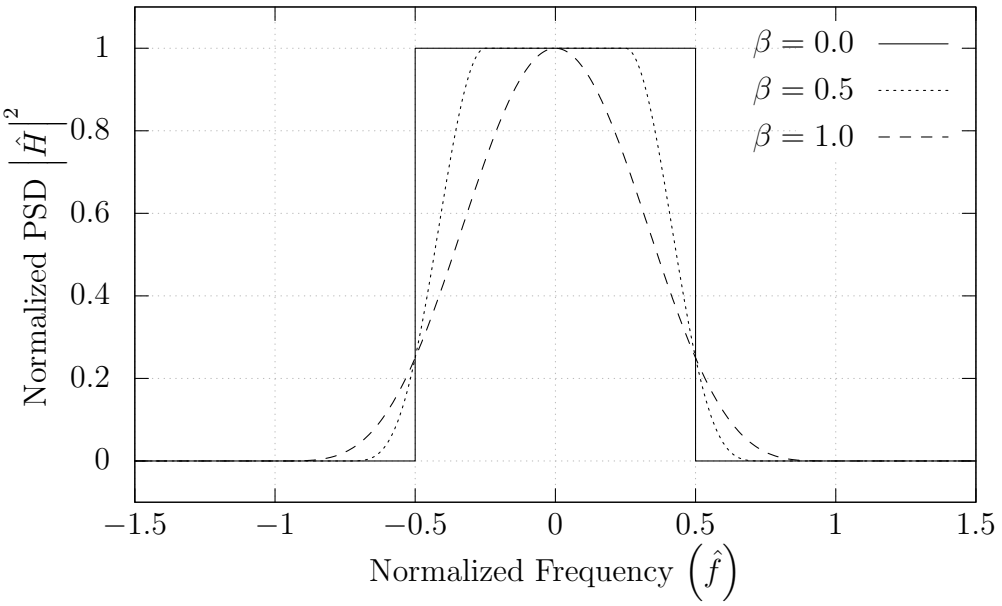

Figure 2.7: Frequency domain representation of a normalized raised-cosine pulse.

its normalized spectral density.

$$\hat{H}\left(\hat{f}\right) = \begin{cases} 1, & \left|\hat{f}\right| \leq \frac{1-\beta}{2} \\ \cos^2\left(\frac{\pi}{2\beta}\left[\left|\hat{f}\right| - \frac{1-\beta}{2}\right]\right), & \text{otherwise} \\ 0, & \left|\hat{f}\right| \geq \frac{1+\beta}{2} \end{cases} \tag{2.5}$$

The roll-off or excess-bandwidth factor, β, falls in the range $[0, 1]$. This parameter allows a trade off between total bandwidth utilization and pulse decay in the time domain. In channels that are more susceptible to distortion, it may be appropriate to choose a value of β closer to 1 in order to mitigate the effects of inter-symbol interference. In channels with heavy bandwidth constraints, it may be appropriate to choose a value of β closer to 0.

When digital signals are represented spectrally, often only the contribu-

tion of the pulse itself is considered. Since the information in each symbol is carried by the pulse, the energy spectral density of a single symbol in a digital signal is equivalent to the energy spectral density of that pulse. However, for a sequence of symbols, the statistics of the source data also contribute. If the statistics of the source data are unknown, however, the pulse spectrum, e.g. figure 2.7 on the preceding page, is the best approximation.

In the event that the source statistics *are* known, the pulse-based spectral approximation can be refined. The noiseless signal traversing the channel $y(t)$ is actually the convolution of the continuously defined transmitted symbols $x(t)$ with the pulse shape $h(t)$:

$$\begin{aligned} x(t) &= \sum_{k=0}^{K} \tilde{x}_k \delta(t - kT_s) \\ &= \tilde{x}_0\delta(t) + \cdots + \tilde{x}_k\delta(t - kT_s) + \cdots + \tilde{x}_K\delta(t - KT_s) \\ y(t) &= x(t) * h(t) \\ Y(f) &= X(f)H(f) \\ |Y(f)|^2 &= |X(f)|^2|H(f)|^2 \end{aligned} \tag{2.6}$$

Thus the energy spectral density of the signal is the product of the energy spectral density of the pulse and the symbol sequence. It is of interest then to calculate the energy spectral density of a symbol sequence.

A sequence of symbols, $\mathbf{X} = \{x_k\}_{k=0}^{K}, x_k \in \mathrm{GF}(2^M), \tilde{x}_k \in \mathbb{C}$, with symbol period T_s, can be represented as a sequence of impulses. Its energy spectral density can be evaluated as follows.

$$X(f) = \mathcal{F}\{x(t)\} = \sum_{k=0}^{K} \tilde{x}_k \mathcal{F}\{\delta(t - kT_s)\} = \sum_{k=0}^{K} \tilde{x}_k e^{-2\pi jkT_s f}$$

$$\begin{aligned} |X(f)|^2 &= \left| \sum_{k=0}^{K} \tilde{x}_k e^{-2\pi jkT_s f} \right|^2 \\ &= |\tilde{x}_0|^2 e^{-2\pi j(0)T_s f} e^{2\pi j(0)T_s f} \end{aligned}$$

$$
\begin{aligned}
&+ \tilde{x}_0\tilde{x}_1^* e^{-2\pi j(0)T_s f} e^{2\pi j(1)T_s f} \\
&\qquad \vdots \\
&+ \tilde{x}_0\tilde{x}_K^* e^{-2\pi j(0)T_s f} e^{2\pi jKT_s f} \\
&+ \tilde{x}_1\tilde{x}_0^* e^{-2\pi j(1)T_s f} e^{2\pi j(0)T_s f} \\
&+ |\tilde{x}_1|^2 e^{-2\pi j(1)T_s f} e^{2\pi j(1)T_s f} \\
&\qquad \vdots \\
&+ \tilde{x}_1\tilde{x}_K^* e^{-2\pi j(1)T_s f} e^{2\pi jKT_s f} \\
&\qquad \vdots \\
&+ \tilde{x}_K\tilde{x}_0^* e^{-2\pi jKT_s f} e^{2\pi j(0)T_s f} \\
&+ \tilde{x}_K\tilde{x}_1^* e^{-2\pi jKT_s f} e^{2\pi j(1)T_s f} \\
&\qquad \vdots \\
&+ |\tilde{x}_K|^2 e^{-2\pi jKT_s f} e^{2\pi jKT_s f} \\
= &\sum_{k=0}^{K}\sum_{n=0}^{K} \tilde{x}_k\tilde{x}_n^* e^{-2\pi(k-n)T_s f}
\end{aligned}
$$

Since energy spectral density becomes unbounded as $K \to \infty$, analysis of power spectral density proves to be more practical for arbitrarily long symbol sequences. The power spectral density (PSD) of a signal is defined in terms of the energy spectral density normalized by the total signal time [8]:

$$\overline{|X(f)|^2} = \frac{1}{(K+1)T_s} \sum_{k=0}^{K}\sum_{n=0}^{K} \tilde{x}_k\tilde{x}_n^* e^{-2\pi(k-n)T_s f}$$

This form, however, can be modified to expose a discrete-time Fourier transform with two changes. First substitute $n = k - \tau$. Second, the limits of the inner summation can be changed to $(-\infty, \infty)$ without affecting the result since symbol values outside the original range are simply defined to be zero.

$$\overline{|X(f)|^2} = \frac{1}{(K+1)T_s} \sum_{k=0}^{K}\sum_{\tau=k}^{k-K} \tilde{x}_k\tilde{x}_{k-\tau}^* e^{-2\pi\tau T_s f}$$

$$= \frac{1}{(K+1)T_s} \sum_{k=0}^{K} \sum_{\tau=-\infty}^{\infty} \tilde{x}_k \tilde{x}_{k-\tau}^* e^{-2\pi\tau T_s f}$$
$$= \frac{1}{(K+1)T_s} \sum_{k=0}^{K} \mathcal{F}_{\tau \to f}\{\tilde{x}_k \tilde{x}_{k-\tau}^*\}(T_s f) \tag{2.7}$$

Equation (2.7) describes the PSD for deterministic symbol sequences. For a symbol RP X_k, it can be expressed in terms of the autocovariance:

$$\mathbb{C}_{XX}[k,\tau] = \mathbb{E}\left[\widetilde{X}_k \widetilde{X}_{k-\tau}^*\right] - \left|\mathbb{E}\left[\widetilde{X}_k\right]\right|^2$$
$$\overline{|X(f)|^2} = \frac{1}{(K+1)T_s} \sum_{k=0}^{K} \mathcal{F}_{\tau \to f}\left\{\mathbb{C}_{XX}\left[k,\tau\right] + \left|\mathbb{E}\left[\widetilde{X}_k\right]\right|^2\right\}(T_s f)$$

For a weakly stationary symbol RP, this reduces to:

$$\overline{|X(f)|^2} = \frac{\mathcal{F}_{\tau \to f}\{\mathbb{C}_{XX}[\tau]\}(T_s f) + \left|\mathbb{E}\left[\widetilde{X}\right]\right|^2 \delta(f)}{T_s}$$

This can be normalized as described in section 1.2 on page 2 to give the following expression for the normalized PSD of a stationary symbol RP:

$$\left|\hat{X}\left(\hat{f}\right)\right|^2 = \frac{\mathcal{F}_{\tau \to \hat{f}}\left\{\mathbb{C}_{XX}[\tau]\right\}\left(\hat{f}\right) + \left|\mathbb{E}\left[\widetilde{X}\right]\right|^2 \delta\left(\hat{f}\right)}{\mathbb{E}\left[\left|\widetilde{X}\right|^2\right]} \tag{2.8}$$

Similarly, the PSD of a weakly cyclostationary RP with a period of N can be reduced to:

$$\left|\hat{X}\left(\hat{f}\right)\right|^2 = \frac{\sum_{n=1}^{N}\left(\mathcal{F}_{\tau \to \hat{f}}\left\{\mathbb{C}_{XX}[n,\tau]\right\}\left(\hat{f}\right) + \left|\mathbb{E}\left[\widetilde{X}\right]\right|^2 \delta\left(\hat{f}\right)\right)}{N\mathbb{E}\left[\left|\widetilde{X}\right|^2\right]} \tag{2.9}$$

which is ultimately just the average of N different weakly stationary PSDs.

A few observations with regard to equation (2.8) are worth noting. First, the $|\mathbb{E}[\widetilde{X}]|^2\delta(\hat{f})$ term is an impulse at $\hat{f} = 0$ that represents the DC power

of the symbol stream. If $\mathbb{E}[\widetilde{X}] = 0$, the equation reduces to the discrete-time Fourier transform of the autocorrelation:

$$\left|\hat{X}\left(\hat{f}\right)\right|^2 = \frac{\mathcal{F}_{\tau\to\hat{f}}\left\{\mathbb{C}_{XX}[\tau]\right\}\left(\hat{f}\right)}{\mathbb{C}_{XX}[0]} = \mathcal{F}_{\tau\to\hat{f}}\left\{\mathbb{R}_{XX}[\tau]\right\}\left(\hat{f}\right)$$

It can therefore be concluded that the PSD of a symbol RP is completely defined by the first and second order statistics of the RP.

So how does a balanced code that controls the RDS allow for spectral shaping? It turns out that it can be challenging to control the second order statistics of a symbol RP directly. If, however, one controls the first order statistics of the RDS, the second order statistics of the symbol RP are affected indirectly.

Consider the PSD of the Manchester code discussed in section 2.2 on page 10. Looking at table 2.2 on page 12 a few properties are evident.

1. Regardless of the source statistics p, there will always be an equal number of 1s and 0s. It can therefore be concluded that no impulse power exists at DC and that $\mathbb{E}[\widetilde{X}] = 0$.

   $$\mathbb{E}\left[\widetilde{X}\right] = (1-p)(-1+1) + p(1-1) = 0$$

2. Again regardless of the source statistics p, the energy per mapped symbol $\mathbb{E}[|\widetilde{X}|^2]$ remains constant.

   $$\mathbb{E}\left[\left|\widetilde{X}\right|^2\right] = (1-p)\left(\frac{(-1)^2}{2} + \frac{1^2}{2}\right) + p\left(\frac{1^2}{2} + \frac{(-1)^2}{2}\right) = 1$$

3. The stationarity, like most constrained sequence codes, is cyclic with a period equal to the length of the codeword, in this case 2.

Property 1 is a basic requirement of all balanced codes [4, p. 195] and with property 2 the PSD can be expressed purely in terms of the cyclic autocovariance from equation (2.9) on the preceding page. From property 3 the

$\tau =$	$-\infty$	$\cdots$	-2	-1	0	1	2	$\cdots$	∞
$\mathbb{C}[1,\tau] =$	0	$\cdots$	0	-1	1	0	0	$\cdots$	0
$\mathbb{C}[2,\tau] =$	0	$\cdots$	0	0	1	-1	0	$\cdots$	0

Table 2.3: Cyclostationary autocovariance data for Manchester coding.

autocovariance can be established from table 2.2 on page 12 though inspection, as shown in table 2.3.

$$\begin{aligned}\left|\hat{X}\left(\hat{f}\right)\right|^2 &= \frac{1}{2}\left(\mathcal{F}_{\tau\to\hat{f}}\left\{\mathbb{C}_{XX}[1,\tau]\right\}\left(\hat{f}\right) + \mathcal{F}_{\tau\to\hat{f}}\left\{\mathbb{C}_{XX}[2,\tau]\right\}\left(\hat{f}\right)\right) \\ &= \frac{1}{2}\left(\left[(-1)e^{-2\pi\hat{f}(-1)} + (1)e^{-2\pi\hat{f}(0)}\right]\right. \\ &\qquad \left. + \left[(1)e^{-2\pi\hat{f}(0)} + (-1)e^{-2\pi\hat{f}(1)}\right]\right) \\ &= 1 - \cos\left(2\pi\hat{f}\right) \end{aligned} \tag{2.10}$$

Equation (2.10) defines the normalized PSD for a Manchester encoded symbol RP. Figure 2.8 on the following page plots this normalized PSD while figure 2.9 on page 22 shows the resulting normalized PSD when this symbol RP is represented with a raised-cosine pulse shape. Contrast this to the normalized PSD for a completely uncorrelated symbol RP. Since the autocovariance of an uncorrelated symbol RP is simply an impulse function at $\tau = 0$ with magnitude equal to the RP variance, the continuous portion of the normalized PSD is a constant:

$$\begin{aligned}&\mathbb{C}_{XX}[\tau] = \sigma_X^2\delta[\tau] \\ &\left|\hat{X}\left(\hat{f}\right)\right|^2 = \frac{\sigma_X^2 + \left|\mathbb{E}\left[\widetilde{X}\right]\right|^2\delta\left(\hat{f}\right)}{\mathbb{E}\left[\left|\widetilde{X}\right|^2\right]} = 1 + \frac{\left|\mathbb{E}\left[\widetilde{X}\right]\right|^2}{\mathbb{E}\left[\left|\widetilde{X}\right|^2\right]}\left(\delta\left(\hat{f}\right) - 1\right)\end{aligned} \tag{2.11}$$

The uncorrelated symbol RP will always have a perfectly flat PSD possibly with an impulse at $\hat{f} = 0$ representing any DC bias. This flatness confirms

that any deviation from a constant in the PSD of a signal resulting from an uncorrelated symbol RP is completely dictated by the spectrum of the pulse. However, any symbol RP with a non-zero autocovariance outside $\tau = 0$ will generate a normalized spectral shape that is both periodic with a period of $\hat{f} = 1$ and symmetric about $\hat{f} = 0$, just as in equation (2.10) on the preceding page and as shown in figure 2.8.

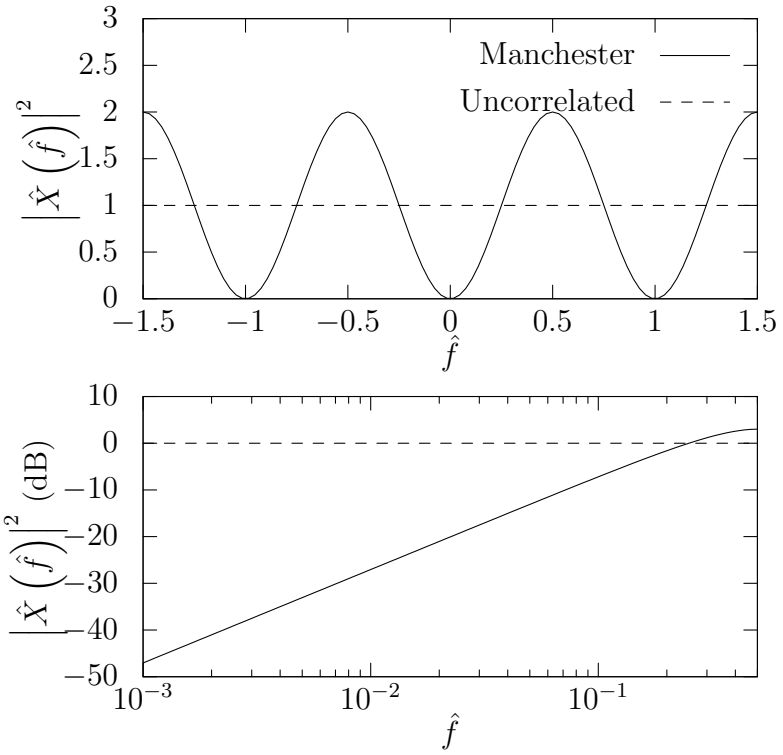

Figure 2.8: Normalized PSD for Manchester coding contrasted with an uncorrelated symbol RP with no DC bias. The scales of the top curve are linear while those for the bottom curve are logarithmic.

When discussing spectral shaping in the context of balanced codes, the

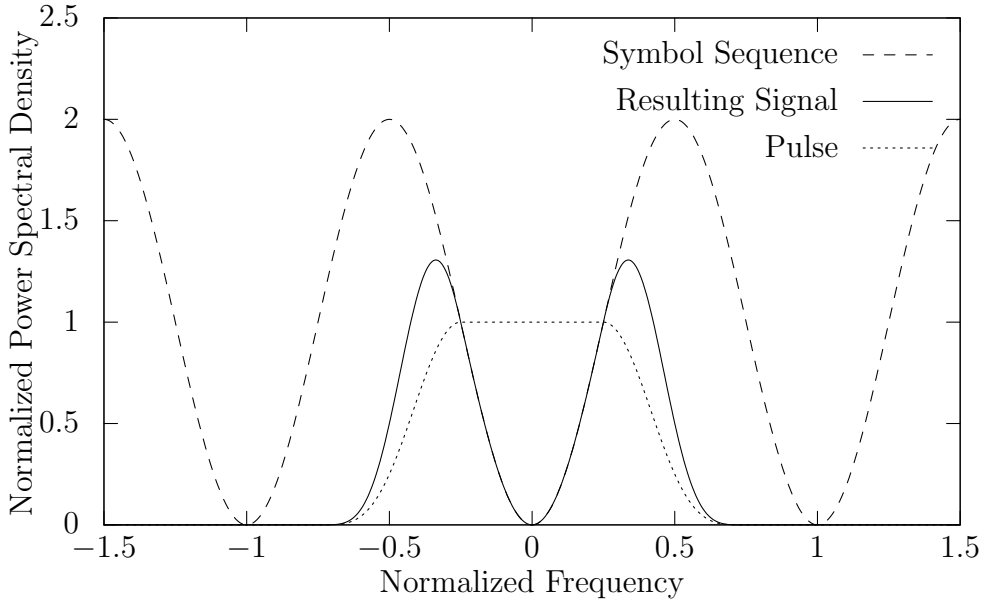

Figure 2.9: Resulting spectrum when a Manchester coded symbol RP is transmitted with raised-cosine pulse shaped with $\beta = 0.5$.

shaping that takes place is the introduction of a null around $\hat{f} = 0$. The width of this null is a measure of the performance of the code and it is directly controlled by the first-order statistics of the RDS, specifically $\mathbb{E}[\Phi^2]$ [4, p. 195]. A balanced code that more effectively minimizes $\mathbb{E}[\Phi^2]$ will ensure less signal power exists near $\hat{f} = 0$. Since the RDS mean is often forced to zero by a balanced code, much of the literature assumes that $\mathbb{V}[\Phi] = \mathbb{E}[\Phi^2]$. Due to both the periodicity and symmetry about $\hat{f} = 0$ of symbol RP PSDs, a logarithmic comparison, as in the lower plot in figure 2.8 on the preceding page, is often preferred. All balanced codes that minimize $\mathbb{E}[\Phi^2]$ will roll-off in the "stop band" at $20\,{}^{\mathrm{dB}}/_{\mathrm{dec}}$ [4]. Note that these codes are often referred to as *first order* spectral null codes. Higher order nulls can be achieved through alternate means. For any arbitrary $X_k \in \mathrm{GF}(2^1), \widetilde{X}_k \in \{-1, 1\}$ symbol RP,

$\mathbb{E}[\Phi^2]$ is lower bounded at $1/2$ [4]. Thus the Manchester code and its PSD establish the best performance for RDS limiting balanced codes. Any such codes with a code rate lower than $1/2$, which is the code rate of the Manchester code, cannot realize further minimization of $\mathbb{E}[\Phi^2]$. Therefore the challenge in RDS limiting balanced codes is realizing a spectral null as close to that of a Manchester code while maximizing the code rate in the range $(1/2, 1)$.

The question still remains as to *why* one would seek to spectrally shape a signal through channel coding. There are numerous answers to this and they are specific to their applications but, like all constrained sequence codes, it is about meeting physical constraints placed on the channel during the design process. A common application lies in baseband signal transmission. Implementation of these systems is greatly simplified when they can be used with de-coupling capacitors or isolating transformers, but designing these components so that they don't attenuate or distort near DC can be complicated and expensive. Balanced coding makes the process trivial. In the realm of passband signals other possibilities present themselves. A null at the carrier frequency allows other auxiliary signals to be inserted at or near this frequency. This might include a low-power pilot tone for carrier recovery or low speed signalling or control data. Beyond that, this null results in I/Q imbalances becoming trivial to filter out along with noise near the carrier.

2.3 Guided Scrambling

So far all the constrained sequence codes discussed in this document involve the use of state dependent tables to map source words to codewords. *Guided scrambling* (GS) offers a different approach [9]. This technique still maps source words to codewords, but the codewords are generated dynamically as source words arrive. This precludes the need for any mapping tables that can become prohibitively large for long codewords. More specifically, the

technique generates a set of candidate codewords as source words arrive and then selects an appropriate word from this set based on a *selection method*.

Prior to delving into the details of GS, a discussion of scrambling itself is warranted. Scrambling, in this context, is a very common technique used in communication systems where a pseudorandom data source is used to introduce the appearance of randomness into a digital symbol sequence. The purpose of this randomness is not, as is often assumed, information security but in ensuring the sequence has sufficient entropy to meet channel constraints. Since most constrained sequence codes simply map source words directly to codewords through a state dependent mapping table, their effectiveness depends on an assumption of equiprobable source words. Because most source data is not actually maxentropic, scramblers are often used to help compensate for probabilistic biases in the source data.

Assume a source sequence $\mathbf{Z}$ and a pseudorandom sequence $\mathbf{\Lambda}$ both with symbols from the GF(2^M) field. In one often used scrambling technique, the scrambled sequence is generated as:

$$\mathbf{X} = \mathbf{Z} \oplus \mathbf{\Lambda}$$

where $\oplus$ denotes symbol-by-symbol addition according to the rules of GF(2^M) addition. The original sequence can be recovered upon reception:

$$\mathbf{Z} = \mathbf{X} \ominus \mathbf{\Lambda}$$

where $\ominus$ denotes symbol-by-symbol subtraction according to the rules of GF(2^M) subtraction. The sequence $\mathbf{X}$ should *hopefully* contain the entropy required by any downstream constrained sequence encoders to perform as expected. The word "hopefully" is emphasized because the scrambling process cannot actually guarantee anything about the statistics of $\mathbf{X}$ since it does not add any redundancy. Assuming $\mathbf{\Lambda}$ is non-periodic the above scrambling system might work fairly well. The reality, however, is that pseudorandom

sequence generators are implemented as some variant of a feedback shift register, and as a result $\mathbf{\Lambda}$ is periodic. If, however, the source sequence can influence the operation of the feedback loop, this periodicity can be disrupted.

One such method used for scrambling is via Galois Field polynomial division. First establish a GF(2^M) divisor polynomial $D(\alpha)$ of degree L.

$$D(\alpha) = \sum_{l=0}^{L} d_l \alpha^l = d_L \alpha^L + \cdots + d_l \alpha^l + \cdots + d_1 \alpha^1 + d_0 \alpha^0$$

The input input sequence $\mathbf{Z}$ can be regarded as a series of source words $\mathbf{S}_i$ of length N:

$$\begin{aligned}
\mathbf{Z} &= \{z_K, \ldots, z_k, \ldots, z_1, z_0\} \\
&= \{\mathbf{S}_I, \mathbf{S}_{I-1}, \ldots, \mathbf{S}_i, \ldots, \mathbf{S}_1, \mathbf{S}_0\} \\
\mathbf{S}_i &= \{s_{i,N-1}, \ldots, s_{i,n}, \ldots, s_{i,1}, s_{i,0}\} \\
s_{i,n} &= z_{iN+n}
\end{aligned}$$

For each source word $\mathbf{S}_i$, form a polynomial $S_i(\alpha)$.

$$S_i(\alpha) = \sum_{n=0}^{N-1} s_{i,n} \alpha^n = s_{i,N-1} \alpha^{N-1} + \cdots + s_{i,n} \alpha^n + \cdots + s_{i,0} \alpha^0$$

Now perform polynomial division to obtain:

$$\begin{aligned}
C_i(\alpha) &= \mathcal{Q}_{D(\alpha)}[S_i(\alpha) \cdot \alpha^L + R_{i-1}(\alpha) \cdot \alpha^N] \\
R_i(\alpha) &= \mathcal{R}_{D(\alpha)}[S_i(\alpha) \cdot \alpha^L + R_{i-1}(\alpha) \cdot \alpha^N]
\end{aligned} \tag{2.12}$$

where $\mathcal{Q}$ and $\mathcal{R}$ denote, respectively, evaluation of quotient and remainder of their argument divided by their subscripted polynomial. The scrambled codeword $\mathbf{C}_i$ is extracted from the coefficients of the polynomial $C_i(\alpha)$. The

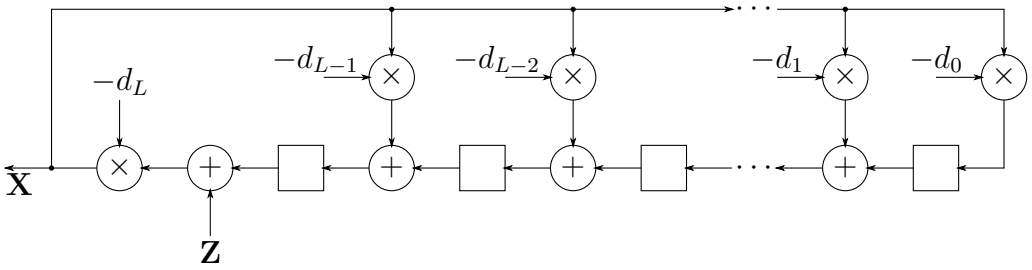

Figure 2.10: GF(2^M) polynomial division scrambler

output sequence $\mathbf{X}$ is the concatenation of the codewords.

$$C_i(\alpha) = \sum_{n=0}^{N-1} c_{i,n}\alpha^n = c_{i,N-1}\alpha^{N-1} + \cdots + c_{i,n}\alpha^n + \cdots + c_{i,0}\alpha^0$$

$$\mathbf{C}_i = \{c_{i,N-1}, \ldots, c_{i,n}, \ldots, c_{i,1}, c_{i,0}\}$$

$$\mathbf{X} = \{\mathbf{C}_I, \mathbf{C}_{I-1}, \ldots, \mathbf{C}_i, \ldots, \mathbf{C}_1, \mathbf{S}_0\}$$

The polynomial $R_i(\alpha)$ is the remainder of the division and its recursive use provides a feedback loop. So long as $D(\alpha)$ is a primitive polynomial, when $S_i(\alpha) = 0$ the scrambler will continue outputting distinct and loosely correlated codewords, repeating with a period $2^{ML} - 1$. In this instance the initial remainder $R_{-1}(\alpha)$ is not relevant to the operation of the scrambler and can simply be set to any non-zero value. The process described by equation (2.12) on the previous page can also be accomplished with the sequential logic shown in figure 2.10. Notice the standard linear-feedback register structure modified with the input sequence now *inside* the feedback loop.

Just as the scrambling process can be described in terms of Galois field polynomial division, the descrambling process can interpreted as multiplication of the codeword by the scrambling polynomial, and removal of the L least significant symbols, a process that can be written as:

$$S_i(\alpha) = \mathcal{Q}_{\alpha^L}[C_i(\alpha)D(\alpha) - R_{i-1}(\alpha) \cdot \alpha^N]$$

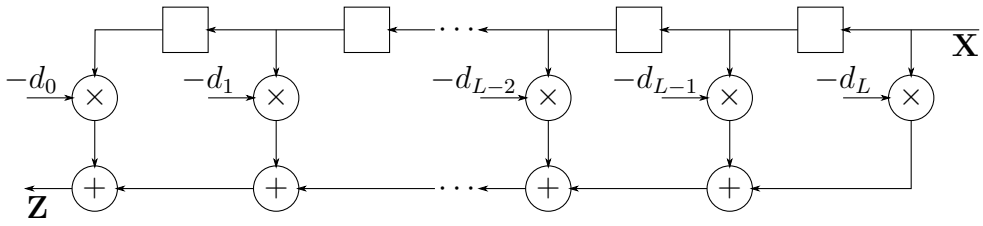

Figure 2.11: GF(2^M) polynomial multiplication descrambler

$$R_i(\alpha) = \mathcal{R}_{\alpha^L}[C_i(\alpha)D(\alpha) - R_{i-1}(\alpha) \cdot \alpha^N] \tag{2.13}$$

Note that, unlike the scrambling process, the remainder polynomial $R_i(\alpha)$ in equation (2.13) has no dependency on $R_{i-1}(\alpha)$ so long as $N \geq L$. In other words, the inversion of this scrambling process converts the feedback nature of the remainder polynomial into a feed forward operation. Although source words affect all future codewords when scrambling, codewords only affect both the current and subsequent source word when descrambling. As a result, the distance by which detection errors are propagated forward by the descrambling process is bounded. Visually, the feed forward nature of the descrambling process is evident in figure 2.11.

It is this polynomial division-based scrambling technique that is used in guided scrambling. Instead of a single scrambler, however, GS employs multiple parallel scramblers to build a *set* of codeword candidates. The first step involves mapping each source word $\mathbf{S}_i$ to a set of *augmented* source words $\mathbf{S}'_i$.

$$\mathbf{S}'_i = \begin{bmatrix} 0 & 0 & \cdots & 0 & s_{i,N-1} & \cdots & s_{i,1} & s_{i,0} \\ 0 & 0 & \cdots & 1 & s_{i,N-1} & \cdots & s_{i,1} & s_{i,0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 2^M-1 & 2^M-1 & \cdots & 2^M-1 & s_{i,N-1} & \cdots & s_{i,1} & s_{i,0} \end{bmatrix}$$

The rows of the set $\mathbf{S}'_{i,j}$ are built by prefixing the source word $\mathbf{S}_i$ with all

possible permutations of symbols from GF(2^M) in an augmenting sequence of length A to give 2^{AM} rows to the set. Each row $\mathbf{S}'_{i,j}$ is then mapped to a polynomial $S'_{i,j}(\alpha)$ and scrambled into a candidate codeword polynomial $C'_{i,j}(\alpha)$ just as with equation (2.12) on page 25.

$$C'_{i,j}(\alpha) = \mathcal{Q}_{D(\alpha)}[S'_{i,j}(\alpha) \cdot \alpha^L + R_{i-1}(\alpha) \cdot \alpha^N]$$
$$R'_{i,j}(\alpha) = \mathcal{R}_{D(\alpha)}[S'_{i,j}(\alpha) \cdot \alpha^L + R_{i-1}(\alpha) \cdot \alpha^N]$$

The set of codeword candidates is then built from the coefficients of all candidate codeword polynomials.

$$\mathbf{C}'_i = \begin{bmatrix} \mathbf{C}'_{i,1} \\ \mathbf{C}'_{i,2} \\ \vdots \\ \mathbf{C}'_{i,j} \\ \vdots \\ \mathbf{C}'_{i,2^{AM}} \end{bmatrix}$$

From the set $\mathbf{C}'_i$, the codeword $\mathbf{C}_i = \mathbf{C}'_{i,J}$ is selected that best meets the constraints prescribed for the channel. The remainder polynomial $R_i(\alpha) = R'_{i,J}(\alpha)$, used in the generation of the next codeword candidate set, is set to that which was generated along with the selected codeword $\mathbf{C}'_{i,J}$.

To recover the original source word during descrambling, the received codeword is multiplied by $D(\alpha)$, and the augmenting sequence is discarded without regard to its value. Since A redundant symbols are being added into each source word of length N, the code rate is $R = \frac{N}{N+A}$.

The mechanism used to decide on the best codeword from the candidate set is called the *selection method*. Much of the GS research has revolved around developing these methods. The original paper [9] focused on selecting codewords that contained the most transitions thereby minimizing run lengths in symbol sequences, and also considered RDS balanced codes

through selection of the codeword with the smallest RDS magnitude at the end of the codeword (WRDS). The authors of [10] recommend minimizing the squared weight (MSW) of the RDS over the codeword. In other words, given a codeword length of $N + A$, the metric associated with each codeword candidate $\mathbf{C}'_{i,j}$ is defined by equation (2.14) where $\phi'_{i,j,n}$ is the RDS after the n^{th} symbol of the j^{th} codeword candidate of the set $\mathbf{C}'_i$.

$$\text{SW}_{i,j} = \sum_{n=1}^{N+A} \left|\phi'_{i,j,n}\right|^2 \tag{2.14}$$

Since RDS balanced codes seek to minimize $\mathbb{E}[\Phi^2]$, the MSW selection method seems fairly intuitive and has been shown to offer very good performance when compared with other RDS-based selection methods [10]. For the remainder of this document any mention of balanced RDS GS coding will assume MSW selection.

Care must also be used in deciding on the scrambling polynomial $D(\alpha)$. By observation of figure 2.11 on page 27 it is evident that errors in detecting $\mathbf{X}$ will continue to affect the output $\mathbf{Z}$ for L subsequent symbols. Thus unlike traditional block codes, the *error multiplication* is dictated not by the codeword length, but the scrambling polynomial's length and number of non-zero terms. Specifically, the number of non-zero terms in the scrambling polynomial dictates how many errors in $\mathbf{Z}$ will be generated for every error in $\mathbf{X}$. The window length in which these errors are dispersed is determined by the degree of the scrambling polynomial L. Therefore to reduce the size of the error multiplication window in $\mathbf{Z}$, shorter scrambling polynomials are desired. Unfortunately, short scrambling polynomials also result in shorter periods for the feedback shift register in the scrambler from figure 2.10 on page 26. Deciding on the degree of the scrambling polynomial is therefore a trade off between the error multiplication window size and feedback shift register period. The degree is, of course, not the only consideration to be made. Choosing a primitive polynomial for the scrambler ensures that the

selection set codewords are very loosely correlated; this has been argued to be ideal for MSW selection [10]. For WRDS selection [11] recommends scramblers that have the factor $\alpha+1$ as this ensures every codeword candidate in the selection set has its complement in the set as well. Since this document will generally assume MSW, $D(\alpha)$ will be assumed primitive unless stated otherwise and its quantity of non-zero terms will be minimized.

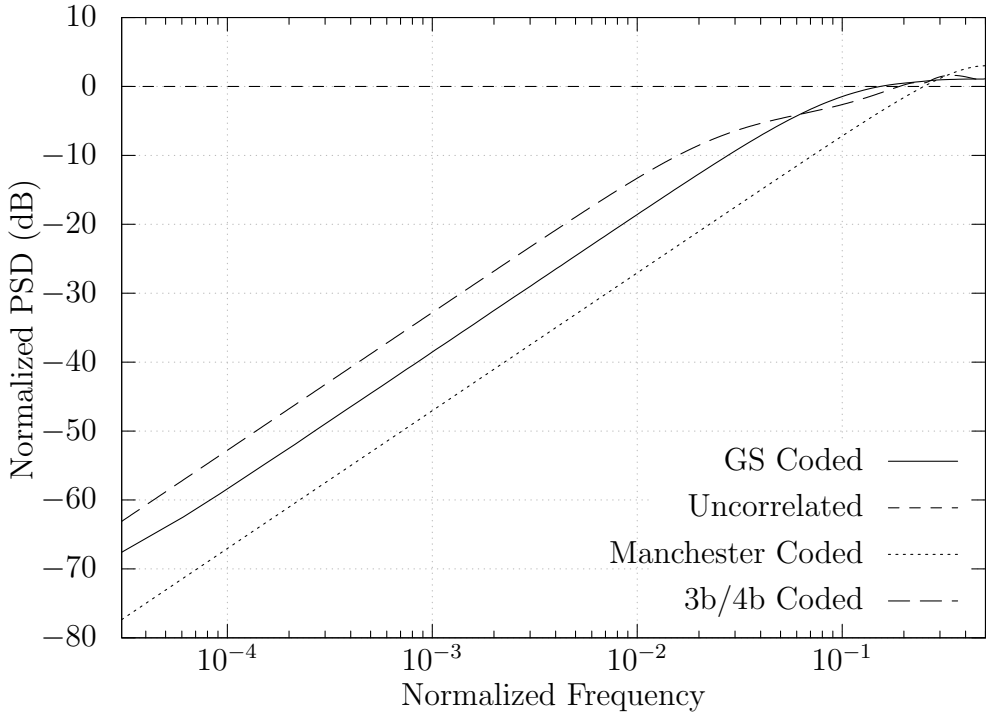

Figure 2.12: Normalized power spectral density for an $x_k \in \mathrm{GF}(2^1), \tilde{x}_k \in \mathbb{R}$ GS symbol sequence with a codeword length of 24 and 6 augmenting symbols. PSD of Manchester coded and 3b/4b coded sequences are included for purposes of comparison.

Overall the performance of balanced GS codes is very promising in comparison with the more traditional block codes. As can be seen in figure 2.12,

the low-frequency rejection of a GS signal with a codeword length of 24 and 6 augmenting symbols performs significantly better than the industry standard 3b/4b code with the identical rate [12]. Ultimately the price for using GS codes is the increased computational resource requirements on the transmit side. For longer codewords, however, these computational requirements can be dwarfed by the lookup table memory requirements for large block codes. Beyond the spectral performance gains, using GS codes also carry the advantage of maintaining encoded sequence statistics somewhat independent of the source statistics.

2.4 Detection

When modulated digital symbols are transmitted over an actual channel, both noise and distortion affect the received signal values. Receiving these corrupted signal values and deciding what symbol was transmitted is called *detection*.

2.4.1 Maximum Likelihood Detection

The optimum approach to deciding on the value of a transmitted symbol x entirely from the received symbol y is called *maximum likelihood* (ML) detection [7, p. 287]. In this approach the detector chooses the transmitted symbol x that makes the received signal value y most likely; it must choose the x that maximizes $\mathbb{P}[Y = y|x]$. Should the received signal RV Y be defined continuously, the detector chooses x that maximizes the continuous probability distribution (PDF) $f_{Y|X}(y, x)$.

For additive white Gaussian noise (AWGN) channels, detection can be greatly simplified. Assume a transmitted symbol RV X and a received symbol

RV Y where:

$$\begin{aligned} X &\in \mathrm{GF}(2^1) \\ \widetilde{X} &\in \{-d/2, d/2\} \\ N &\sim \mathcal{N}(0, n_0) \\ Y &= \widetilde{X} + N \\ Y|x &\sim \mathcal{N}(\tilde{x}, n_0) \\ f_{Y|X}(y, x) &= \frac{\exp\left(-\frac{(y-\tilde{x})^2}{2n_0}\right)}{\sqrt{2\pi n_0}} \end{aligned}$$

The symbol $x = 0$ is detected when:

$$\begin{aligned} f_{Y|X}(y, 0) &> f_{Y|X}(y, 1) \\ \frac{\exp\left(-\frac{(y-[-d/2])^2}{2n_0}\right)}{\sqrt{2\pi n_0}} &> \frac{\exp\left(-\frac{(y-[d/2])^2}{2n_0}\right)}{\sqrt{2\pi n_0}} \\ (y + d/2)^2 &< (y - d/2)^2 \\ \cancel{y^2} + yd + \frac{\cancel{d^2}}{\cancel{4}} &< \cancel{y^2} - yd + \frac{\cancel{d^2}}{\cancel{4}} \\ y &< 0 \end{aligned}$$

Similarly, $x = 1$ is detected when $y > 0$. Ultimately this establishes a decision boundary at $y = 0$. The probability of erroneous detection can be derived analytically as the probability of receiving all possible signal values y which lie outside the correct decision regions established by the boundary at $y = 0$. These regions of error are the Gaussian tails as seen in figure 2.13 on the next page and give the probability of error as a function of the minimum distance between constellation points d and the single axis noise variance n_0:

$$\begin{aligned} \mathbb{P}[\text{error}] &= \mathbb{P}[(Y > 0 \cap X = 0) \cup (Y < 0 \cap X = 1)] \\ &= \mathbb{P}[X = 0]\mathbb{P}[Y > 0|X = 0] + \mathbb{P}[X = 1]\mathbb{P}[Y < 0|X = 1] \end{aligned}$$

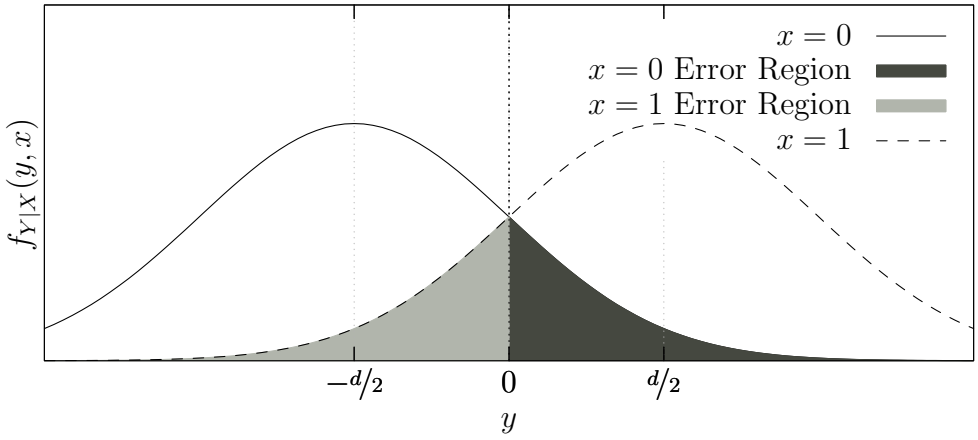

Figure 2.13: GF(2^1) AWGN decision and error regions with $n_0 = d^2$.

$$
\begin{aligned}
&= \mathbb{P}[X = 0] \int_0^\infty f_{Y|X}(y, 0)dy + \mathbb{P}[X = 1] \int_{-\infty}^0 f_{Y|X}(y, 1)dy \\
&= \mathbb{P}[X = 0]Q\left(\frac{d}{2\sqrt{n_0}}\right) + \mathbb{P}[X = 1]Q\left(\frac{d}{2\sqrt{n_0}}\right) \\
&= Q\left(\frac{d}{2\sqrt{n_0}}\right) \qquad (2.15)
\end{aligned}
$$

Similar logic can be applied to the GF(2^2) constellation in figure 2.14 giving these decision regions:

$$
\begin{aligned}
&(\Re(y) > 0 \cap \Im(y) > 0) \Leftrightarrow x = 0 \\
&(\Re(y) < 0 \cap \Im(y) > 0) \Leftrightarrow x = 1 \\
&(\Re(y) > 0 \cap \Im(y) < 0) \Leftrightarrow x = 2 \\
&(\Re(y) < 0 \cap \Im(y) < 0) \Leftrightarrow x = 3
\end{aligned}
$$

To calculate the probability of error it is easier to consider first the probability

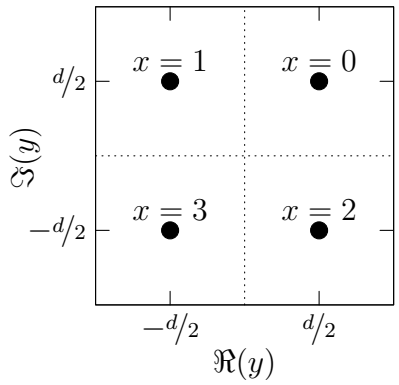

Figure 2.14: GF(2^2) Complex Constellation Pattern

of success.

$$\begin{aligned}
\mathbb{P}[\text{success}] &= \mathbb{P}[(\Re(Y) > 0 \cap \Im(Y) > 0 \cap X = 0) \\
&\qquad \cup (\Re(Y) < 0 \cap \Im(Y) > 0 \cap X = 1) \\
&\qquad \cup (\Re(Y) > 0 \cap \Im(Y) < 0 \cap X = 2) \\
&\qquad \cup (\Re(Y) < 0 \cap \Im(Y) < 0 \cap X = 3)] \\
&= \mathbb{P}[X = 0]\mathbb{P}[\Re(Y) > 0 \cap \Im(Y) > 0 | X = 0] \\
&\qquad + \mathbb{P}[X = 1]\mathbb{P}[\Re(Y) < 0 \cap \Im(Y) > 0 | X = 1] \\
&\qquad + \mathbb{P}[X = 2]\mathbb{P}[\Re(Y) > 0 \cap \Im(Y) < 0 | X = 2] \\
&\qquad + \mathbb{P}[X = 3]\mathbb{P}[\Re(Y) < 0 \cap \Im(Y) < 0 | X = 3] \\
&= \mathbb{P}[X = 0] \int_0^\infty f_{\Re(Y)|\Re(X)}(y, 0)dy \int_0^\infty f_{\Im(Y)|\Im(X)}(y, 0)dy \\
&\qquad + \mathbb{P}[X = 1] \int_{-\infty}^0 f_{\Re(Y)|\Re(X)}(y, 1)dy \int_0^\infty f_{\Im(Y)|\Im(X)}(y, 1)dy
\end{aligned}$$

$$
\begin{aligned}
&+ \mathbb{P}[X=2] \int_{0}^{\infty} f_{\Re(Y)|\Re(X)}(y,2)dy \int_{-\infty}^{0} f_{\Im(Y)|\Im(X)}(y,2)dy \\
&+ \mathbb{P}[X=3] \int_{-\infty}^{0} f_{\Re(Y)|\Re(X)}(y,3)dy \int_{-\infty}^{0} f_{\Im(Y)|\Im(X)}(y,3)dy \\
&= \mathbb{P}[X=0] \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
&\quad + \mathbb{P}[X=1] \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
&\quad + \mathbb{P}[X=2] \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
&\quad + \mathbb{P}[X=3] \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
&= \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
\mathbb{P}[\text{error}] &= 1 - \mathbb{P}[\text{success}] = 1 - \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2 \\
&= 2Q\left(\frac{d}{2\sqrt{n_0}}\right) - Q^2\left(\frac{d}{2\sqrt{n_0}}\right) \qquad (2.16)
\end{aligned}
$$

Similar strategies can be applied to the GF(2^4) constellation pattern in figure 2.15 on the following page by dividing the points into three separate categories. The first category is in the corners when $x \in \{0, 2, 8, 10\}$ where the probability of error, by inspection, must equal that derived for the GF(2^2) constellation pattern in equation (2.16):

$$\mathbb{P}[\text{error}|x \in \text{corner}] = 2Q\left(\frac{d}{2\sqrt{n_0}}\right) - Q^2\left(\frac{d}{2\sqrt{n_0}}\right) \qquad (2.17)$$

The second category consists of the centre points where $x \in \{5, 7, 13, 15\}$. The starting point is, again, the probability of success:

$$\mathbb{P}[\text{success}|x \in \text{centre}] = \mathbb{P}[0 < \Re(y) < d \cap 0 < \Im(y) < d|x = 13]$$

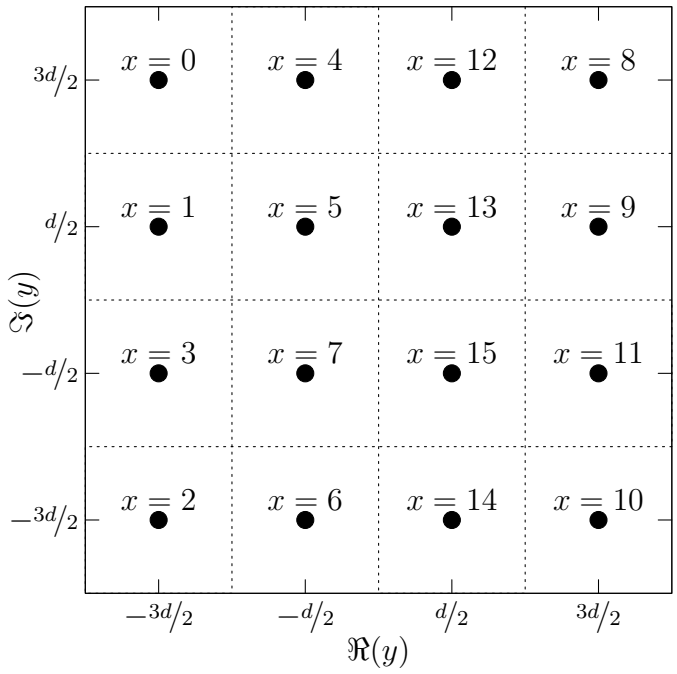

Figure 2.15: GF(2^4) Complex Constellation Pattern

$$= \left(1 - 2Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)^2$$

$$\mathbb{P}[\text{error}|x \in \text{centre}] = 1 - \mathbb{P}[\text{success}|x \in \text{centre}]$$
$$= 4Q\left(\frac{d}{2\sqrt{n_0}}\right) - 4Q^2\left(\frac{d}{2\sqrt{n_0}}\right) \tag{2.18}$$

The third category consists of the edge points ($x \in \{1, 3, 4, 6, 9, 11, 12, 14\}$):

$$\mathbb{P}[\text{success}|x \in \text{edge}] = \mathbb{P}[d < \Re(y) \cap 0 < \Im(y) < d|x = 9]$$
$$= \left(1 - Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)\left(1 - 2Q\left(\frac{d}{2\sqrt{n_0}}\right)\right)$$

$$\mathbb{P}[\text{error}|x \in \text{edge}] = 1 - \mathbb{P}[\text{success}|x \in \text{edge}]$$

$$= 3Q\left(\frac{d}{2\sqrt{n_0}}\right) - 2Q^2\left(\frac{d}{2\sqrt{n_0}}\right) \tag{2.19}$$

Combining equations 2.17, 2.18 and 2.19 the probability of erroneous detection is:

$$\begin{aligned}\mathbb{P}[\text{error}] = \mathbb{P}[X \in \text{corners}] &\left(2Q\left(\frac{d}{2\sqrt{n_0}}\right) - Q^2\left(\frac{d}{2\sqrt{n_0}}\right)\right) \\ + \mathbb{P}[X \in \text{centre}] &\left(4Q\left(\frac{d}{2\sqrt{n_0}}\right) - 4Q^2\left(\frac{d}{2\sqrt{n_0}}\right)\right) \\ + \mathbb{P}[X \in \text{edge}] &\left(3Q\left(\frac{d}{2\sqrt{n_0}}\right) - 2Q^2\left(\frac{d}{2\sqrt{n_0}}\right)\right)\end{aligned} \tag{2.20}$$

Typically the symbols are considered equiprobable which can result in some simplification of equation (2.20) but for the purposes of this work, they will be identified separately as shown above.

2.4.2 Maximum a Posteriori Detection

The optimum approach to deciding on the value of a transmitted symbol x from the received symbol y with consideration for the statistics of X is called *maximum a posteriori* (MAP) detection [7, p. 288]. In contrast with ML detection, the detector chooses the most likely transmitted symbol x given the received value y; it chooses the x that maximizes $\mathbb{P}[X = x|y]$ where:

$$\mathbb{P}[X = x|y] = \frac{f_{Y|X}(y, x)\mathbb{P}[X = x]}{f_Y(y)}$$

Since $f_Y(y)$ is independent of x, the same decision can be reached by simply maximizing $f_{Y|X}(y, x)\mathbb{P}[X = x]$. Note that the probability of the source symbols X impact these decisions.

For the GF(2^1) AWGN example, interesting contrasts can be made to the

ML strategy. This time, the symbol $x = 0$ is detected when:

$$\begin{aligned}
f_{Y|X}(y,0)\mathbb{P}[X=0] &> f_{Y|X}(y,1)\mathbb{P}[X=1] \\
\frac{\exp\left(-\frac{(y-[-d/2])^2}{2n_0}\right)\mathbb{P}[X=0]}{\sqrt{2\pi n_0}} &> \frac{\exp\left(-\frac{(y-[d/2])^2}{2n_0}\right)\mathbb{P}[X=1]}{\sqrt{2\pi n_0}} \\
\frac{-(y+d/2)^2}{2n_0} + \ln\mathbb{P}[X=0] &> \frac{-(y-d/2)^2}{2n_0} + \ln\mathbb{P}[X=1] \\
(y+d/2)^2 - 2n_0\ln\mathbb{P}[X=0] &< (y-d/2)^2 - 2n_0\ln\mathbb{P}[X=1] \\
(y+d/2)^2 &< (y-d/2)^2 + 2n_0\ln\frac{\mathbb{P}[X=0]}{\mathbb{P}[X=1]} \\
\cancel{y^2} + yd + \cancel{\frac{d^2}{4}} &< \cancel{y^2} - yd + \cancel{\frac{d^2}{4}} + 2n_0\ln\frac{\mathbb{P}[X=0]}{\mathbb{P}[X=1]} \\
y &< \frac{n_0}{d}\ln\frac{\mathbb{P}[X=0]}{\mathbb{P}[X=1]}
\end{aligned}$$

Note that with equiprobable input symbols, the decision boundary is at $y = 0$. However, as the symbol probabilities change, instead of the decision boundary resting consistently at $y = 0$ as in the ML case, the boundary $y = \epsilon$ is adjusted to $\frac{n_0}{d}\ln\frac{\mathbb{P}[X=0]}{\mathbb{P}[X=1]}$ as in figure 2.16 on the next page.

The smaller $x = 0$ error region comes at a cost to the larger error region for $x = 1$ but given that $x = 0$ is more probable to begin with, error rates are ultimately reduced. Of course when $x = 0$ and $x = 1$ approach equiprobability, $\epsilon \to 0$ and MAP detection is equivalent to ML detection.

The probability of error can be calculated similarly to the $\mathrm{GF}(2^1)$ case in section 2.4.1.

$$\begin{aligned}
\mathbb{P}[\text{error}] &= \mathbb{P}[(Y > \varepsilon \cap X = 0) \cup (Y < \varepsilon \cap X = 1)] \\
&= \mathbb{P}[X=0]\mathbb{P}[Y > \varepsilon | X = 0] + \mathbb{P}[X=0]\mathbb{P}[Y < \varepsilon | X = 1] \\
&= \mathbb{P}[X=0]\int_{\varepsilon}^{\infty} f_{Y|X}(y,0)dy + \mathbb{P}[X=1]\int_{-\infty}^{\varepsilon} f_{Y|X}(y,1)dy \\
&= \mathbb{P}[X=0]Q\left(\frac{\varepsilon - (-d/2)}{\sqrt{n_0}}\right) + \mathbb{P}[X=1]Q\left(\frac{-\varepsilon - (-d/2)}{\sqrt{n_0}}\right)
\end{aligned}$$

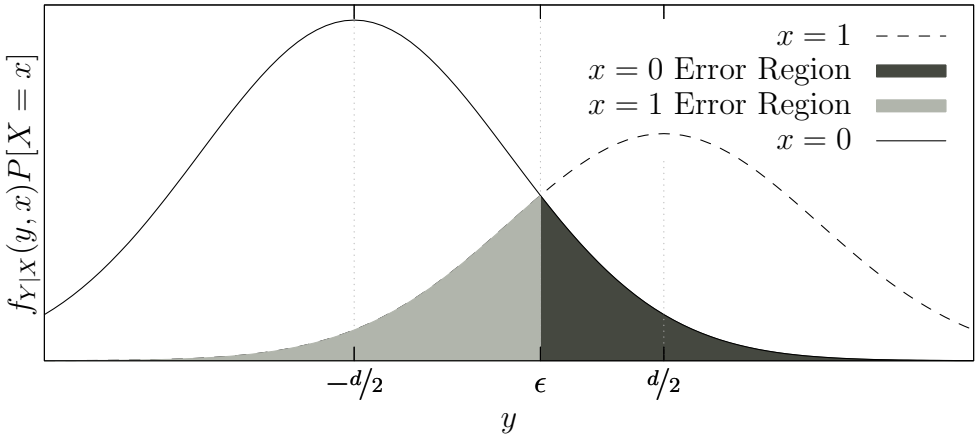

Figure 2.16: GF(2^1) AWGN MAP decision and error regions with $n_0 = d^2, \mathbb{P}[X = 0] = 0.6, \mathbb{P}[X = 1] = 0.4$.

$$= \mathbb{P}[X = 0]Q\left(\frac{d/2 + \varepsilon}{\sqrt{n_0}}\right) + \mathbb{P}[X = 1]Q\left(\frac{d/2 - \varepsilon}{\sqrt{n_0}}\right) \tag{2.21}$$

Unsurprisingly, as $\epsilon \to 0$, equation (2.21) converges to equation (2.15) on page 33. See figure 2.17 on the following page for an error rate comparison of ML versus MAP detection with unequal probabilities.

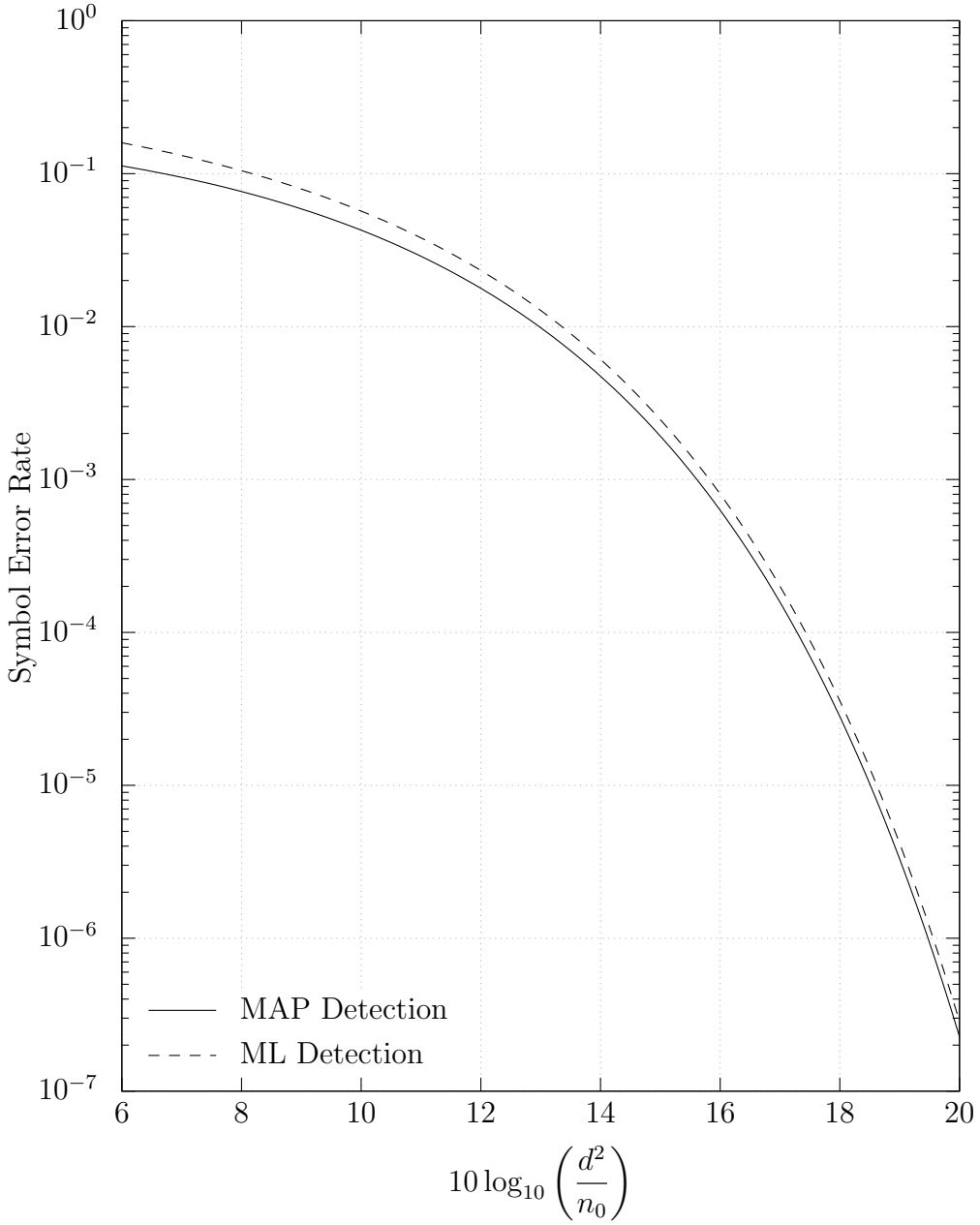

Figure 2.17: GF(2^1) AWGN MAP error rate with $\mathbb{P}[X = 0] = 0.8, \mathbb{P}[X = 1] = 0.2$.

Chapter 3

Principles of Intelligent Detection of GS Coded Sequences

The goal of this research work is to employ an intelligent detection technique to GS signals so as to reduce the resulting error rate on AWGN channels. In order to do so, the premise will be to take advantage of known characteristics of GS coded signals. The focus on GS codes, as opposed to balanced codes in general, is due to the unique property of GS coding that with appropriate parameter selection, the signal statistics can be, for practical purposes, assumed independent of the source data [13]. Although block codes combined with scramblers also reduce dependence on source statistics, they provide no guarantees as certain source symbol sequences may result in statistically undesirable signals.

This chapter outlines principles on which intelligent detection of GS coded sequences can be performed. Section 3.1 develops a statistical model to describe GS coded sequences that offers insight into the possibility for their intelligent detection. Section 3.2 presents an algorithmic approach to the

proposed detection technique while section 3.3 discusses the implementation of this technique.

3.1 Modelling of GS Coded Sequence Statistics

Since GS codes that perform codeword selection using the MSW criteria base their selection entirely on the RDS, the chosen strategy was to model the RDS statistics as opposed to the mapped symbol statistics. For the sake of computational feasibility, the model will seek to satisfy four criteria:

1. Statistics should be cyclostationary. This has been shown in [14] for GS codes.
2. Dependency between RDS values over time should be accurately described by their correlation.
3. It should be memoryless. That is, the model should be accurately described as a first order Markov chain.
4. Complete circular symmetry should exist in the case of complex valued signals.

3.1.1 First Order Statistics

In order for a spectral shaping code to maximize the null width at $\hat{f} = 0$ it must endeavour to minimize $\mathbb{E}[\Phi^2]$ [4, p. 208]. Since most codes attempt to ensure that $\mathbb{E}[\Phi] \to 0$ it is reasonable to approximate $\mathbb{E}[\Phi^2] = \mathbb{V}[\Phi]$. Therefore with a fixed source entropy, to maximize the width of the null at DC the coding process must minimize the variance of the RDS. Minimization of

the variance of a continuous random variable for a fixed amount of information occurs with a Gaussian distribution [3]. RDS values are, however, defined discretely, but the probability mass function should nonetheless be well approximated by a Gaussian curve if the variance of these values is to be minimized. It can therefore be assumed that balanced codes with a wide spectral null have RDS values that can be well approximated as a Gaussian random variable. Given the good performance available with GS, as outlined in section 2.3, it is assumed that the first order statistics of the RDS should follow a circularly-symmetric Gaussian profile. Due to the complexity of GS sequences with moderately long codewords, however, analytic solutions for the relevant statistics are not feasible, and therefore this assumption is examined through simulations with 2×10^9 symbols that were used to generate the RDS probability mass functions (PMF).

For circularly symmetrical RDS statistics, constellation patterns must be chosen that are 2-fold symmetric about the real and imaginary axes. For the purposes of simulation, therefore, the constellation patterns shown in figure 2.13 on page 33, figure 2.14 on page 34, and figure 2.15 on page 36 were used with $d = 2$. Several GS codes were simulated including all possible configurations up to a maximum codeword length of 24, rate of $1/2$ and 4096 scramblers.

In all these simulations, RDS patterns such as that in figure 3.1 on the following page and figure 3.2 on page 45 arise. In these figures, the logarithm of the probability of an RDS value is represented by the size of the circle centred at the RDS value.[1] These figures demonstrate the complete first order independence and symmetry between $\Im(\Phi)$ and $\Re(\Phi)$. As expected, the pattern qualitatively seems to follows a circularly-symmetric complex Gaussian profile.

The circular symmetry allows the PMFs to be collapsed into a single

[1] Averaged over the cyclostationary period

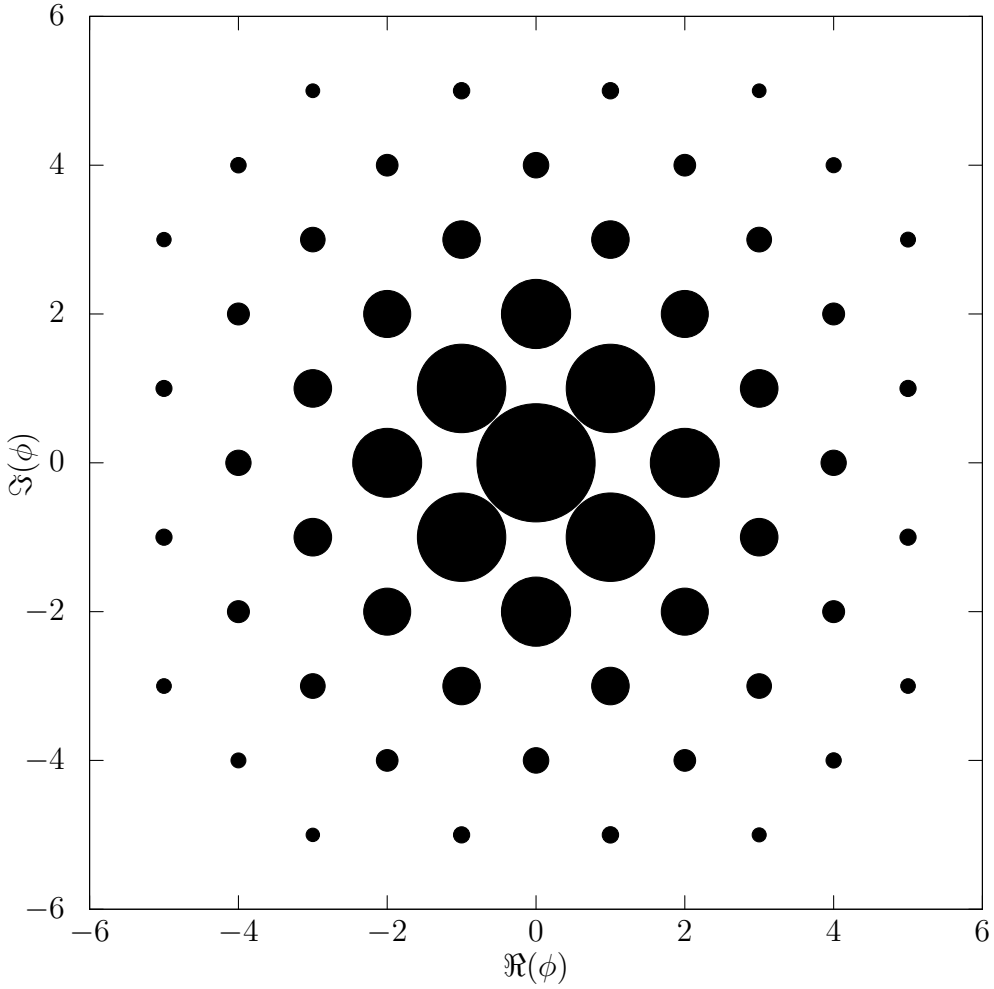

Figure 3.1: Simulated RDS PMF on the complex plane for $x_k \in \mathrm{GF}(2^2), \Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS signals with a codeword length of 12 containing 3 augmenting symbols. Point size is proportional to the logarithm of the probability. Note the first order circular symmetry.

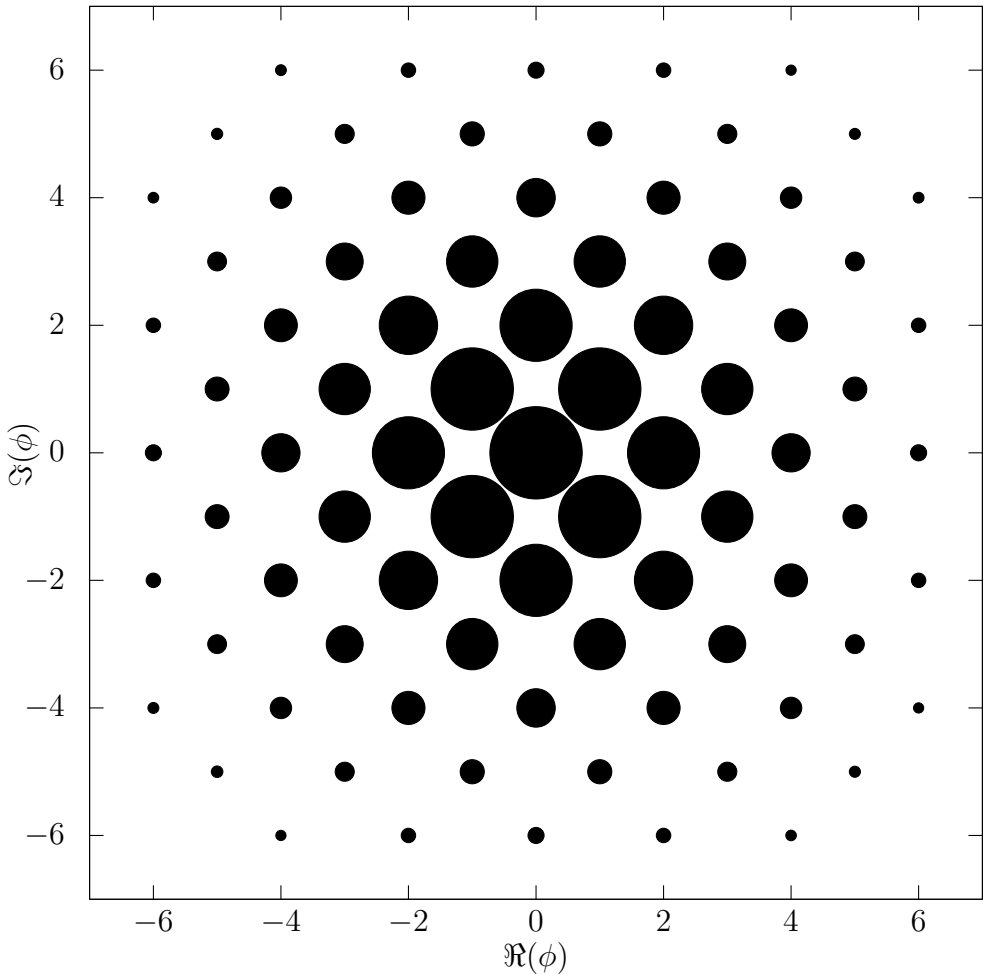

Figure 3.2: Simulated RDS PMF on the complex plane for $x_k \in \mathrm{GF}(2^4), \Re\&\Im(\tilde{x}_k) \in \{-3, -1, 1, 3\}$ GS signals with a codeword length of 12 containing 3 augmenting symbols. Point size is proportional to the logarithm of the probability. Note the first order circular symmetry.

dimension for easier analysis. Figure 3.3 on the following page presents a collection of RDS PMFs for rate ¾ $x_k \in \mathrm{GF}(2^2), \tilde{x}_k \in \mathbb{C}$ GS codes averaged over the cyclostationary period while figure 3.4 on page 48 gives an XY comparison of these approximations. Although appearing more binomial in nature, the distributions are well approximated by a Gaussian curve. It should be noted that the accuracy of the Gaussian approximation is dependent on having larger selection sets, similar to the relationship between spectral performance and size of the selection set.

Table 3.1 on page 49 tabulates the RDS variances for rate ¾ $x_k \in \mathrm{GF}(2^2), \Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS codes with up to six augmenting symbols. It also lists RMS error in the Gaussian approximation, where this RMS error is calculated according to:

$$\varepsilon = \sqrt{\frac{1}{2\overline{\Phi}+1} \sum_{\phi=-\overline{\Phi}}^{\overline{\Phi}} \left(\zeta \exp\left(\frac{-\phi^2}{2\sigma^2}\right) - \mathbb{P}[\Phi = \phi] \right)^2} \tag{3.1}$$

These results reinforce the accuracy of the Gaussian approximation, particularly with larger codeword candidate sets. Similar results can be observed with different fields sizes and code configurations.

Based on these simulation results, it is concluded that, in the first order, a sampled continuity corrected Gaussian curve is a good approximation for the RDS of GS signals with a sufficient number of scramblers. Specifically:

$$\mathbb{P}[\Phi_n = \phi] \approx \zeta_n \exp\left(\frac{-\phi^2}{2\mathbb{C}_{\Phi\Phi}[n,0]}\right) \tag{3.2}$$

where $\mathbb{C}_{\Phi\Phi}[n,0]$ is the variance at the n^{th} position in the cyclostationary period and ζ_n is a continuity correction to ensure the sum of all RDS probabilities is 1.

Given this Gaussian approximation, all dependency between RDS values of GS coded sequences over time should be well approximated by their corre-

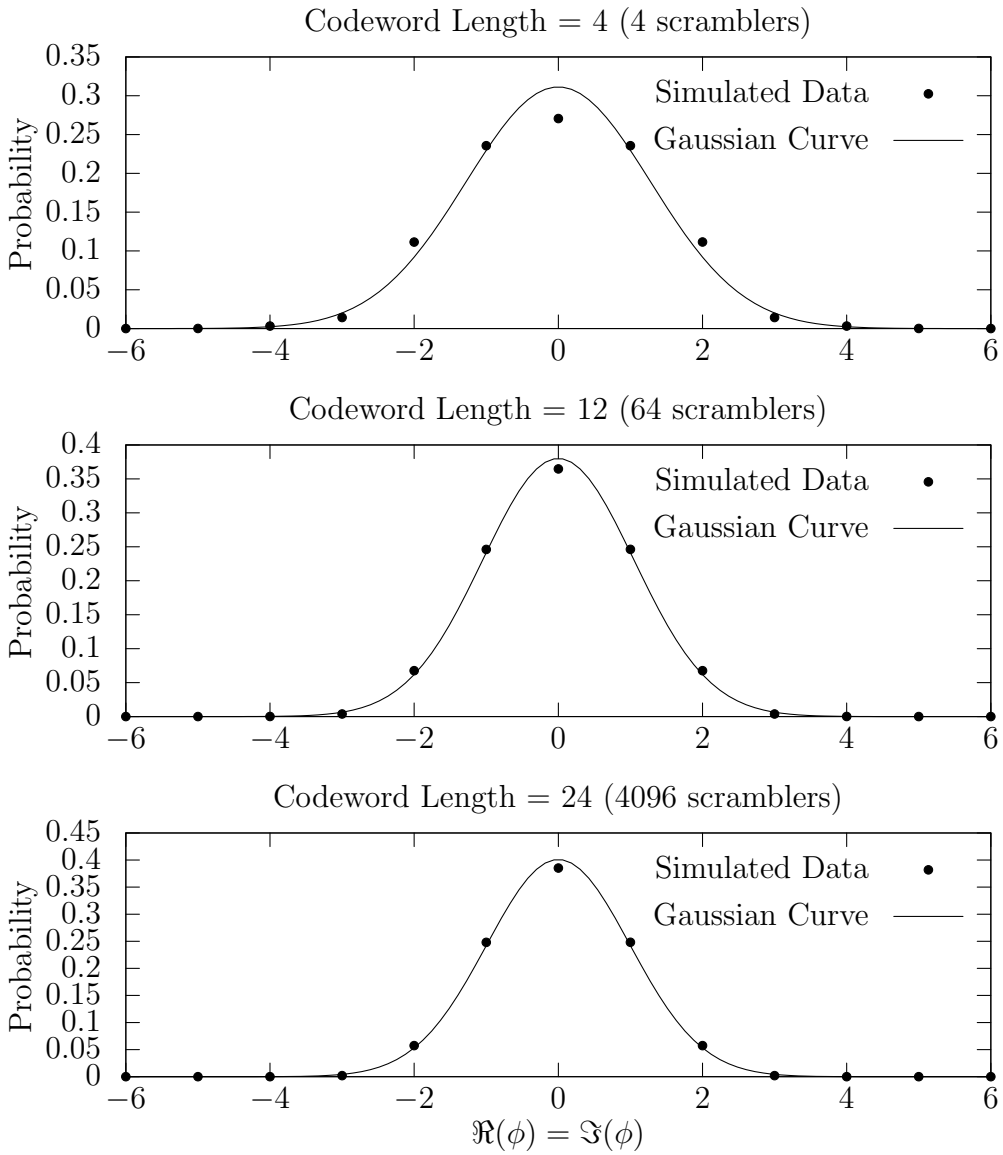

Figure 3.3: PMF for the real or imaginary component of the RDS for $\Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS signals of rate $3/4$. The curve is a continuity corrected Gaussian PDF generated from the variance of the RDS.

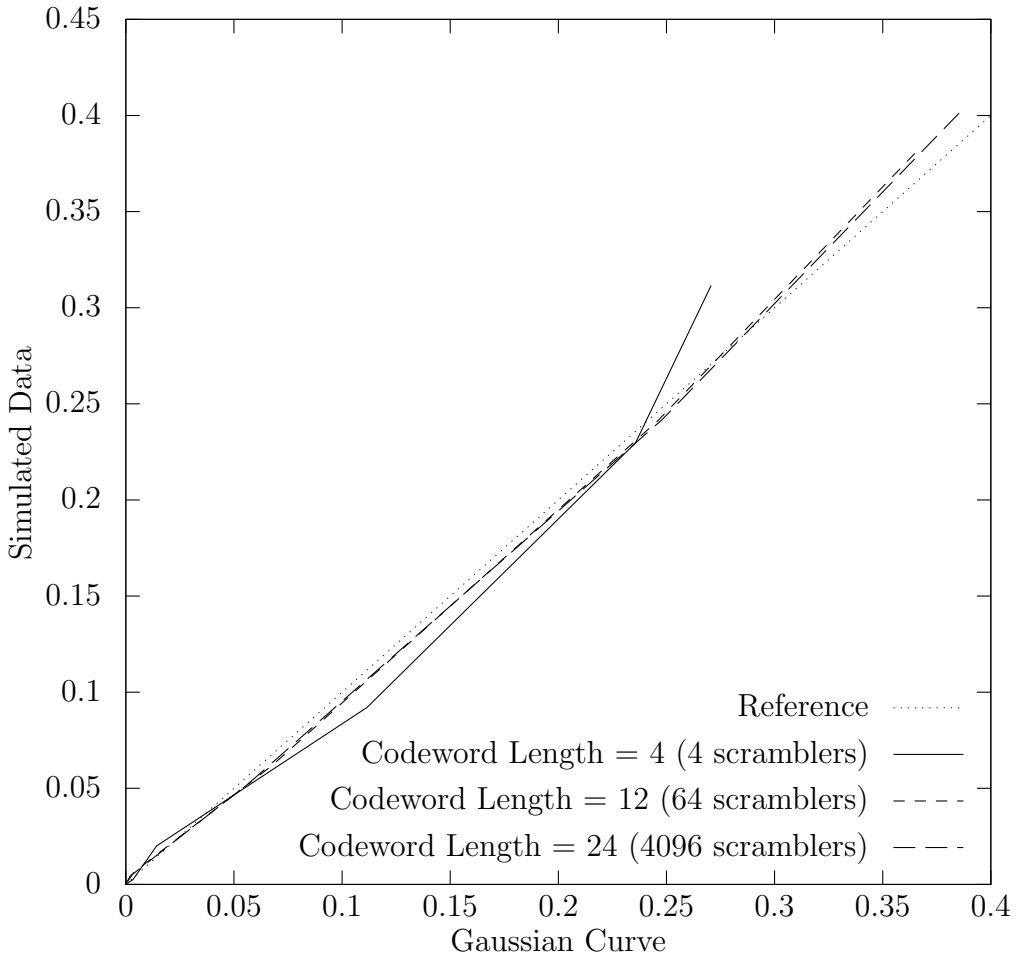

Figure 3.4: XY Gaussian approximation comparison for the real or imaginary component of the RDS for $\Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS signals of rate $^3/_4$.

Codeword Length	Scramblers	RDS Variance	RMS Error
4	4	1.64	1.40×10^{-2}
8	16	1.24	3.08×10^{-3}
12	64	1.10	5.28×10^{-3}
16	256	1.05	5.04×10^{-3}
20	1024	1.01	5.39×10^{-3}
24	4096	0.99	5.36×10^{-3}

Table 3.1: RDS variance and Gaussian approximation error for rate ¾ $\Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS codes. The RMS error is calculated as per equation (3.1) on page 46.

lation. The second order statistics should therefore sufficiently describe the dependency [1, p. 94].

3.1.2 Second Order Statistics

Just as in the first order, so long as constellation patterns are chosen to be 2-fold symmetric about the real and imaginary axes, circular symmetry is preserved in the second order as well. Simulations show that, unsurprisingly, $\mathbb{E}\left[\Re(\Phi_k)\Im(\Phi_{k-\tau})\right] = 0$. Given this symmetry, the autocovariances for the real and imaginary RDS processes can be considered equivalent and analyzed along a single dimension.

Analysis of the second order statistics will first assume the Markovian property:

$$\mathbb{P}[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}] = \mathbb{P}[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}, \ldots, \Phi_0 = \phi_0]$$

The process can then be fully described by the probability of transitioning from one RDS state to another. Assuming an RDS state space truncated at $\pm\overline{\Phi}$, the probabilities are assembled into a cyclostationary transition matrix

$\mathbf{P}_n$ where n signifies the position within the cyclostationary period.

$$\mathbf{P}_n = \begin{bmatrix} P_{n,(-\overline{\Phi})\to(-\overline{\Phi})} & \cdots & P_{n,(-\overline{\Phi})\to\phi_j} & \cdots & P_{n,(-\overline{\Phi})\to(+\overline{\Phi})} \\ P_{n,(1-\overline{\Phi})\to(-\overline{\Phi})} & \cdots & P_{n,(1-\overline{\Phi})\to\phi_j} & \cdots & P_{n,(1-\overline{\Phi})\to(+\overline{\Phi})} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{n,\phi_i\to(-\overline{\Phi})} & \cdots & P_{n,=\phi_i\to\phi_j} & \cdots & P_{n,\phi_i\to(+\overline{\Phi})} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{n,(\overline{\Phi}-1)\to(-\overline{\Phi})} & \cdots & P_{n,(\overline{\Phi}-1)\to\phi_j} & \cdots & P_{n,(\overline{\Phi}-1)\to(+\overline{\Phi})} \\ P_{n,(+\overline{\Phi})\to(-\overline{\Phi})} & \cdots & P_{n,(+\overline{\Phi})\to\phi_j} & \cdots & P_{n,(+\overline{\Phi})\to(+\overline{\Phi})} \end{bmatrix} \tag{3.3}$$

Due to the first order Gaussian nature examined in the previous section, the transition probabilities $P_{n,\phi_i\to\phi_j}$ should also be well approximated by a continuity corrected Gaussian curve.

$$\begin{aligned} P_{n,\phi_i\to\phi_j} &= \mathbb{P}[\Phi_n = \phi_j | \Phi_{n-1} = \phi_i] \\ &= \zeta_{n,i} \begin{cases} \exp\left(\frac{(\phi_j-\mu_{n,i})^2}{2\sigma_n^2}\right), & \phi_j - \phi_i \in \widetilde{X} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3.4}$$

Note that the transition probability can only be nonzero if the mapped symbol, $\tilde{x}$, exists to force the transition. The mean $\mu_{n,i}$ and variance σ_n^2 are conditioned [1, p. 94] using the previous RDS value ϕ_i and the simulated cyclic autocovariance $\mathbb{C}_{\Phi\Phi}[n,\tau]$ with $\tau \in \{0,1\}$.

$$\begin{aligned} \mu_{n,i} &= \frac{\mathbb{C}_{\Phi\Phi}[n,1]}{\mathbb{C}_{\Phi\Phi}[n-1,0]}\phi_i \\ \sigma_n^2 &= \mathbb{C}_{\Phi\Phi}[n,0] - \frac{\mathbb{C}_{\Phi\Phi}[n,1]^2}{\mathbb{C}_{\Phi\Phi}[n-1,0]} \end{aligned}$$

The coefficient $\zeta_{n,i}$ is, again, a continuity correction to ensure that each row of the transition matrix sums to exactly 1.

In order to show that $\mathbf{\Phi}_k$ is, in fact, well approximated as a Markov chain, the autocovariance beyond $\tau = 1$ should be well predicted by the transition matrix. To simplify this analysis, the cyclostationary process $\mathbf{\Phi}_k$ will be

averaged across the length of a codeword into a stationary process. Given a transition matrix $\mathbf{P}$, an RDS state vector $\mathbf{S}$, and a steady state probability vector $\mathbf{P}_0$, the autocovariance should be well approximated as:

$$\begin{aligned}
\mathbf{P} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{P}_n \\
\mathbf{S} &= \left\{-\overline{\Phi},\ldots,0,\ldots,\overline{\Phi}\right\} \\
\mathbf{P}_0 &= \left\{\frac{1}{N}\sum_{n=1}^{N}\mathbb{P}[\Phi_n=\phi]\right\}_{\phi=-\overline{\Phi}}^{\overline{\Phi}} \\
\mathbb{C}_{\Phi\Phi}[\tau] &= \mathbb{E}[\Phi_k\Phi_{k-\tau}] \\
&\approx \mathrm{diag}(\mathbf{P}_0)\times\mathbf{P}^{\tau}\times\mathbf{S}^{\top}\times\mathbf{S}
\end{aligned} \tag{3.5}$$

Figure 3.5 on the next page presents some simulated autocovariances for $x_k \in \mathrm{GF}(2^2), \Re\&\Im(\tilde{x}_k) \in \{-1,1\}$ GS signals of rate ¾ compared to autocovariance data calculated from the transition matrix as per equation (3.5). These curves show that, just as its first order statistics can be well approximated by a continuity corrected Gaussian curve with sufficient scramblers, the RDS can be well approximated as a Markov chain given sufficient scramblers. Table 3.2 on page 53 tabulates the error in the Markovian approximation of the autocovariance evaluated according to:

$$\varepsilon = \sqrt{\frac{1}{T}\sum_{\tau=0}^{T}\left(\mathrm{diag}(\mathbf{P}_0)\times\mathbf{P}^{\tau}\times\mathbf{S}^{\top}\times\mathbf{S}-\mathbb{C}_{\Phi\Phi}[\tau]\right)^2} \tag{3.6}$$

Similar to relationships demonstrated earlier, accuracy of this approximation increases as more scramblers are added. Also included is the correlation between adjacent RDS values. Just as the RDS first order variance decreases as more scramblers are added[2], so does this correlation.

It is, therefore, concluded that the RDS can be well approximated as a cyclostationary sampled Gaussian Markov chain so long as there is a sufficiently

[2]See table 3.1 on page 49.

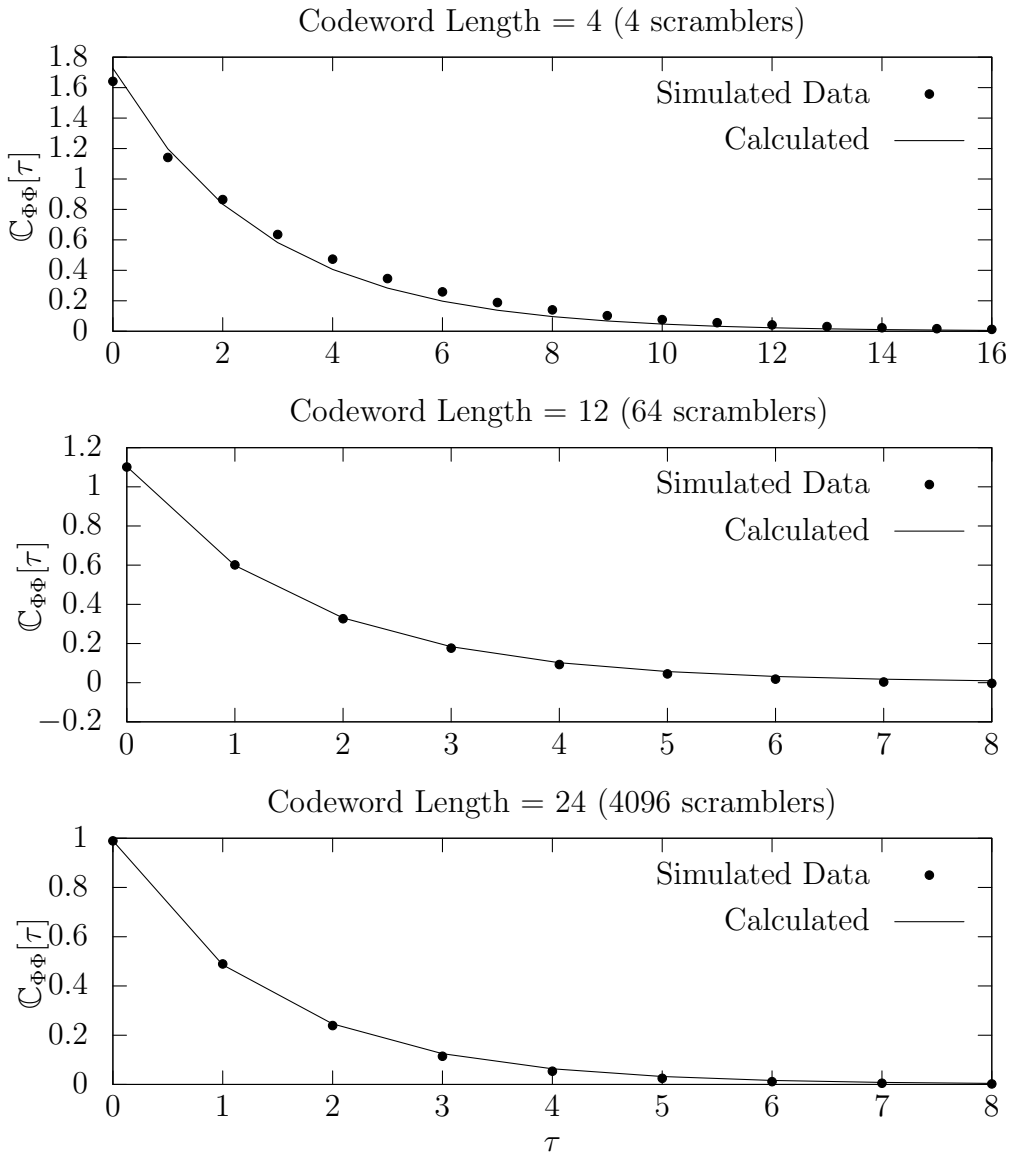

Figure 3.5: Averaged cyclostationary real or imaginary autocovariances for $x_k \in \mathrm{GF}(2^2), \Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS signals of rate 3/4. The curve is calculated from the transition matrix as per equation (3.5) on the preceding page.

Codeword Length	Scramblers	$\mathbb{R}_{\Phi\Phi}[1]$	RMS Error
4	4	0.70	2.32×10^{-2}
8	16	0.60	4.49×10^{-3}
12	64	0.55	4.30×10^{-3}
16	256	0.52	3.05×10^{-3}
20	1024	0.51	2.51×10^{-3}
24	4096	0.49	2.39×10^{-3}

Table 3.2: Autocorrelation ($\tau = 1$) and Markov approximation error for rate ¾ $\Re\&\Im(\tilde{x}_k) \in \{-1, 1\}$ GS codes. The RMS error is calculated as per equation (3.6) on page 51.

large selection set. Given that this requirement also leads to better spectral performance, it is expected that it will be a characteristic of implemented GS codes. Thus the model meets all criteria outlined at the beginning of section 3.1 and can be used as the basis for probabilistic detection.

3.2 Detection

The accuracy of detection of GS coded symbols can be improved with an intelligent detection technique that takes into consideration the statistics of the RDS values. With the model described in section 3.1, RDS values can be conditionally mapped to probability values. MAP detection, as described in section 2.4.2 on page 37, requires choosing a source symbol x_k that maximizes the following probability.

$$\mathbb{P}[X_k = x_k | y_k] = \frac{f_{Y_k|X_k}(y_k, x_k)\mathbb{P}[X_k = x_k]}{f_{Y_k}(y_k)}$$

GS coding, however, does not control the coded data statistics directly but does impact the RDS statistics. However, since the RDS is an outcome of the

coded symbol values, it is assumed that the coded statistics can be accurately described by the RDS statistics.

$$\begin{aligned}\mathbb{P}[X_k = x_k] &= \mathbb{P}[\Phi_k = \phi_{k-1} + \tilde{x}_k | \Phi_{k-1} = \phi_{k-1}] \\ &= \mathbb{P}[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}]\end{aligned}$$

This probability is simply extracted from the matrix described by equation (3.3) on page 50 populated with values from equation (3.4).

The fact that this probability is conditioned by the previous real[3] RDS value makes it necessary to describe it in terms of *sequence* probabilities.

$$\begin{aligned}\mathbf{X} &= \{x_k\}_{k=1}^{K}, x_k \in \mathrm{GF}(2^M), \tilde{x}_k \in \mathbb{R} \\ \mathbf{\Phi} &= \{\phi_k\}_{k=0}^{K} = \{\phi_{k-1} + \tilde{x}_k\}_{k=1}^{K}, \phi_0 = 0, \phi_k \in \mathbb{R} \\ \mathbb{P}[X = \mathbf{X}] = \mathbb{P}[\Phi = \mathbf{\Phi}] &= \prod_{k=1}^{K} \mathbb{P}\left[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}\right]\end{aligned}$$

Therefore the conditional probabilities can be recursively described for sequences.

$$\mathbb{P}[X_k = x_k | y_k] = \frac{f_{Y_k|X_k}(y_k, x_k)\mathbb{P}\left[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}\right]}{f_{Y_k}(y_k)} \tag{3.7}$$

$$\begin{aligned}\mathbf{Y} &= \{y_k\}_{k=1}^{K}, y_k \in \mathbb{R} \\ \mathbb{P}[X = \mathbf{X} | \mathbf{Y}] &= \frac{f_{Y|X}(\mathbf{Y}, \mathbf{X})\mathbb{P}[X = \mathbf{X}]}{f_Y(\mathbf{Y})} \\ &= \prod_{k=1}^{K} \frac{f_{Y_k|X_k}(y_k, x_k)\mathbb{P}\left[\Phi_k = \phi_k | \Phi_{k-1} = \phi_{k-1}\right]}{f_{Y_k}(y_k)}\end{aligned} \tag{3.8}$$

Equation (3.8) again describes a first order Markov chain [1, p. 424]. This Markov chain can be represented as the tree in figure 3.6 on the following page where the RDS, ϕ_k, defines the states and all possible values of x_k

[3]Complex valued signals can be detected as two independent real valued signals so long as constellation patterns are two-fold symmetric.

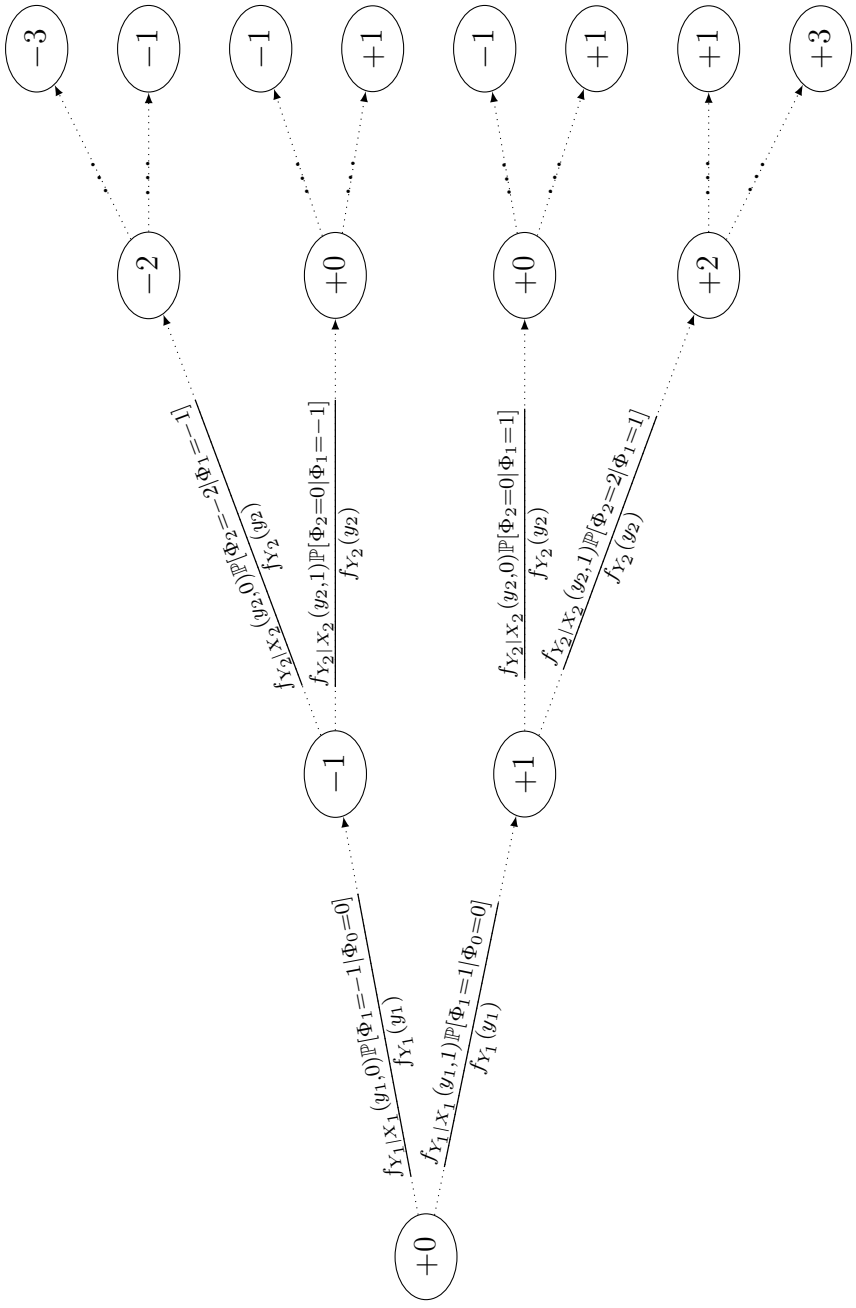

Figure 3.6: $x_k \in \mathrm{GF}(2^1), \tilde{x}_k \in \{-1, 1\}$ GS probability tree with RDS states as the nodes and symbols causing transitions.

force transitions with probabilities considering the observed value, y_k, as per equation (3.7) on page 54. The probability of all possible sequences can be calculated by taking the product of all transition probabilities. The detector should thereby choose the sequence that has the maximum probability. For computational simplicity, however, the best path through the tree can instead be found by determining values that minimize equation (3.9), discarding $f_{Y_k}(y_k)$ as it is common to all paths and performing summations rather than multiplying probabilities.

$$\begin{aligned}\psi_k(\phi_{k-1}, x) &= -\ln\left(f_{Y_k|X_k}(y_k, x)\right) - \ln\left(\mathbb{P}\left[\Phi_k = \phi_{k-1} + \tilde{x} | \Phi_{k-1} = \phi_{k-1}\right]\right) \\ \Psi(\mathbf{X}) &= \sum_{k=1}^{K} \psi_k \left(\phi_0 + \sum_{i=1}^{k-1} \tilde{x}_i, x_k\right) \end{aligned} \tag{3.9}$$

Since RDS states reoccur each time interval the tree can be folded into a Viterbi style trellis [15] as seen in figure 3.7 on the next page with one caveat. Trellis detection assumes a finite number of states whereas [16] shows that the RDS values of MSW selected GS signals are not necessarily hard bounded. This can be rectified by truncating highly improbable RDS states as in figure 3.7 on the following page. Intuition leads to the assumption that this would create an error floor but such has yet to be observed in simulation.

For AWGN channels the path metric $\psi_k(\phi_{k-1}, x)$ can be further simplified by considering $f_{Y_k|X_k}(y_k, x)$.

$$\begin{aligned} f_{Y_k|X_k}(y_k, x) &= \frac{\exp\left(\frac{-|y_k - \tilde{x}|^2}{2n_0}\right)}{\sqrt{2\pi n_0}} \\ \psi_k(\phi_{k-1}, x) &= -\ln\left(f_{Y_k|X_k}(y_k, x)\right) \\ &\qquad - \ln\left(\mathbb{P}\left[\Phi_k = \phi_{k-1} + \tilde{x} | \Phi_{k-1} = \phi_{k-1}\right]\right) \\ &= \frac{|y_k - \tilde{x}|^2}{2n_0} + \ln(\sqrt{2\pi n_0}) \\ &\qquad - \ln\left(\mathbb{P}\left[\Phi_k = \phi_{k-1} + \tilde{x} | \Phi_{k-1} = \phi_{k-1}\right]\right) \end{aligned}$$

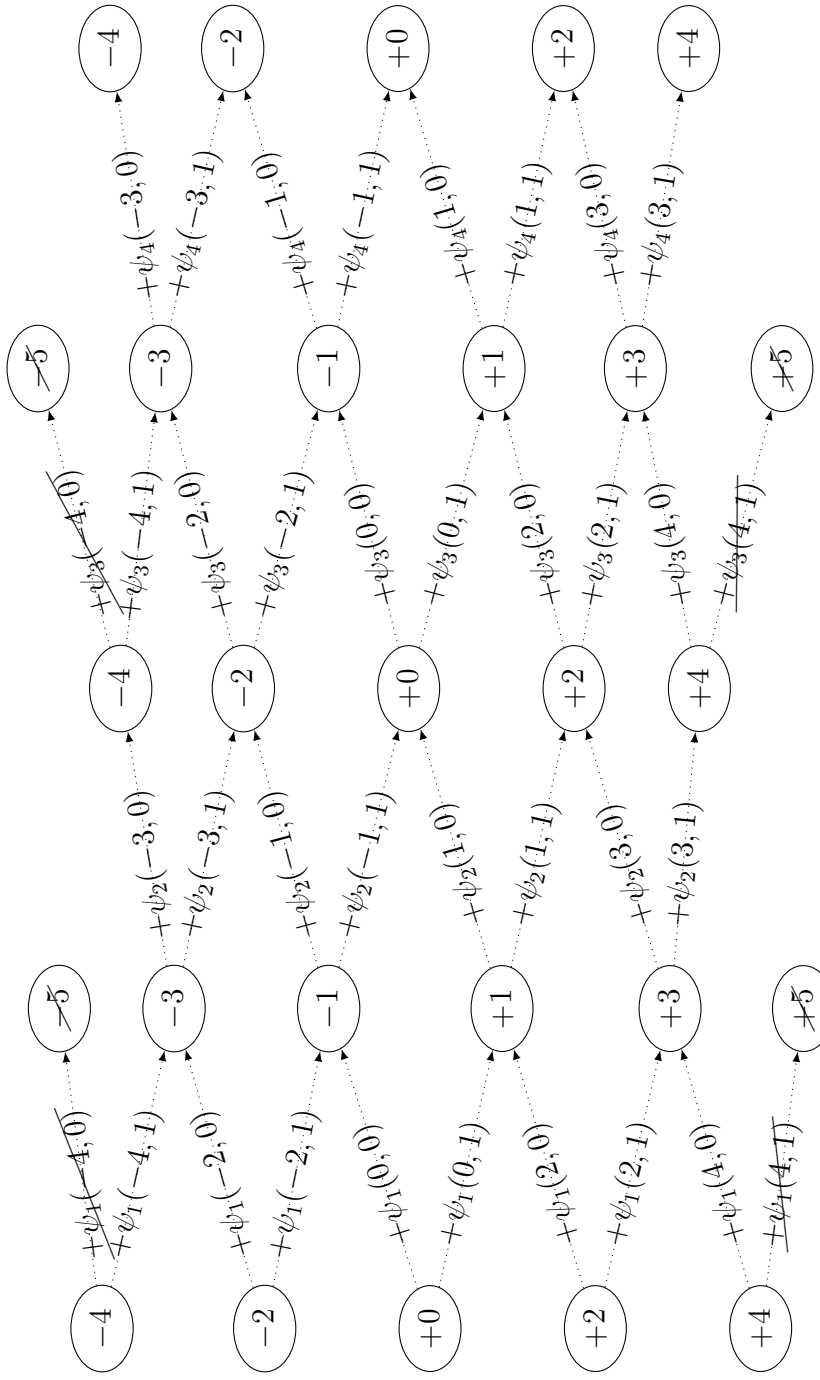

Figure 3.7: $x_k \in \mathrm{GF}(2^1), \tilde{x}_k \in \{-1, 1\}$ GS detection trellis with RDS states truncated beyond ± 4. The path metric is defined in equation (3.10) on the following page.

This expression demonstrates that the $\ln(\sqrt{2\pi n_0})$ term can be discarded as it is constant to all paths resulting in a path metric that is a function of the Euclidean distance, noise power and RDS probability.

$$\psi_k(\phi_{k-1}, x) = |y_k - \tilde{x}|^2 - 2n_0 \ln\left(\mathbb{P}\left[\Phi_k = \phi_{k-1} + \tilde{x} | \Phi_{k-1} = \phi_{k-1}\right]\right) \quad (3.10)$$

Since this intelligent detection algorithm is ultimately an application of the Viterbi algorithm, the computational complexity in its implementation is already established in the literature [7, p. 176]. Specifically complexity grows linearly with sequence length K and quadratically with the number of RDS states $2\overline{\Phi} + 1$.

3.3 Implementation

In order to test and simulate the detection mechanism described in section 3.2 on page 53 along with the coding technique outlined in section 2.3, use of software-defined radio (SDR) was selected. Section 3.3.1 provides an overview of the SDR suite that was selected as a platform to build the processing blocks discussed in section 3.3.2 on the following page.

3.3.1 GNU Radio

The GNU Radio[4] SDR suite was first released in 2001 as a free and open-source software (FOSS) high performance alternative to many of the sluggish proprietary options available at the time. The platform is divided into two parts: a scheduler and a collection of common signal processing blocks.

The scheduler component is the backend of the suite that interconnects the signal processing blocks. It flows signal streams between blocks and decides when specific blocks require compute time in order to maximize

[4] `https://www.gnuradio.org`

throughput. It was described using the C++98 [17] programming language for performance reasons while the interface is fully accessible via the Python scripting language. This allows the high level signal flowgraphs to be easily implemented using Python while maintaining the performance benefits of C++ for the lower level computationally intensive processing blocks.

The signal processing blocks in the GNU Radio SDR suite are typically implemented in C++ to maximize computational performance. Although the suite comes with many common signal processing blocks, the application programming interface (API) is well documented allowing users to build their own blocks as part of *out-of-tree modules*.

Web based resources for the GNU Radio project are plentiful. The source code is hosted on GitHub[5] while installation[6], GUI[7], Python API[8], and out-of-tree module[9] instructions are available on the project Wiki.

3.3.2 Blocks

All signal processing blocks developed for this research have been collected and made available as part of the `gr-gs`[10] Guided Scrambling GNU Radio Module. All blocks are described using the C++14 [18] programming language while simulation scripts are written in Python. Altogether twelve blocks were created to fully realize this work and they are all demonstrated in the `gs_demo.grc` graphical flow-graph contained in the module. For the sake of brevity this section will focus solely on the guided scrambling and detector blocks.

[5] `https://github.com/gnuradio/gnuradio`
[6] `https://wiki.gnuradio.org/index.php/InstallingGR`
[7] `https://wiki.gnuradio.org/index.php/TutorialsCoreConcepts`
[8] `https://wiki.gnuradio.org/index.php/TutorialsWritePythonApplications`
[9] `https://wiki.gnuradio.org/index.php/OutOfTreeModules`
[10] `https://github.com/eddic/gr-gs`

Guided Scrambler

The guided scrambler block implements multiple scramblers as shown in figure 2.10 on page 26 through heavily optimized C++ code while the selection method is kept independent through object oriented polymorphism. This allows trivial experimentation with different selection methods. Due to the parallel nature of the guided scrambling process described in section 2.3, the block was implemented using the multithreaded facilities available in C++14.

Detector

The detector is implemented as a trellis with both variable width and variable length defined using recursive object oriented principles. Due to the two-fold symmetry, real and imaginary components are detected entirely independently. The trellis dynamically deallocates nodes as it self-closes and dynamically allocates nodes as new signal points arrive.

Path metrics through the trellis are calculated using equation (3.10) on page 58. The conditioned source symbol probabilities are implemented as a lookup table of equation (3.3) on page 50. Since this transition matrix is typically very sparsely populated, it is compressed into a three dimensional matrix mapping the codeword position, previous RDS, and symbol value to probabilities:

```
sourceProbability[codewordPosition][rds][symbol]
```

Output sequences are extracted from the trellis once it self-closes. Self-closing, in this context, refers specifically to when a column in the trellis is collapsed into a single remaining node due to downstream path decisions cascading back upstream. Once the column collapses to a single node, it and all upstream nodes can be extracted as a section of the output sequence. The non-deterministic nature of the delay time before the trellis self-closes presents numerous engineering challenges, which ultimately motivated the

dynamic nature of the trellis implementation. Although less computationally efficient than a deterministic implementation, the trellis can grow and shrink as needed to accommodate the unpredictable nature of the closing time. In the event that this detection algorithm was to be implemented in a more computationally efficient manner, it would be beneficial to statistically characterize the closing times for all relevant code configurations and noise power levels.

In order to improve the trellis closing times in this implementation a non-optimal modification was made to the traditional Viterbi algorithm. When new columns are created as new signal points are received, nodes with path probabilities less than one tenth that of the best path probability in the column are instantly discarded. It was considered highly improbably that the statistics of paths with such low probability would change to the point that they would eventually be selected. This dramatically improved closing times while having no measurable effect on the error rates. This is especially helpful for low noise power scenarios due to the reduced weight of the MAP term in equation (3.10) on page 58.

Chapter 4

Results

Results, in this context, refer specifically to the error rates achieved using the intelligent detection strategies outlined in this document, as compared to the more traditional ML detection methods. Two specific cases are examined. Section 4.1 presents results for a rate $^9/_{12}$ GF(2^2) code while section 4.2 on page 65 considers the performance of a multilevel $^9/_{12}$ GF(2^4) code. The $^9/_{12}$ code rate was selected as a reasonable mid-point between no coding whatsoever and the $^1/_2$ rate Manchester code.

All error curves are generated through simulation at 0.5 dB noise power intervals. Data points are completed once 10,000 intelligently detected symbol errors have occurred, giving typical curve generation times in the 1–2 month range.

4.1 Case 1: GF(2^2) $^9/_{12}$

This case study involved equiprobable uncorrelated source symbol sequences with $X_k \in \mathrm{GF}(2^2), \widetilde{X}_k \in \mathbb{C}$. The constellation mapping is that in figure 2.14 on page 34. Source data was assembled into source words of length 9, to which 3 augmenting symbols were added, giving codewords of length 12.

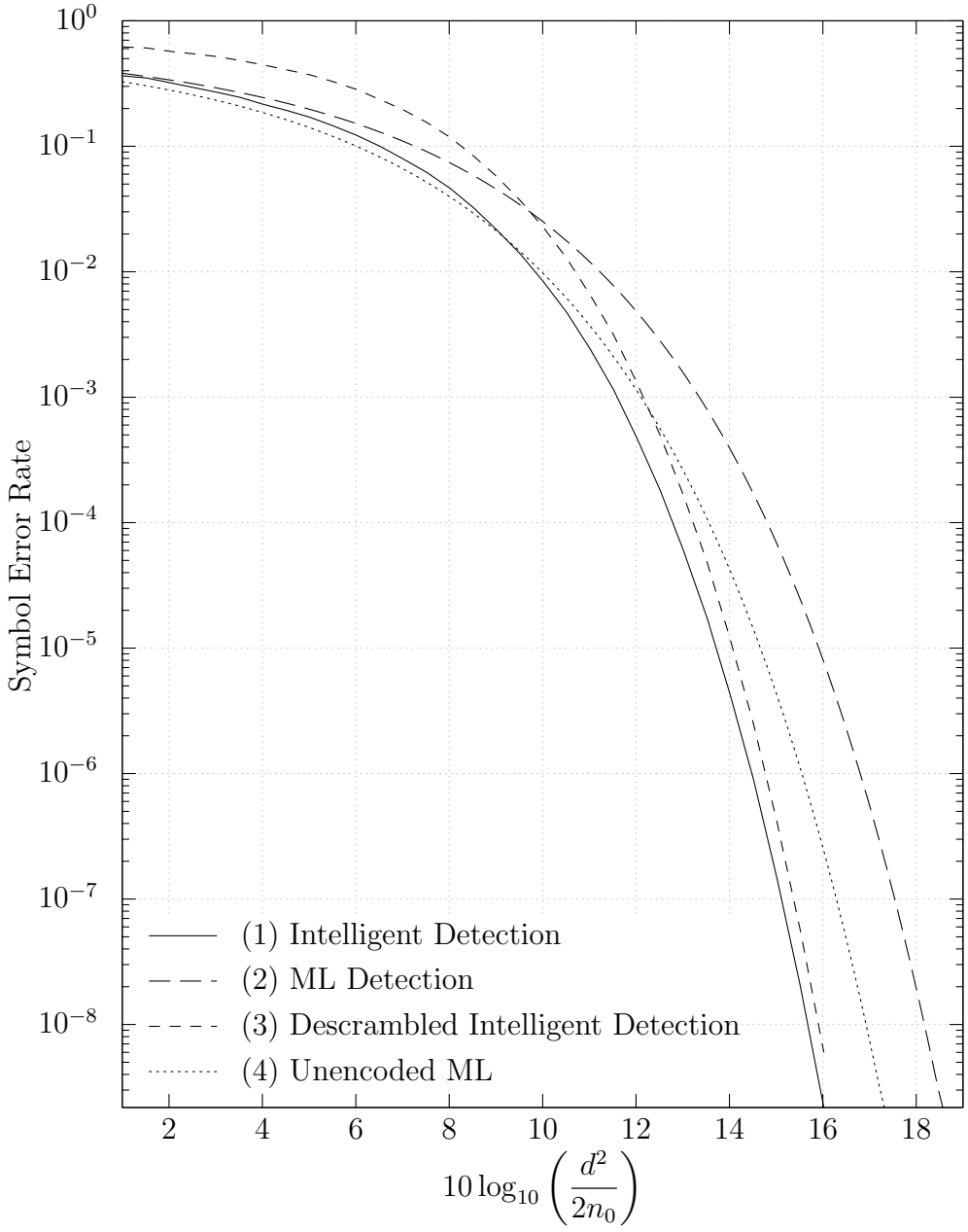

Figure 4.1: Error rates for GF(2^2) GS signals with a codeword length of 12 containing 3 augmenting symbols.

This configuration used $2^{2\times3} = 64$ scramblers. Source word candidates were scrambled using the GF(2^2) primitive polynomial $\alpha^5 + \alpha + 2$. Codewords were selected using the MSW selection method. The resulting error curves are shown in figure 4.1 on the preceding page.

Curve (1) was generated through simulation giving the symbol error rate after intelligent detection of the GS symbol sequence as described in the previous chapter, while curve (2) gives the symbol error rate with ML detection. This curve was also generated in the same simulation, and matches the expected result as described by equation (2.16) on page 35. Comparison of these curves shows the merit of this detection technique in cases where this GS code is already being used for spectral shaping purposes. The system gains the equivalent of an additional $> 2\,\mathrm{dB}$ of signal to noise ratio (SNR) at low error rates for "free" and continues to outperform traditional ML detection at higher error rates.

Curve (3) is generated through simulation and gives the symbol error rate of the intelligently detected GS symbol sequence *after* it has been descrambled. The increased error rate vs curve (1) is due to the error multiplication discussed in section 2.3. Given that the scrambler used has three non-zero terms, the error rate at low noise powers is slightly less than three times that of curve (1). The reason it isn't exactly three times that of curve (1) is due to erroneous augmenting symbols being discarded and not affecting the error rate. Curve (4) is the error rate of an ML detected equiprobable uncorrelated symbol sequence with reduced noise power. It matches equation (2.16) on page 35 but with n_0 scaled by 9/12 to account for the reduced baud rate due to the lack of redundancy. Comparing these curves shows merit in GS combined with this detection method as an error control code at high SNRs without any consideration for its spectral shaping properties. Below error rates of $\sim 10^{-3}$ the error control properties of this code outweigh its error multiplication during descrambling, giving the equivalent of an additional

$\sim 1\,\text{dB}$ of SNR.

4.2 Case 2: GF(2^4) 9/12

Although there is a lack of any previous research exploring the use of GS for generating balanced multilevel symbol sequences, there was never any reason to assume it would not apply. Since the implementation used in this research is not specific to any particular Galois field, it was deemed worthwhile to explore GS using the constellation pattern shown in figure 2.15 on page 36 along with the detection strategy that is the topic of this work. Like the case described in the previous section, source data was assembled into source words of length 9, to which 3 augmenting symbols were added giving codewords of length 12. This time, however, the configuration involved $2^{4\times 3} = 4096$ scramblers. The primitive scrambling polynomial used was $\alpha^4 + \alpha^2 + 2\alpha + 4$ and the selection method was MSW. See figure 4.2 on the following page for the error curves.

As in section 4.1 on page 62, curves (1) and (2) compare error rates for intelligent and ML detection, respectively. Similar to the GF(2^2) case explored in section 4.1, the rate 9/12 code in GF(2^4) shows the equivalent of a $\sim 2\,\text{dB}$ SNR gain at low error rates. Curves (3) and (4) also show similar gains post-descrambling as compared to the equiprobable ML detected sequence with scaled noise power.

All error rates discussed thus far have considered a ratio of noise power $2n_0$ to the minimum distance squared d^2 between constellation points. However, examining results from a ratio of average symbol energy E_s to noise power $2n_0$ gives interesting insights. Assuming equiprobable symbols, the average symbol energy E_s for the GF(2^4) constellation pattern in figure 2.15 is:

$$E_s = \mathbb{E}\left[\widetilde{X}^2\right] = \frac{4}{16}\left|\frac{3d}{2} + \frac{3d}{2}j\right|^2 + \frac{8}{16}\left|\frac{3d}{2} + \frac{d}{2}j\right|^2 + \frac{4}{16}\left|\frac{d}{2} + \frac{d}{2}j\right|^2$$

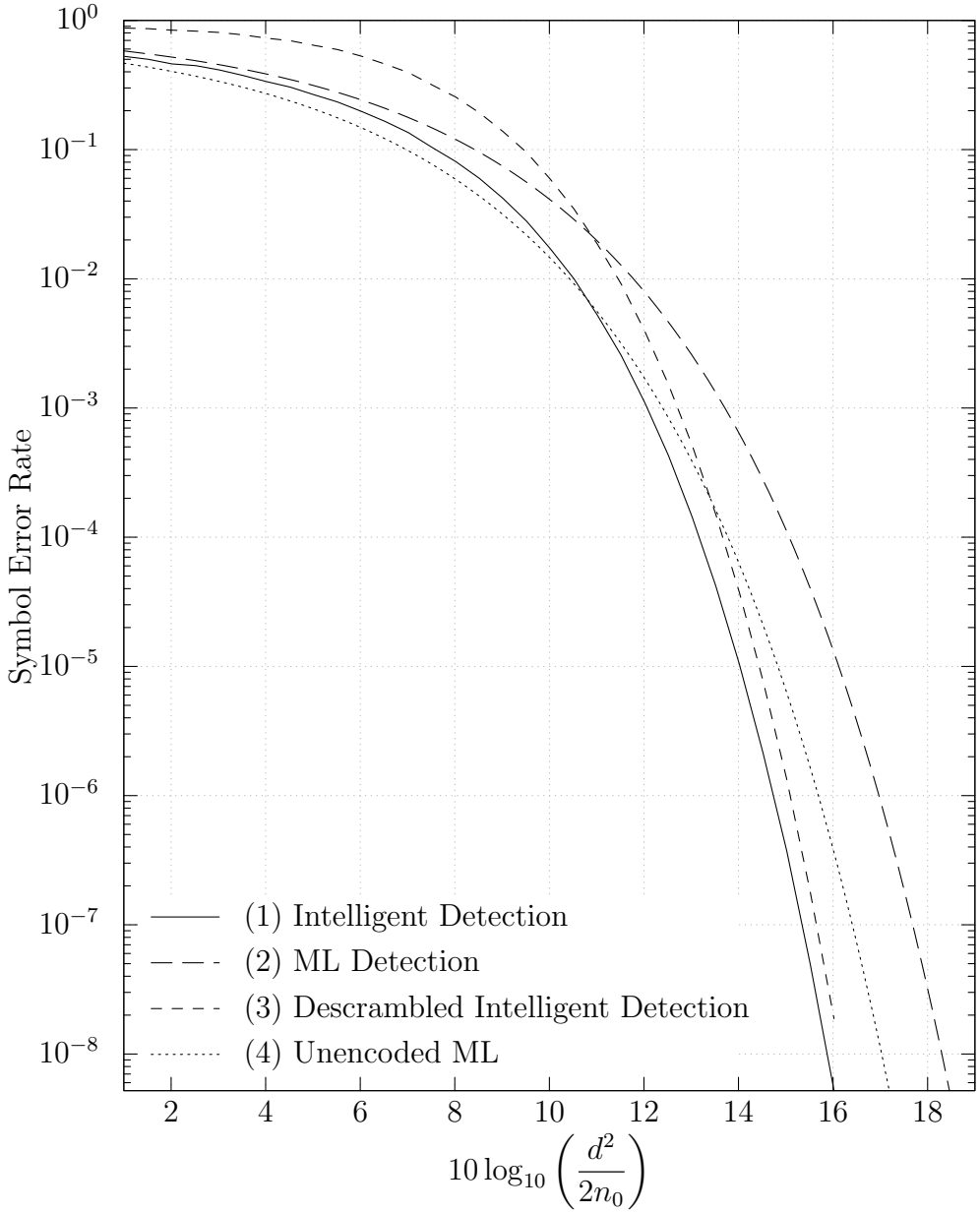

Figure 4.2: Error rates for GF(2^4) GS signals with a codeword length of 12 containing 3 augmenting symbols, where the horizontal axis is defined in terms of d^2/n_0.

$$= \frac{4}{16}\left(\frac{18d^2}{4}\right) + \frac{8}{16}\left(\frac{10d^2}{4}\right) + \frac{4}{16}\left(\frac{2d^2}{4}\right) = \frac{5d^2}{2} = 2.5d^2$$

Simulations show, however, that when GS balanced coding is used on multilevel symbol sequences, symbol probabilities are not equiprobable but follow a Gaussian profile. For the code configuration discussed in this case, the following first order symbol statistics are observed:

$$\begin{aligned} E_s = \mathbb{E}\left[\widetilde{X}^2\right] &\approx 1.90d^2 \\ \mathbb{P}[X \in \text{corner}] &\approx 0.12 \\ \mathbb{P}[X \in \text{centre}] &\approx 0.42 \\ \mathbb{P}[X \in \text{edge}] &\approx 0.12 \end{aligned}$$

Therefore applying a rate 9/12 balanced GS code to the multilevel GF(2^4) symbol sequence results in, approximately, a 25% reduction in signal power, which is practically equivalent to the amount of redundancy added to the sequence.

Since this complex valued constellation would typically be used with wireless channels, performance analysis in terms of signal power as opposed to minimum distance is warranted. Figure 4.3 on the following page reexamines these error rates in this context. The ML curves, although generated by simulation, are accurately described by equation (2.20) on page 37 with the above probabilities. Comparing curves (1) and (2) shows no improvement to that of figure 4.2 on the preceding page, however comparing curves (3) and (4) shows significant potential for this to be used as a low rate error control system for power limited multilevel signals at error rates less than $\sim 10^{-2}$, while simultaneously providing spectral shaping properties as an added bonus.

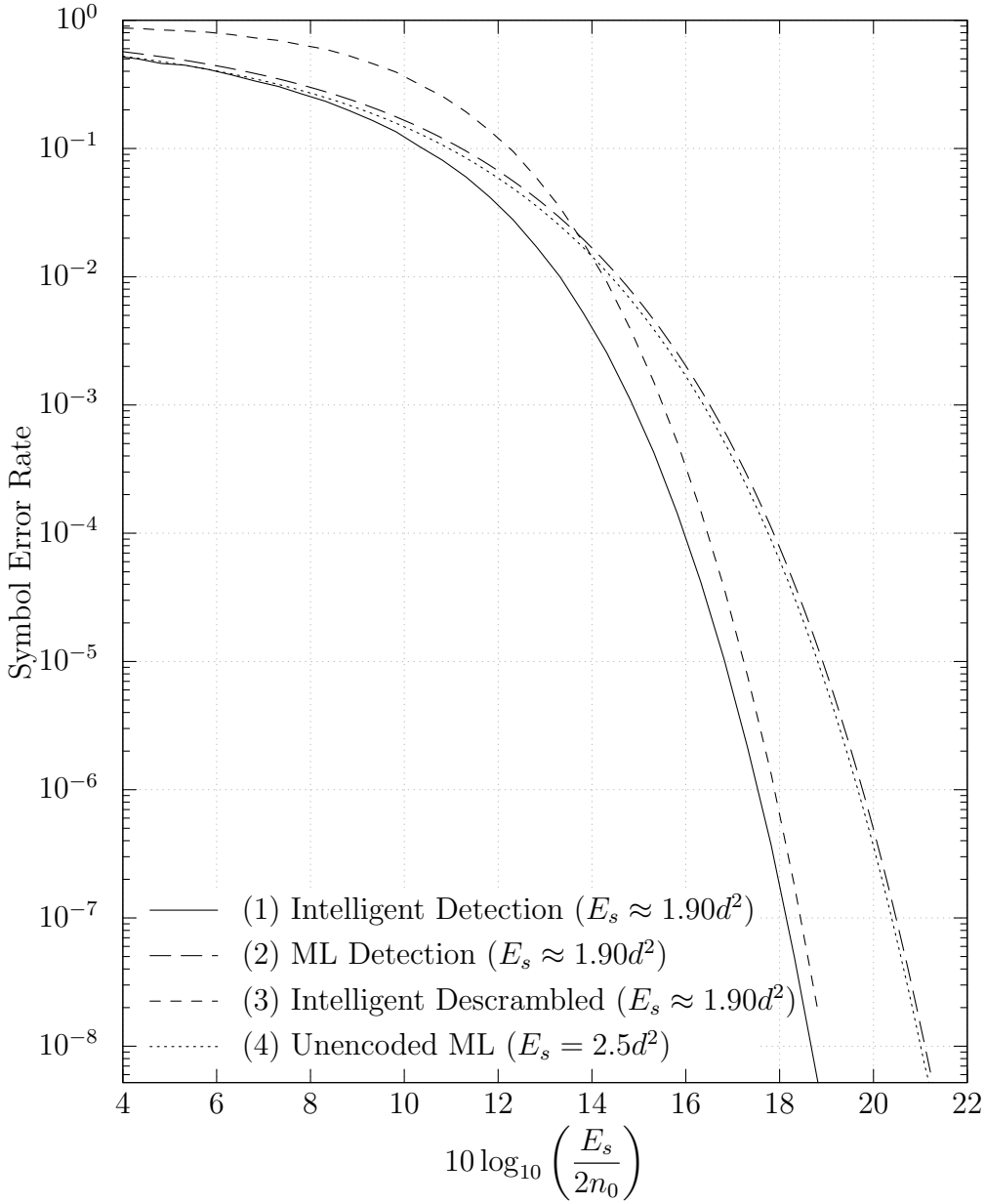

Figure 4.3: Error rates for GF(2^4) GS signals with a codeword length of 12 containing 3 augmenting symbols, where the horizontal axis is defined in terms of E_s/N_0.

4.3 Insights & Observations

Examining the internal behaviour of the intelligent detection algorithm while simulating the error curves has resulted in some observations of note.

The trellis path metric, as described by equation (3.10) on page 58, contains two terms. The first term is simply the ML or Euclidean distance term. It describes the distance between the received signal point and the constellation mapped symbol value. The second term is the RDS probability or MAP term. This second term is weighted by the noise power n_0. This weighting might lead one to assume that the effectiveness of this detection technique would decrease with noise power, converging toward ML detection as in figure 2.17 on page 40. As seen in the error curves shown in this chapter, however, this is not the case. The reason for this is that although the relevance of the second term is attenuated on a symbol-by-symbol basis at low noise powers, its total value will still accumulate over long sequences due to erroneous decisions affecting this same term in future iterations of the metric. It is for this reason that the intelligent detection method is still highly effective, albeit more computationally demanding, at lower noise powers.

The significance of the modified symbol probabilities shown in the previous section should not be understated. Although, on a theoretical level, the benefits of unequal symbol probabilities for the purpose of reducing signal power and improving error rates has been well established for quite some time, it has been considered impractical to implement. Specifically, it has been shown that symbol probabilities that are well approximated by a Gaussian distribution result in more efficient error rates when plotted against SNR. Experimenting with GS and the GF(2^4) signalling constellation from the previous section has shown that these symbol probabilities *do* follow a Gaussian distribution and it is expected that these gains would be realized to an even greater degree for larger signalling constellations.

Chapter 5

Conclusions

5.1 Summary

This research stands as a case study on the practical value in extracting what might be rich error control potential from constrained sequence codes. Chapter 1 provided an overview of the document along with relevant notation and conventions. Chapter 2 on page 6 detailed background information relevant to the research, specifically in the areas of channel coding, signal detection and guided scrambling. Chapter 3 on page 41 illustrated the construction of a statistical model for GS coded sequences and used this model to develop and implement intelligent detection. Results presented in chapter 4 on page 62 showed that this intelligent detection technique provides significant gains in error rates over more traditional detection techniques.

5.2 Future Research

The development of this detection technique and the associated `gr-gs` software suite opened the door for intriguing observations to be made. Specifically, the modular nature of the selection method implementation allowed

for easy experimentation with novel approaches. One promising approach is introduced in the following section.

5.2.1 MSW^2

When referring to balanced codes, this document has thus far focused exclusively on so called first-order balanced codes achieved by bounding the sequence RDS. However, the generation of balanced codes can also be accomplished by bounding other metrics, including the *running digital sum sum* (RDSS). Just as the RDS is the running sum of mapped digital symbol values over time (see equation (2.4) on page 10), the RDSS, θ_k is the running sum of RDS values over time:

$$\theta_k = \theta_{k-1} + \phi_k = \theta_0 + \sum_{n=1}^{k} \phi_n \tag{5.1}$$

It has been shown that if the RDSS is controlled in a manner similar to the RDS in a first-order balanced code, higher order spectral nulls can be realized that roll off in lower frequencies at $40\,{}^{\mathrm{dB}}/_{\mathrm{dec}}$ as opposed to just $20\,{}^{\mathrm{dB}}/_{\mathrm{dec}}$ [4, p. 243]. Just as RDS balanced codes reach maximum spectral shaping capabilities at rate $1/2$ with the Manchester code, RDSS balanced codes reach a similar peak at rate $1/4$ with the code described in table 5.1 on the next page resulting in a normalized sequence PSD of:

$$\left|\hat{X}\left(\hat{f}\right)\right|^2 = 4\sin^2\left(\pi\hat{f}\right)\sin^2\left(2\pi\hat{f}\right)$$

Although traditional MSW selection chooses the codeword candidate that best minimizes $\mathbb{E}[\Phi^2]$, it provides no mechanism for further gains in the face of candidate *ties*. In the event that two codeword candidates have the same MSW metric, randomly choosing between the two appears an appropriate way to proceed. The MSW^2 selection method proposes tracking the RDSS as well and using this information to assist in the selection process thereby

Source Word	Codeword
0	0110
1	1001

Table 5.1: RDSS equivalent of the Manchester code.

reducing or eliminating ties. Therefore as opposed to the metric being decided as per equation (2.14) on page 29 for MSW, the metric is defined as follows:

$$\mathrm{SW}_{i,j}^2 = \sum_{n=1}^{N+A} \left|\theta'_{i,j,n}\right|^2 + \Xi \sum_{n=1}^{N+A} \left|\phi'_{i,j,n}\right|^2 \tag{5.2}$$

The coefficient Ξ allows the selection process to establish priority weightings to either the RDS or RDSS. The limit as $\Xi \to \infty$ signifies only using the RDSS for tie breaking while the limit as $\Xi \to 0$ gives the reverse where the RDS is used for breaking ties while the primary selection criteria is the RDSS. Figure 5.1 on the next page compares the spectral performance of a rate $^{16}/_{24}$ GS coded binary sequence, generated when codewords are selected using equation (5.2), with different values of Ξ. Figure 5.2 on page 74 plots the change in RDS and RDSS statistics as Ξ is varied.

Since the resulting statistics of the RDSS could be modelled in much the same way as the RDS, they could also be used for intelligent detection opening the door for error control gains to be realized at rates below $^1/_2$.

5.3 Final Thoughts

Typically error control codes and constrained sequence codes have been isolated from each other and kept to their intended purposes. The assumption that constrained sequence codes have limited value for error control has left

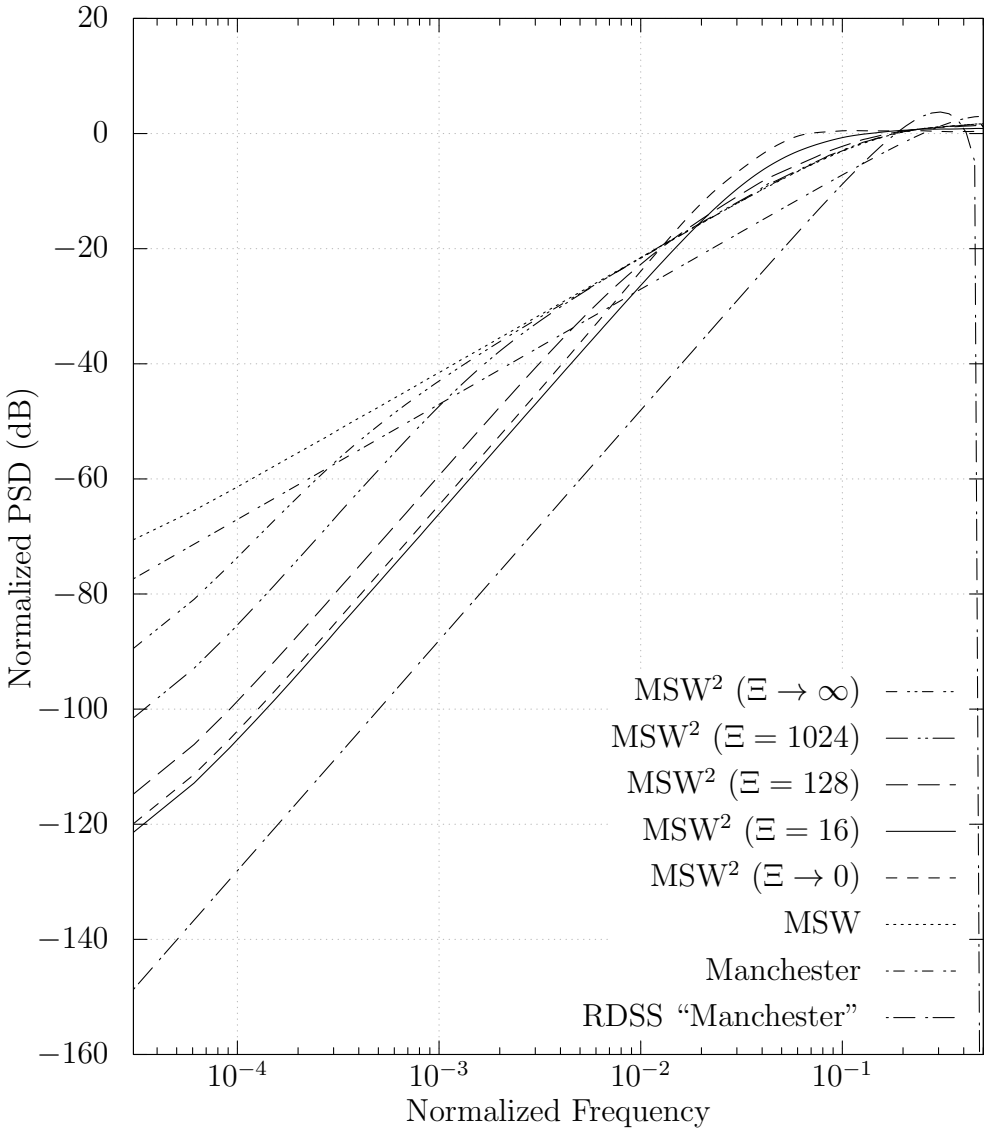

Figure 5.1: Normalized power spectral density for a GS coded sequence with $x_k \in \mathrm{GF}(2^1)$, $\tilde{x}_k \in \mathbb{R}$, MSW^2 selection, a codeword length of 24, and 8 augmenting symbols.

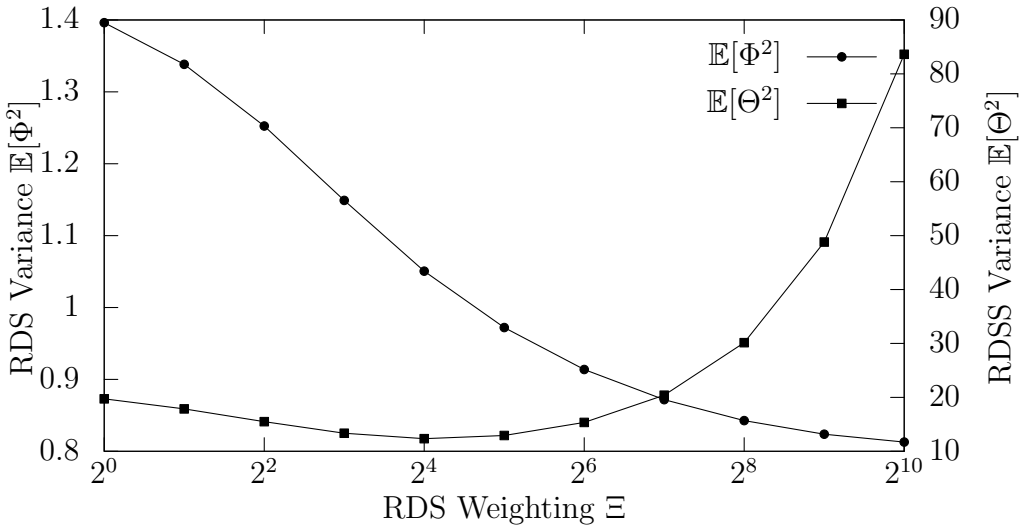

Figure 5.2: RDS and RDSS statistics for a GS coded sequence with $x_k \in \mathrm{GF}(2^1)$, $\tilde{x}_k \in \mathbb{R}$, MSW^2 selection, a codeword length of 24, and 8 augmenting symbols as the RDS weighting, Ξ, is varied.

an avenue largely unexplored that might lead to significant gains in communication channel use efficiency.

Guided scrambling is an excellent candidate for the aforementioned possibilities in that the largely non-deterministic[1] nature of the sequences that can be generated through different selection methods opens many doors to a wide variety of constraints with associated probabilistic detection techniques. The author firmly believes that the exploration of selection methods providing tighter constraints might lead to lower rate codes that could rival modern error control codes while simultaneously generating sequences with highly desirable physical properties.

Finally it is hoped that the reader is convinced of the value in using

[1] As opposed to traditional block codes.

free and open-source software-defined radio systems, particularly GNU Radio, for high-performance simulations in communications research. Through GNU Radio, the `gr-gs`[2] guided scrambling module should prove itself a valuable platform for those pursuing further research into guided scrambling and associated intelligent detection techniques.

[2] `https://github.com/eddic/gr-gs`

Bibliography

[1] H. Kobayashi, B. L. Mark, and W. Turin, *Probability, Random Processes, and Statistical Analysis: Applications to Communications, Signal Processing, Queueing Theory and Mathematical Finance*. Cambridge; New York: Cambridge University Press, Dec. 2011.

[2] R. Togneri and C. J. S. deSilva, *Fundamentals of Information Theory and Coding Design*. Chapman and Hall/CRC, 1 edition ed., Jan. 2003.

[3] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.

[4] K. A. Schouhamer Immink, *Codes for mass data storage systems*. Eindhoven (The Netherlands): Shannon Foundation Publ., 2004. OCLC: 492318105.

[5] C. G. Bell, J. C. Mudge, and J. E. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*. Digital Press, May 2014.

[6] P. Horowitz and W. Hill, *The Art of Electronics*. New York, NY: Cambridge University Press, 3 edition ed., Mar. 2015.

[7] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication*. Boston: Springer, 3rd ed. 2004 edition ed., Sept. 2003.

[8] B. P. Lathi, *Signal Processing and Linear Systems*. New York: Oxford University Press, Apr. 2001.

[9] I. J. Fair, W. D. Grover, W. A. Krzymien, and R. I. MacDonald, "Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems," *IEEE Transactions on Communications*, vol. 39, pp. 289–297, Feb. 1991.

[10] K. A. Schouhamer and L. Patrovics, "Performance assessment of dc-free multimode codes," *IEEE Transactions on Communications*, vol. 45, pp. 293–299, Mar. 1997.

[11] I. J. Fair, Q. Wang, and V. K. Bhargava, "Polynomials for guided scrambling line codes," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 499–509, Apr. 1995.

[12] D. R. Smith, *Digital transmission systems*. Boston: Kluwer Academic Publishers, 3rd ed ed., 2004.

[13] I. J. Fair, V. K. Bhargava, and Q. Wang, "On the spectral characteristics of continuous guided scrambling line codes," in *Proceedings of Canadian Conference on Electrical and Computer Engineering*, pp. 784–789 vol.2, Sept. 1993.

[14] H. Lampow-Maundy and I. J. Fair, "Frame synchronization in guided scrambling line codes," *IEEE Transactions on Communications*, vol. 48, pp. 1992–1996, Dec. 2000.

[15] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, Apr. 1967.

[16] Y. Zhu and I. J. Fair, "Modified minimum squared weight selection criterion for DC-free multimode codes," *Electronics Letters*, vol. 41, pp. 973–975, Aug. 2005.

[17] "Programming languages – C++," Tech. Rep. ISO/IEC 14882:1998, Sept. 1998.

[18] "Programming languages – C++," Tech. Rep. ISO/IEC 14882:2014, Dec. 2014.