National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**DETERMINATION OF MAXIMAL-LENGTH SEQUENCES BY
WEIGHT DISTRIBUTION ANALYSIS**

by

Sean Faulkner, B.Sc., B.Eng.

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfilment of the requirements
for the degree of

Master of Engineering

**Ottawa—Carleton Institute of Electrical Engineering
Faculty of Engineering
Department of Electronics
Carleton University
August 1989**

© Sean A. Faulkner, 1989

**National Library
of Canada**

**Bibliothèque nationale
du Canada**

Canadian Theses Service Service des thèses canadiennes

**Ottawa, Canada
K1A 0N4**

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-54371-X

Canada

**The undersigned recommend to the Faculty of Graduate Studies
and Research the acceptance of the thesis:**

**"Determination of Maximal-Length Sequences by
Weight Distribution Analysis"**

**submitted by Sean Faulkner in partial fulfillment of the
requirements for the degree of Master of Engineering.**

Prof. J.S. Wight
Thesis Supervisor

Prof. A.R. Boothroyd
Acting Chairman,
Department of Electronics

September, 1989

ABSTRACT

A new technique has been developed and analyzed which allows the determination of the spreading code used by an intercepted DS-SS signal. This technique is based on an analysis of the weight distribution of a cross-correlation function between two maximal-length sequences. The technique is combined with knowledge of the correlation and decimation properties of m-sequences to form the basis of a code search strategy. This strategy gives improved results over any known existing method. The technique has been shown to give correct results for input SNR's down to about -12 dB. The time required to perform the code search has been reduced by orders of magnitude compared to existing methods. As well, the properties of m-sequences have been compiled and combined resulting in some interesting discoveries concerning decimations, correlations and weight distributions.

ACKNOWLEDGEMENTS

The author extends his most sincere appreciation for the friendship and guidance of his thesis supervisor, Professor James S. Wight. His encouragement and support were most welcome and his "good brain to bounce ideas off of" was full of good ideas.

To the Department of National Defence, I extend my thanks for allowing me to proceed into the MEng program under the Post-Graduate Training on-Scholarship program.

I would like to thank Dr. Frank H. Palmer and the staff of the Communications Electronic Warfare Section at the Defence Research Establishment Ottawa for use of their resources, both materiel and intellectual. Details of their help and support would fill volumes.

I am very grateful to my parents who have supported and continue to support my academic endeavours. The encouragement which has been provided through many years of schooling has been most appreciated.

I would like to thank the Leonowicz's for their support, including their provision of babysitting when the crunch was on.

Finally, I wish to thank my wife, Anne-Marie for being so patient and understanding and who spent so many lonely nights taking care of Andrew.

I would also like to thank the many people who I have forgotten to thank here. It takes a lot of support to finish a thesis and every bit of it is very much appreciated.

TABLE OF CONTENTS

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Tables	vii
List of Figures	viii
List of Symbols and Abbreviations	x
Chapter 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Background and Objectives	2
1.3 Thesis Organization	4
Chapter 2 THEORY OF SPREAD SPECTRUM MODULATION	6
2.1 Introduction	6
2.2 Types of Spread Spectrum Signals	8
2.2.1 Direct—Sequence Spread Spectrum	12
2.3 Correlation	19
2.3.1 Digital Correlation	19
2.3.2 Acousto—Optic Correlation	22
Chapter 3 SPREAD SPECTRUM CODING	30
3.1 Introduction	30
3.2 Classification of Spread Spectrum Codes	31
3.3 Galois Field Theory	34
3.4 Linear Recursions and Code Generators	39
3.5 Properties of M—Sequences	43
3.5.1 Decimation	46
3.5.2 Correlation	52
3.5.3 Weight Distribution	58
3.6 Spread Spectrum Code Interception	61
3.6.1 Introduction	61

3.6.2	Decoding of Intercepted M-Sequences	63
3.6.2.1	Berlekamp's Algorithm	66
3.6.3	Null Filtering	70
3.6.4	Autocorrelation Techniques	72
Chapter 4	A WEIGHT DISTRIBUTION ANALYSIS TECHNIQUE	77
4.1	Introduction	77
4.2	Code Families	78
4.3	Cross-Correlation Analysis	82
4.4	Weight Distribution Analysis	96
4.4.1	Calculation of Weight Distribution in Additive White Gaussian Noise	98
4.4.2	Derivation of the MSD Function Between a Weight Distribution and the Adaptive Gaussian	101
4.4.3	Derivation of the SKEW Function Between a Weight Distribution and the Adaptive Gaussian	104
Chapter 5	RESULTS	108
5.1	Introduction	108
5.2	Code Analysis Software	108
5.2.1	Program XCORR	109
5.2.2	Program FAMILIES	114
5.2.3	Program NOISEANA	117
5.2.4	Miscellaneous Programs	120
5.3	Weight Distribution Analysis Results	121
5.4	Code Search Strategy Based on Weight Distribution Analysis	136
5.5	A Comparison Between The Code Search Strategy and Brute Force Autocorrelation Techniques	143
Chapter 6	CONCLUSIONS	146
6.1	Summary of Results	146
6.2	Future Work	149
References		150

LIST OF TABLES

3.1	The elements of $GF(2^4)$ generated by $p(x) = 1 + x + x^4$ [16]	36
3.2	Shift register states and output for the sequence generated using $g(x) = 1 + x + x^4$	42
3.3	The number and length of m-sequences of linear span L and the maximum cross-correlation magnitude	45
3.4	Characteristic polynomials and decimations for the R8's	48
3.5	Cross-correlation values of reciprocal sequences [18]	57
3.6	Example of the application of the Berlekamp algorithm	69
4.1	The number of code families for the R3's to R15's	81

LIST OF FIGURES

2.1	a) Hybrid FH/DS–SS Modulator [2] b) Frequency Spectrum of Hybrid FH/DS–SS System	11
2.2	BPSK Direct–Sequence Spread Spectrum Modulation [3]	13
2.3	BPSK DS Spreading and Despreading [3]	14
2.4	Ideal Power Spectra of DS–BPSK Signals	16
2.5	Real DS–BPSK Power Spectra [26]	17
2.6	Block Diagram of a Single Dwell Time PN Acquisition System with Non–Coherent Detection [1]	20
2.7	Time–Integrating Correlator (TIC) [26]	24
2.8	Acousto–Optical Diffraction Geometry for Input at Single Frequency [35]	24
2.9	Correlation Process for AO Photodetectors [26]	26
2.10	Pedestal Removal Process for TIC [26]	27
3.1	M–Sequence Generator Architectures	41
3.2	Sequence Generator for $g(x) = 1 + x + x^4$	42
3.3	Chart of Imbeddedness up to R31	50
3.4	Decimation Relations for the R5's [22]	51
3.5	The Autocorrelation Function for M–sequences	53
3.6	Plot of $(\text{SNR})_{\text{in}}$ vs $(\text{SNR})_{\text{out}}$ for Radiometer and Doubler [4]	62
3.7	Probability of Error for Binary Signals [10]	64
3.8	Null Filter Sequence Generator Architecture	71
3.9	Code Search Time for the Correlation Techniques	75
4.1	Code Families for the R7's and R8's	80

4.2	Cross—Correlation Functions for the R8's (1)	84
4.3	Cross—Correlation Functions for the R8's (2)	85
4.4	Cross—Correlation Functions for the R8's (3)	86
4.5	Decimation Relations for the R8's	87
4.6	Two Cross—Correlation Functions in the Same R8 Code Family	88
4.7	Noiseless Weight Distribution Function for Code Family #2 of R8's	97
5.1	Functional Hierarchy of the XCORR program	111
5.2	Functional Hierarchy of the FAMILIES program	115
5.3	Functional Hierarchy of the NOISEANA program	118
5.4	Code Family #1 Weight Distribution in Noise	123
5.5	Code Family #2 Weight Distributions in Noise	124
5.6	Code Family #3 Weight Distributions in Noise	125
5.7	Code Family #4 Weight Distributions in Noise	126
5.8	Code Family #5 Weight Distributions in Noise	127
5.9	Code Family #6 Weight Distributions in Noise	128
5.10	Code Family #7 Weight Distributions in Noise	129
5.11	Code Family #8 Weight Distributions in Noise	130
5.12	Code Family #9 Weight Distributions in Noise	131
5.13	Code Family #10 Weight Distributions in Noise	132
5.14	MSD and SKEW Functions for the R7's	133
5.15	MSD and SKEW Functions for the R8's	134
5.16	MSD and SKEW Functions for the R9's	135
5.17	Weight Distribution Merge Graph for the R8's	138
5.18	Comparison of Code Search Times for Various Techniques	144

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOLS

$a_i[b_i]$	inverse decimation pair
a_n	individual elements of the binary sequence $\{a_n\}$
α	root of primitive polynomial
$\chi(\cdot)$	mapping operator from $\{0,1\}$ -valued sequences to $\{+1,-1\}$ -valued sequences
C_i	i^{th} cyclotomic coset
δ	relative phase shift operator (for shift-and-add)
$\delta(x)$	Kronecker delta function
ϵ	relative phase shift between sequences in time ($\epsilon = \tau T_c$)
$\eta[R_q(\tau)]$	weight distribution of the cross-correlation function
$\eta[R_q(\tau)]_i$	i^{th} element of the weight distribution corresponding to the number of occurrences of the i^{th} cross-correlation magnitude
$E[\cdot]$	expected value operator
f_c	code generator frequency, or chip rate
f_o	carrier frequency
f_d	data rate
$f(x)$	general polynomial
$f^*(x)$	reciprocal polynomial of $f(x)$
$G(\alpha)$	Gaussian probability density function
$GF(q)$	Galois field, or finite field with q elements
$g(x)$	generating polynomial, or characteristic polynomial, of an m -sequence
L	linear span, or number of stages in a shift register code generator
L_L	number of distinct values that $R_{-1}(\tau)$ takes on
$m_\alpha(x)$	minimal polynomial of α
$\mu(\cdot)$	Möbius function
N	period of an m -sequence
$N_1(d)$	number of irreducible polynomials of degree d over $GF(2)$

n_i	short form for $\eta[R_q(\tau)]_i$
N_o	noise power spectral density
$N_p(d)$	number of primitive polynomials of degree d over $GF(2)$
$n(t)$	noise function
$\phi(k)$	Euler ϕ -function
q, r	arbitrary decimations
r_i	short form for $[R_q(\tau)]_i$
$r_{xy}(\epsilon)$	cross-correlation function (temporal)
$R_a(\tau)$	full-period autocorrelation function (sampled)
$R_a(K, n, \tau)$	partial-period autocorrelation function (sampled)
$\langle R_a(K, n, \tau) \rangle$	time-averaged mean of the partial-period autocorrelation function
$\langle R_a(K, n, \tau) ^2 \rangle$	time-averaged variance of the partial-period autocorrelation function
$R_{ab}(\tau)$	full-period cross-correlation function (sampled)
$R_q(\tau)$	full-period cross-correlation function between two m -sequences related by the decimation q
$[R_q(\tau)]_i$	i^{th} cross-correlation magnitude
$R_{-1}(\tau)$	full-period cross-correlation function between an m -sequence and its reciprocal
R_n 's	the m -sequences with linear span n
σ	standard deviation
σ_r	short form for $\sigma_{r_{xy}(\epsilon)}$
$\sigma_{r_{xy}(\epsilon)}$	standard deviation of the cross-correlation function for a specific phase shift
σ^2	variance
$\sigma_{r_{xy}(\epsilon)}^2$	variance of the cross-correlation function for a specific phase shift
$S[q]$	decimation by q of sequence S
$S^*[r]$	reciprocal decimation
s_n	individual elements of a binary sequence $\{s_n\}$
T	period of m -sequence in time ($T = NT_c$)
$\text{Tr}_2^{2^n}(z)$	trace polynomial from $GF(2^n)$ to $GF(2)$
$T_p(n)$	time to find the families for the R_n 's

T_s	time to switch phase for digital correlator
$\bar{T}_s$	time to switch phase for acousto—optic correlator
τ	relative phase shift between two sequences
$W_{xy}(\alpha)$	weight distribution function
$Y(N)$	number of cyclotomic cosets modulo N
$\mathbb{I}'$	the set of positive integers

ABBREVIATIONS

AJ	Anti—Jamming
AM	Amplitude Modulation
AO	Acousto—Optic
AWGN	Additive White Gaussian Noise
BCH	Bose, Chaudhuri, and Hocquenghem
BPSK	Binary Phase—Shift—Keying
CEW	Communications Electronic Warfare
dB	decibels
DREO	Defence Research Establishment Ottawa
DS	Direct—Sequence Spread Spectrum
DSCS	Defence Satellite Communications System
EHF	Extremely High Frequencies (30—300 GHz)
EWD	Electronic Warfare Division
FH	Frequency—Hopped Spread Spectrum
FM	Frequency Modulation
gcd	Greatest Common Divisor
GPS	Global Positioning System
JTIDS	Joint Tactical Information Distribution System
LPD	Low Probability of Detection
LPE	Low Probability of Exploitation
LPI	Low Probability of Intercept
M—sequence	Maximal—Length Sequence
mod	modulus
MSD	Mean—Squared Difference

MSK	Minimum Shift–Keying
PG	Processing Gain
PLRS	Position Locating Reporting System
PN	Pseudo–Noise
PSK	Phase–Shift–Keying
QPSK	Quadrature PSK
SINCGARS	Single Channel Ground–Airborne Radio System
SKEW	Skewness Function
SNR	Signal–to–Noise Ratio
SS	Spread Spectrum
TDMA	Time Division Multiple Accessing
TH	Time–Hopped Spread Spectrum
TIC	Time–Integrating Correlator
UHF	Ultra High Frequencies (300 MHz – 3 GHz)

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

One of the characteristics of direct-sequence spread spectrum signals is their low probability of interception. The signal interception problem is complicated in two ways: (1) a large frequency band must be monitored and (2) the power spectral density of the signal to be intercepted is lowered in the spreading process as compared to an unspread signal. The signal spreading is achieved through the use of spreading codes of which only the transmitter-receiver pair have knowledge. To despread the signal and thus recover the data, the transmitter and receiver must maintain synchronization between their codes.

To further complicate the spread spectrum signal interception problem, this spreading code can be generated using codes which are difficult for an interceptor to determine. Techniques such as signal encryption, modulation hybridization and interleaving also complicate the task of the interceptor. Other factors to be considered include a lack of knowledge of the data modulation, chip rates, carrier frequencies and code phase as well as the existence of multiple-access interference, jamming, and in-band noise.

In this thesis, many of the secondary difficulties will be ignored in order to concentrate on the problem of determining the spreading code which is used in an intercepted spread spectrum transmission. In particular, emphasis will be placed on

determining the code assuming that the spreading sequence employed is a binary linear-feedback shift register sequence of maximal period or binary m-sequence.

It is assumed that the reader has some knowledge of spread spectrum systems, communications theory and modern algebra.

1.2 BACKGROUND AND OBJECTIVES

In the past, techniques have been developed which could be utilized to determine which m-sequence was being employed as the spreading code. There are two groupings of techniques which could be used for this determination. The first set of techniques involved decoding the individual bits, or chips of the code. The methods which will be detailed from this set include Berlekamp's algorithm (one of many techniques to solve simultaneous linear equations) and the null filtering technique.

Berlekamp's algorithm [17] is an iterative algorithm which can be applied to solve for the generating polynomial of the spread spectrum code generator. By receiving $2L$ sequential error-free chips of a sequence, one can determine the code generator for an L -stage m-sequence of period $N = 2^L - 1$. Error-free implies that there are no errors due to noise and no data transitions occurring within the set of $2L$ chips. Null filtering involves setting the feedback taps of a code generator to the settings which are suspected to be those of the code to be intercepted. By feeding the intercepted sequence into this code generator, the sequence can be found by trying all possible settings until the all-zero sequence is output (hence, the name null filter). This all-zero sequence is the indication that the feedback taps correspond to those of the intercepted sequences code generator.

Unfortunately, both of these methods begin to fall apart in the presence of code chip errors and it is accepted that these techniques generally fail if the input Signal-to-Noise Ratio (SNR) is less than about 5 dB. Since the purpose of Direct-Sequence Spread Spectrum (DS-SS) is to present a low probability of interception (LPI) to an unknown interceptor, then typically a DS-SS signal would provide an input SNR to an interceptor which is much less than 5 dB. Thus, these techniques will have a very high probability of failure and the spreading code will not be determined.

The second set of techniques involves analysis of the correlation functions. The technique which will be discussed from this set is the brute force autocorrelation method. With this method, the intercepted code is correlated against all possible reference codes until an autocorrelation peak is detected. The advent of acousto-optic correlators enhances this method because many phases of a correlation function can be determined in parallel as will be discussed later. While the autocorrelation technique has the advantage that it can perform well in low SNR environments, it has the disadvantage that for X possible reference codes, an average of $X/2$ full-period correlations will have to be performed, assuming that the autocorrelation peak is detected the first time around.

In this thesis, a method will be proposed which is based on analysis of the weight distribution of a cross-correlation function between two m -sequence codes. Decimation properties of m -sequences will be exploited to improve the technique substantially. These properties have not previously been exploited, especially not in combination with weight distribution and correlation properties. While requiring significant but feasible processing, this technique is an improvement over all the aforementioned techniques. The method has been demonstrated theoretically to

work down to input SNR's of about -12 dB which is a significant improvement over the decoding techniques. The number of correlation functions which must be calculated is orders of magnitude less than the correlation methods in large part due to the decimation properties which will be exploited. Unfortunately, this number increases as the SNR gets lower, until it has degenerated effectively to the number of calculations required by the autocorrelation method.

1.3 THESIS ORGANIZATION

Chapters 2 and 3 discuss the theory required for an understanding of the work performed in this thesis. In Chapter 2, a basic discussion of spread spectrum modulation theory will be undertaken with an emphasis on Direct-Sequence Binary Phase-Shift-Keying (DS-BPSK). Correlators will then be described as they apply to SS systems. A basic description will be given of a Time-Integrating Correlator (TIC) developed at the Defence Research Establishment Ottawa (DREO) using acousto-optic (AO) technology as well as some background information on acousto-optics as it applies to this work.

In Chapter 3, the theory of SS codes will be developed with particular attention paid to m-sequences. The special properties of these sequences will be discussed, including decimation, correlation and weight distribution properties. Previous work which has been performed on the interception of this class of codes will also be presented.

Chapter 4 details all of the original work which has been performed in the course of this thesis. The observation that m-sequence codes can be grouped into decimation families will be discussed. The technique of code interception based on

cross-correlation analysis will be presented and demonstrated to be inadequate. Also, the new technique of code interception based on the analysis of the weight distribution of the cross-correlation function will be derived.

Chapter 5 gives results based on the new interception technique and shows the improvement that this technique can give compared to using the brute force autocorrelation peak technique. As well, the software which was developed for this thesis work will be discussed. A code search strategy based on weight distribution analysis is presented and examples given.

In Chapter 6, a summary of results and recommendations for future work are presented.

CHAPTER 2

THEORY OF SPREAD SPECTRUM MODULATION

2.1 INTRODUCTION

Although they have been in use for at least four decades, spread spectrum (SS) communications have only come under intense scrutiny over the last ten years. While SS modulation was predominantly used in military communications, it has now spread into the civilian realm. Such projects as the Space Shuttle, the NAVSTAR Global Positioning System (GPS) [3], Glonass (Soviet GPS) and Starfix [24] have prompted academic and industrial groups to devote time and effort to the analysis and development of spread spectrum systems. However, the military will continue to be the primary user of SS systems with such systems as the Joint Tactical Information Distribution System (JTIDS), DSCS and EHF SATCOM to name a few. Prior to discussing the various types of spread spectrum modulation, it is necessary to understand what SS communications are.

Although the definition of spread spectrum varies somewhat from source to source, an acceptable definition can be derived from [Dixon, 2] where *spread spectrum* is defined as:

Modulation techniques used to widen the RF bandwidth of a signal such that the transmitted bandwidth is much wider than would be necessary for information transmission alone. Spread spectrum has two distinguishing characteristics:

1. The transmitted bandwidth is much larger than the information bandwidth, and
2. The spectral characteristics of the transmitted signal are determined by the spreading waveform rather than the information waveform.

Thus it is apparent from this definition that such techniques as wideband FM and the use of error-control coding, both of which result in a wider RF bandwidth than that resulting from the information alone, are not considered spread spectrum.

There are numerous advantages to using spread spectrum signals. Some of these advantages are as follows:

1. Low Probability of Detection (LPD), Low Probability of Intercept (LPI) or Low Probability of Exploitation (LPE);
2. Anti-Jamming (AJ);
3. Multiple-Accessing;
4. High-resolution ranging;
5. Interference Rejection including multipath rejection;
6. Accurate universal timing; and
7. Low-density power spectra for signal hiding and to meet satellite downlink power spectral density guidelines.

It should be noted that there are also disadvantages inherent in the use of SS techniques. Some of the disadvantages include the following:

1. Increased transmitter and receiver complexity and cost;
2. Synchronization difficulties; and

3. Increased bandwidth beyond that required for the transmission of the information.

Full descriptions of these and other properties can be found in any one of several texts on spread spectrum such as [1],[2],[6].

2.2 TYPES OF SPREAD SPECTRUM SIGNALS

Since the advantages of SS communications depend on the specific form of SS modulation utilized, a short discussion of each of the more common types of SS modulation will now be undertaken. The main forms of SS modulation are as follows:

- 1) Direct—sequence (DS);
- 2) Frequency—hopped (FH);
- 3) Time—hopped (TH);
- 4) Chirp; and
- 4) Hybrid (e.g., DS/FH, TH/FH, TH/DS, DS/FH/TH, ...).

Direct—sequence SS signals result from phase—shift—keying a carrier at a high rate using a spreading code. If a high enough code rate is used, the resulting modulated signal can disappear into the background noise of a receiver at a sufficient distance from the transmitter. This produces a signal with a low probability of intercept (LPI). DS—SS signals with their inherent LPI characteristic are the main topic of this thesis, and will be discussed in detail in Section 2.2.1

Frequency—hopped SS signals result from rapidly shifting or "hopping" the carrier between different frequencies. Frequency—hopping systems are generally classified as either slow or fast hopping. A slow hopper transmits more than one

data bit per hop frequency. This can provide a signal with some AJ capability and resistance to multipath and selective fading effects, but slow hoppers are not usually considered to have LPI. A fast hopping system hops to more than one frequency during each data bit. This provides more resistance to jamming, multipath and selective fading as well as providing some measure of LPI if the hop rate is high enough and if enough hop frequencies are used. Frequency-hopped signals have obvious benefits to the military, and frequency-hopping techniques are being used in a number of systems including the Have Quick UHF AM radio and the Single Channel Ground-Airborne Radio System (SINCGARS). Refer to references [1], [2], [3], and [6] for more information.

In time-hopped SS systems, the code sequence is used to control the time of transmission. The transmitter turns on and off at time slots which are selected pseudorandomly within each time frame. The lower duty cycle of the transmitter implies faster chip rates. Thus, a bandwidth is obtained which is spread by a factor of x , where x equals the number of time slots in a given time frame. The advantage offered by time-hopping is that interfering signals must remain on all the time to jam the time-hopped signal. This implies that the jammer will require x times as much power as the transmitter. Normally, time-hopping is used in conjunction with other techniques (such as frequency-hopping) in order for the SS signal to exhibit LPI and other characteristics.

Chirp or Pulsed FM SS signals are signals that feature a change in transmitted frequency from the beginning to the end of the transmission period. The most common type of chirp is a linear chirp, either an up chirp (the frequency increases linearly during the transmission) or a down chirp (the frequency decreases). Although this technique is useful for communications, it has typically

found application with radar systems.

Hybrid SS techniques involve combining several SS techniques into one system. Examples of this include direct-sequence/frequency-hopped spread spectrum (DS/FH-SS) and direct-sequence/frequency-hopped/time-hopped SS (DS/FH/TH-SS), as well as any other combination or permutation of SS techniques. DS/FH modulation consists of a DS modulated signal whose centre frequency hops in a pseudorandom manner. An example block diagram of a FH/DS modulator is seen in Figure 2.1 along with the frequency spectrum of the FH/DS signal. These hybrid signals can provide excellent immunity to jamming and other interference, as well as presenting a low probability of interception. DS/FH techniques are being used in the Position Locating Reporting System (PLRS), currently under development for the U.S. military. DS/FH/TH signalling is being used in the JTIDS and a brief discussion of this can be found in Ziemer and Peterson [3, pp. 647-649].

Figure 2.1: a) Hybrid FH/DS-SS Modulator (redrawn from [2])
 b) Frequency Spectrum of Hybrid FH/DS-SS Systems

2.2.1 Direct-Sequence Spread Spectrum

One of the most popular of all spread spectrum techniques is that of direct-sequence. This is primarily because DS signals exhibit LPI, AJ and high resolution ranging and they are relatively easy to build (compared to say FH-SS). Interception of the spreading codes associated with DS-SS signals is the focus of this thesis and so the remainder of the thesis will focus on direct-sequence SS.

Direct-sequence SS signals are produced by phase-shift-keying (PSK) a carrier frequency at a high rate. The modulation can be either binary PSK (BPSK), quadrature PSK (QPSK), a higher level of PSK, or other forms of PSK, such as minimum shift keying (MSK). The signal of interest in this thesis is BPSK DS-SS.

A simplified diagram of a BPSK DS-SS transmitter and receiver can be seen in Figure 2.2. Figure 2.3 illustrates the DS spreading and despreading operation for BPSK data and spreading modulation which will now be briefly described. The constant envelope data-modulated carrier $s_d(t)$ of radian frequency ω_0 occupies a bandwidth equal to the data rate f_d (for square pulse data bits) where bandwidth implies the peak-to-null bandwidth. Spectral spreading of the BPSK signal is accomplished by multiplying $s_d(t)$ by the spreading code $c(t)$ of rate f_c resulting in the transmitted signal $s_t(t)$. The bandwidth is thus spread by a factor of f_c/f_d , also called the processing gain (PG) of the SS signal. The transmitted signal propagates via a distortionless path having delay T_d and is received together with some interference and/or channel noise (assumed to be a Gaussian in this thesis).

Demodulation is accomplished by mixing the received signal with the spreading code which has been delayed by an estimate of the delay $\hat{T}_d$. This correlation of the received signal with the delayed spreading waveform is called

(a)

(b)

Figure 2.2: BPSK Direct-Sequence Spread Spectrum Modulation
a) Transmitter b) Receiver (from [3])

Figure 2.3: BPSK DS Spreading and Despreading [3]

despreading. Determination of the delay estimate $\hat{T}_d$ is achieved through the use of synchronization circuitry which will be discussed later. This despread signal $c(t)s_t(t)$ is then phase demodulated to regain the data waveform.

A DS-BPSK signal has power spectra as shown in Figure 2.4. The spectrum has the characteristic $\text{sinc}^2 x = (\sin x / x)^2$ shape and the bandwidth (peak-to-null) is equal to f_c , the chip rate. Figure 2.4a shows the power spectrum without data modulation. This is simply the power spectrum for an m-sequence of frequency f_c and period N/f_c seconds consisting of spectral lines separated by f_c/N Hz and a bandwidth f_c . With data modulation applied, each spectral line has a $\text{sinc}^2 x$ superimposed upon it due to the data of bandwidth f_d . For the case where the data rate is less than the chip rate over the code period (i.e., $f_d < f_c/N$), the power spectrum appears as shown in Figure 2.4b. The chip spectral lines have $\text{sinc}^2(f/f_d)$ functions superimposed upon them. This is not normally seen in practice since a good spread spectrum design will ensure that the data bandwidth is large enough to achieve a constant envelope signal. Figure 2.4c demonstrates the typical case where $f_d > f_c/N$, whereby a continuous envelope is achieved. It is thus clear that an interceptor will not be able to determine the code chip rate by observing the power spectrum (when data has been applied) as has been cited in the open literature [2], and so alternate methods will be required for that task.

The best feature of DS-SS signals is their LPI characteristic which is demonstrated in Figure 2.5. Figure 2.5a shows the power spectrum of a voice signal with a narrow bandwidth of 25kHz. The spread version of this signal is shown in Figure 2.5b where the spectrum has been expanded. The spread signal has a peak amplitude which is equal to the unspeak peak amplitude divided by the processing gain (in other words, f_d/f_c times the amplitude of the unspeak signal). As the total

Figure 2.4: Ideal Power Spectra of DS-BPSK Signals

Figure 2.5: Real DS-BPSK Power Spectra [26]

energy contained in the signal remains the same, the power spectral density of the spread spectrum signal is much lower and even a modest amount of noise may make its detection very difficult at sufficiently great distances. Figure 2.5c illustrates a spread signal with a low SNR. SS signals with signal-to-noise ratios much less than zero dB are utilized in situations where LPI is desired and these signals will not be visible on a spectrum analyzer.

A major problem associated with SS communications is that of detection and synchronization [2]. In a typical spread spectrum receiver in start-up mode, only the code sequence may be accurately known, although estimates of the chip rate, carrier frequency and other parameters are known. Exact carrier frequency and chip rate are limited by the accuracy of the transmitter reference crystals and by Doppler effects. The relative code phase is also likely to be completely unknown. With these system characteristics it can prove difficult to determine that a transmission has occurred, let alone determine what information was sent.

An interceptor has a much more difficult problem. Initially, every parameter is unknown except perhaps what band of frequencies the signal can be found in. The parameters to be determined for an interceptor include carrier frequency, modulation format, chip and data rate, the spreading code and its relative phase shift and whether interleaving, error-correction coding or encryption are being used. The determination of many of the above unknowns will be detailed in Section 3.6.1. The parameter which will be determined using the techniques of this thesis is the spreading code and its phase.

The circuitry which can be utilized to determine the spreading code (for the interceptor) and to synchronize the relative phase shift (for friendly receiver and interceptor) is a correlator. It is the correlation process which forms the basis for

the SS processing gain and which is the heart of a spread spectrum system.

2.3 CORRELATION

In this thesis, two technologies for performing SS correlation have been looked into. These can be loosely termed digital correlation and acousto-optic (AO) correlation. AO correlation implies that an acousto-optic Time-Integrating Correlator is to be used whereas digital correlation encompasses other methods of performing correlation. In modern SS systems, this typically means correlation performed using a combination of analog and digital technology. However, older systems utilized analog technology to perform the correlation. These will not be discussed here. The following discussion will briefly describe digital correlation. After that, a general description of AO and AO correlation will ensue. Any differences between the techniques will be highlighted in the on-going discussion.

2.3.1 Digital Correlation

A block diagram of a typical correlation circuit used for code acquisition is shown in figure 2.6. The received SS signal is mixed with a reference pseudo-noise (PN) code and integrated. The integrated signal is sampled and compared to a threshold to determine if the code has been acquired. If not, then the reference PN code is phase-shifted and the process is repeated for the new relative phase shift. Assuming that the reference code is the same as the received code, then eventually the threshold will be crossed and the code phase will be acquired.

For these correlation receivers, the magnitude of the cross-correlation

Figure 2.6: Block Diagram of a Single Dwell Time PN Acquisition System with Non-Coherent Detection [1]

function

$$r_{xy}(\epsilon) = \int_0^T x(t)y(t+\epsilon)dt \quad (2.1)$$

represents the output of the integrator when the signal $y(t)$ is matched to the input $x(t)$. The locally generated PN code will be denoted $y(t)$ and the received code will be denoted $x(t)$. For the time being the received noise will be ignored.

The codes are periodic signals of period T consisting of sequences of rectangular pulses of duration T_c , i.e., $T = NT_c$. So the codes can be written as

$$x(t) = \sum_{n=-\infty}^{\infty} a_n p(t-nT_c) \quad (2.2)$$

for the received code, and

$$y(t) = \sum_{n=-\infty}^{\infty} b_n p(t-nT_c) \quad (2.3)$$

for the reference code. Since the codes are periodic then $x(t) = x(t+T)$ and $y(t) = y(t+T)$. If it is assumed that the relative phase shift ϵ is an integral multiple of T_c , then the parameter $r_{xy}(\epsilon)$ becomes

$$\begin{aligned} r_{xy}(\epsilon) &= \int_0^T \left[\sum_{n=-\infty}^{\infty} a_n p(t-nT_c) \right] \left[\sum_{n=-\infty}^{\infty} b_{n+\tau} p(t-nT_c+\tau T_c) \right] dt \\ &= T_c \sum_{n=-\infty}^{N-1} a_n b_{n+\tau} \end{aligned} \quad (2.4)$$

The periodicity of the sequence $\{b_n\}$ implies that

$$\{b_\tau, b_{\tau+1}, \dots, b_{\tau+N-1}\} = \{b_\tau, b_{\tau+1}, \dots, b_{N-1}, b_0, \dots, b_{\tau-1}\} \quad (2.5)$$

where the right-hand side of (2.5) is the τ^{th} cyclic shift of the original sequence

$$\{b_0, b_1, \dots, b_{N-1}\} \quad (2.6)$$

The function which will be considered for analysis in this thesis is the periodic *cross-correlation function* between the sequences $\{a_n\}$ and $\{b_n\}$ which is defined by

$$R_{ab}(\tau) = \sum_{n=0}^{N-1} a_n b_{n+\tau} \quad (2.7)$$

It is seen that

$$r_{xy}(\epsilon) = T_c R_{ab}(\tau) \quad (2.8)$$

when $\epsilon = \tau T_c$. Since ϵ is assumed to be an integral multiple of T_c , $R_{ab}(\tau)$ will be utilized in this thesis with $r_{xy}(\epsilon)$ easily determined using (2.8). If arbitrary pulse shapes or phase shifts are to be considered, the derivation must be performed again to determine $R_{ab}(\tau)$.

The periodic cross-correlation function $R_{ab}(\tau)$ described by (2.7) is the basis of the terminology "digital correlation". $R_{ab}(\tau)$ has been simulated extensively by the author on a computer for various m-sequences of different lengths. As well, the phase shifts τ were varied and many partial-period cross-correlations were calculated. The *partial-period cross-correlation* is defined as

$$R_{ab}(K,i,\tau) = \sum_{n=0}^{K-1} a_{n+i} b_{n+i+\tau} \quad (2.9)$$

where the cross-correlation is computed between K chips of $\{a_n\}$ and K chips of $\{b_n\}$, starting at position i , with $\{b_n\}$ cyclically shifted by τ chips.

An interceptor utilizing correlation techniques will want to use partial-period cross-correlations. This is so because a code search using full-period cross-correlation over all possible sequences will require far too much time to be successful. This will be discussed in Section 3.6.4.

2.3.2 Acousto-Optic Correlation

Acousto-optic (AO) technology can also be utilized to perform correlation between two codes. AO technology can satisfy requirements for the processing of

signals with large time-bandwidth products and high frequencies while consuming low power. As well, such systems as the acousto-optic Time-Integrating Correlator (TIC) developed at DREO allow the determination of many phases of a correlation function in parallel. The principal components of the DREO TIC are shown in Figure 2.7 with the diffraction geometry associated with the Bragg cell illustrated in Figure 2.8.

The principal component of any system utilizing AO is the Bragg cell. The transducers of a Bragg cell convert an electronic input at frequency f_0 into a travelling acoustic wave with velocity ν and wavelength $\lambda_a = \nu/f_0$. The laser light has wavelength λ_l in free space. Under certain conditions, the result of the acousto-optic interaction in the Bragg cell is the production of two significant beams outside of the cell: the undiffracted main beam and the principal diffracted beam. The information carried by the input signal at frequency f_0 is transferred to the principal diffracted beam which is offset from the undiffracted beam by an angle

$$\theta = 2 \sin^{-1} \left[\frac{\lambda_l f_0}{2\nu} \right]. \quad (2.10)$$

Typically, the argument of (2.10) is small and so the angle θ is proportional to the input frequency f_0 .

The correlator operation will now be described. The light from a laser is expanded and split into two paths by a beam-splitter. The two beams are then folded by mirrors and illuminated onto Bragg cells B_1 and B_2 . The RF signals produced by the antenna and the local code generator are applied to the transducers of the Bragg cells. The Bragg cells transfer the information carried by the RF signals to the two diffracted light beams which are mixed together by the beam mixer. This signal is imaged by a lens onto a photodetector array in such a way that the two signals are counter-propagating. Hence, each element of the

Figure 2.7: Time-Integrating Correlator [26]

Figure 2.8 Acousto-optical diffraction geometry for input at single frequency. (from [35]).

photodetector array performs a time-integration of the two signals. A build-up of energy will occur at the meeting point of the two signals if the signals are identical and a cancellation of energy will occur everywhere if the signals are different.

This process is illustrated in Figure 2.9 using two identical counter-propagating waveforms A and B and three elements of the photodetector array. The first column of Figure 2.9 shows successive positions of the counter-propagating signals. In the second column, the product $A \times B$ of the signals is demonstrated at the successive times and the result of the time-integration performed by the photodetector elements is shown in the third column. As a result it can be seen that the individual elements of the array correspond to different relative phase shifts between the two signals. Thus, detector 2 shows perfect correlation between the signals A and B and the other detectors demonstrate correlation where the signals have been time-offset.

The correlation function produced by the TIC is composed of

$$\int_T [A^2(t) + B^2(t) + 2A(t)B(t)] dt \quad (2.11)$$

where incidental optical and acoustic effects have been ignored. The integration of the third term can be recognized as the correlation between the two input signals. The A^2+B^2 terms contribute to the formation of a pedestal over which the correlation term appears. This pedestal function is removed as described below based on Figure 2.7. A frame of data for a positive correlation peak is collected and stored. Then a phase shift of 180° is applied to one of the signals and a second frame of data is collected using the same received input signal (recorded previously and repeated). These two frames of data are subtracted using digital post-processing hardware to remove the pedestal with an added result of an improved SNR on the correlation function. This is illustrated in Figure 2.10.

Figure 2.9: Correlation Process for AO Photodetectors [26]

Figure 2.10: Pedestal Removal Process for TIC [26]

A TIC has the advantage that many phases of the correlation function can be generated simultaneously. Since the transit time of the Bragg cells is about $40\mu\text{s}$ for the DREO TIC, then for a typical value of code rate of $f_c = 10\text{ MHz}$, it would be thought that the TIC could process 400 phases of the correlation function in parallel. However, due to the counter-propagating nature of the two signals, the TIC will process twice that or 800 phases of the correlation function in parallel. If the code is shorter than $40\mu\text{s}$, the entire correlation function will be visible in one frame. If the code is longer than $40\mu\text{s}$, only a $40\mu\text{s}$ window is visible at a time and many correlations with appropriate phase shifts will have to be performed in order to build the entire correlation function from the $40\mu\text{s}$ segments. The time integration is performed by a linear detector array having 2048 elements and the integration time can be controlled using the post-processing hardware.

Code search using a TIC involves successively correlating the received signal against various locally generated codes until a correlation peak is detected. The presence of a correlation peak indicates that the code from the code generator and the received code are identical. Correlation peaks have been detected with a reference signal at 0 dB and the unknown signal at -43 dB relative to the noise using an R16 with an integration time of 30 ms and a chip rate of 4 MHz [26].

It is clear from the above discussions that the acousto-optic correlation process can generate a correlation function much more quickly than digital correlation. For a code of length N and chip period T_c , a digital correlator will require approximately

$$N^2T_c + NT_s \quad (2.12)$$

seconds to calculate a full-period correlation function, where T_s is the time it takes for the digital correlator to switch code phase by one chip. A TIC would require

$$N^2T_c/M + N\bar{T}_s/M \quad (2.13)$$

seconds to obtain the same function where M is the number of phases calculated in parallel (typical value is 800) and $\bar{T}_s$ is the time for the AO correlator to switch code phase by M chips. Thus, the use of an acousto-optic TIC decreases the time requirement in a correlation by a factor of approximately M , if the switching times are assumed to be small compared to the code period.

In this thesis, digital correlation will be assumed to be used although the discussions could easily be applied to AO correlation. The only difference that is within the scope of this thesis is that the AO correlation would be performed M times as quickly as the digital correlation is.

CHAPTER 3

SPREAD SPECTRUM CODING

3.1 INTRODUCTION

The main thrust of this thesis involves a study of properties of the spreading codes used in DS-SS systems. Although it is important to understand how DS-SS works, the codes which are used in these systems can be dealt with as an independent issue. Henceforth, the remainder of the thesis will be limited to coding issues with some small mention of SS systems.

There is a large number of codes which can be used in direct-sequence spread spectrum systems. The choice of which code to use in a given DS-SS system depends on the system application. Gold codes and Bent sequences are useful in DS multiple-access systems while composite codes and JPL codes are used when rapid acquisition is essential. Non-linear feed-forward logic codes are recommended when security is essential. Short descriptions of the large variety of codes available will be undertaken and their utility will be highlighted.

The most common sequences used in SS systems are m-sequences, due to several of their properties which will be discussed later. Since these sequences are the basis for this thesis, the m-sequences will be dealt with separately from the other types of codes.

The chapter begins with a brief description of the types of spreading codes which are available so as to provide a starting point for future researchers. A review of the mathematics needed to analyze m-sequences will then be undertaken.

This is followed by a discussion of linear recursions and sequence generator fundamentals. Subsequently, a comprehensive discussion is undertaken of m-sequences and their properties. Finally, the problem of code interception will be introduced and previous techniques for the interception of m-sequences will be detailed.

3.2 CLASSIFICATION OF SPREAD SPECTRUM CODES

There is a large variety of codes available to spread a DS-SS signal depending upon the application required. These spreading codes are typically binary sequences generated in a pseudo-random fashion. That is, while they appear to be random sequences according to their statistics, they are in fact deterministic and are reproducible by an identical sequence generator. Due to their noiselike nature, spreading codes are also known as pseudo-noise or pseudorandom codes.

Spreading codes are generated with shift-register stages and can be classified into two main categories — linear codes and non-linear codes. Linear codes are generated using linear feedback operators, such as modulo-two adders and comprise a large set of code types. Non-linear codes are the result of using non-linear feedback or feed-forward elements, such as AND or OR gates. These codes are extremely difficult for the unintended recipient to reconstruct, and for this reason can be used to provide information security and excellent resistance to code-synchronous jamming techniques. Of course, for true message security it is expected that the message itself would be encrypted using techniques such as those discussed in [12].

To be useful as a spreading code for a DS-SS system, a code must meet several requirements. First, the code must have a full-period autocorrelation

function which has a sharp, narrow peak and a high peak-to-sidelobe ratio. This is necessary if proper synchronization of the receiver is to occur. Next, the code must have a length appropriate to the amount of spreading desired (i.e., to get the correct processing gain) and be long enough to be non-repetitive within the duration of a transmission. The code must be balanced or nearly balanced (equal number of zeros and ones) in order to suppress the chip rate from appearing on the power spectrum of the signal [2]. Also, if the DS-SS system will be operating in a multiple-access environment with other DS-SS users, then the codes must exhibit low cross-correlation properties in order to minimize interference.

Linear codes are probably the most commonly used type of codes. Common linear codes include m-sequences, Gold codes, and composite codes.

M-sequences are the most hardware efficient codes since they achieve the maximal length sequence available from a given number of shift-register stages. Their full-period autocorrelation properties are optimal as well, and for these reasons m-sequence codes are the most popular for non-multiple-access DS-SS systems. In most discussions on m-sequences such as [2] and [3] it is stated that m-sequences have little value when security is essential. However, this statement presupposes a noiseless environment which is not seen in practice. If m-sequences are used in a low SNR environment, it becomes very difficult to determine which m-sequence is being used. Therefore, m-sequences can be useful in an application even if security is essential. These codes will be detailed later in the thesis.

Gold codes [29] [30] are non-maximal length codes produced by the modulo-two addition of a preferred pair of m-sequences of equal period. They are attractive for multiple-access schemes since a large number of codes can be generated by a Gold code generator, and because selected sets of Gold codes have

good cross-correlation properties.

Composite codes [31] are produced by the modulo-two addition of two or more m -sequences whose lengths are relatively prime to each other. These codes are useful for rapid acquisition of long codes. However, they suffer from the problem that large secondary correlation peaks occur at multiples of the period of each component code giving intercept vulnerability and susceptibility to false synchronization in the presence of noise.

Non-linear codes are used for the most part to ensure message security in the sense that the message will not be susceptible to imitative jamming. Basically, a non-linear code is any code generated by a code generator which has non-linearities placed in either the feedback or feed-forward path. It is important to remember that any non-linear code has a linearly generated equivalent (although the hardware required would be considerably more complex).

Golomb [14] discusses many types of non-linear code generators including deBruijn sequences. DeBruijn sequences are sequences of period 2^n which achieve code balance. There also exists a set of codes known as Bent sequences [32]. These sequences are designed using a Fourier transform domain approach over a Galois field. By introducing non-linear feed-forward logic which has certain characteristics on an m -sequence generator, the result is balanced binary sequences with optimal correlation properties and high complexity. Groth [33] and Key [34] have also studied sequences obtained by employing different types of non-linear feed-forward logic on m -sequence generators. Unfortunately, little open literature exists on the topic of non-linear spreading codes as most information remains in the classified literature.

3.3 GALOIS FIELD THEORY

A field with a finite number of elements is called a *finite field*, or *Galois field*, and is denoted by $GF(q)$, where q is a prime. A field $GF(q^m)$ having q^m elements is called an *extension field* of the field $GF(q)$ having q elements. The *order* of a field is defined as the number of elements in the field. The Galois fields of interest for binary sequence generation are $GF(2)$ and its extension fields $GF(2^m)$, for $m \in \mathbb{N}$. The basic details of fields and finite field arithmetic can be found in [15].

Consider polynomials in x whose coefficients are elements of $GF(2)$. A polynomial of degree m over $GF(2)$ has the form $f(x) = f_0 + f_1x + f_2x^2 + \dots + f_mx^m$, where f_i is an element of $GF(2)$, that is, $f_i = 0$ or 1 . The *reciprocal polynomial* $f'(x)$ is defined as

$$f'(x) = x^m f(1/x). \quad (3.1)$$

Arithmetic operations are defined for these polynomials in exactly the same way as for polynomials with real coefficients except that the coefficients are operated on modulo-two.

A polynomial $f(x)$ over $GF(2)$ of degree m is said to be *irreducible* over $GF(2)$ if $f(x)$ is not divisible by any polynomial over $GF(2)$ of degree less than m . In fact, any irreducible polynomial over $GF(2)$ of degree m divides $x^{2^m-1} + 1$ [16]. An irreducible polynomial $f(x)$ of degree m is said to be *primitive* if the smallest positive integer n for which $f(x)$ divides $x^n + 1$ is $n = 2^m - 1$.

The number $N_1(d)$ of irreducible polynomials over $GF(2)$ of degree d is [1]:

$$N_1(d) = \frac{1}{d} \sum_{m/d} \mu\left[\frac{d}{m}\right] 2^m \quad (3.2)$$

where the sum is over all integer divisors m of d and $\mu(\cdot)$ is the Möbius function [1]:

$$\mu(i) = \begin{cases} 1 & i = 1 \\ (-1)^j & i = \text{product of } j \text{ distinct primes} \\ 0 & i \text{ divisible by a perfect square} \end{cases} \quad (3.3)$$

The number $N_p(d)$ of primitive polynomials over $GF(2)$ of degree d is [1]:

$$N_p(d) = \frac{2^d - 1}{2} \prod_{i=1}^J \frac{k_i - 1}{k_i} \quad (3.4)$$

where $k_i, i = 1, \dots, J$ are the distinct primes which divide $2^d - 1$.

The 2^m elements of a Galois field $GF(2^m)$ are the members of the set $\{0, 1, \alpha, \alpha^2, \dots, \alpha^{2^m-2}\}$ where α is a root of a primitive polynomial of degree m over $GF(2)$. The exponent of α^i is equal to $i \bmod 2^m - 1$ since $\alpha^{2^m-1} = 1$. These elements can be represented in two ways: the power representation as above and the polynomial representation. The polynomial representation is derived from the power representation such that the polynomial representation of α^i is equal to $\alpha^i \bmod p(\alpha)$ where $p(x)$ is the primitive polynomial from which the Galois field is generated.

The two representations for the elements of $GF(2^4)$ generated by the polynomial $p(x) = 1 + x + x^4$ are given in Table 3.1 (taken from [16]). Later it will be seen that the polynomial representation has a correspondence to the state of the m -sequence shift register generator. A shift by one of the generator is equivalent to multiplication by α of the state polynomial and so the state of the generator for any given shift can be calculated using this table.

For any $\alpha \in GF(2^m)$, let $m_\alpha(x)$ be the polynomial of smallest degree over $GF(2)$ such that $m_\alpha(\alpha) = 0$. This polynomial $m_\alpha(x)$ is called the *minimal polynomial* of α and it is irreducible.

There are $\phi(k)$ integers from 1 to k which are relatively prime to k , with $\phi(k)$ the Euler ϕ -function defined as:

Table 3.1: The elements of $GF(2^4)$ generated by $p(x) = 1 + x + x^4$ [16]

Power representation	Polynomial representation
0	0
1	1
α	α
α^2	α^2
α^3	α^3
α^4	$1 + \alpha$
α^5	$\alpha + \alpha^2$
α^6	$\alpha^2 + \alpha^3$
α^7	$1 + \alpha + \alpha^3$
α^8	$1 + \alpha^2$
α^9	$\alpha + \alpha^3$
α^{10}	$1 + \alpha + \alpha^2$
α^{11}	$\alpha + \alpha^2 + \alpha^3$
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$
α^{13}	$1 + \alpha^2 + \alpha^3$
α^{14}	$1 + \alpha^3$

$$\phi(k) = \begin{cases} 1 & \text{if } k = 1 \\ \prod_{i=1}^m k_i^{\alpha_i-1} (k_i-1) & \text{if } k > 1 \end{cases} \quad (3.5)$$

where k is a product of distinct primes,

$$k = \prod_{i=1}^m k_i^{\alpha_i}. \quad (3.6)$$

These $\phi(k)$ integers form an abelian group G under multiplication modulo- k . In this thesis, $k = 2^n - 1$. The set of numbers $H = \{ 1, 2, 4, \dots, 2^{n-1} \}$ forms a subgroup of G with n elements for $k = 2^n - 1$. This subgroup, when multiplied by any other element of the group, yields a *coset*.

Proper cyclotomic cosets, C_i , can be obtained by multiplying elements of H by an element r of G such that r is relatively prime to k . For example, with $k = 31$, the proper cyclotomic cosets are

$$C_1 = \{ 1, 2, 4, 8, 16 \}$$

$$C_2 = \{ 3, 6, 12, 24, 17 \}$$

$$C_3 = \{ 5, 10, 20, 9, 18 \}$$

$$C_4 = \{ 7, 14, 28, 25, 19 \}$$

$$C_5 = \{ 11, 22, 13, 26, 21 \}$$

$$C_6 = \{ 15, 30, 29, 27, 23 \}$$

The number of proper cyclotomic cosets is exactly $\phi(2^n-1)/n$ and the order of each proper coset is n . Note that the number of primitive polynomials (3.4) is equal to the number of proper cyclotomic cosets for the binary case. In other words, the number $N_p(N)$ of primitive polynomials over $GF(2)$ of degree N is

$$\begin{aligned} N_p(N) &= \phi(N)/L \\ &= \phi(2^L - 1)/L \end{aligned} \quad (3.7)$$

where $N = 2^L - 1$. A cyclotomic coset $C = rH$ is called *improper* if r is not relatively prime to k . In other words, if

$$\gcd(r, k) \neq 1 \quad \text{for } 1 \leq r \leq k. \quad (3.8)$$

For the example above, $C_0 = \{0\}$ is the only improper coset. Proper and improper cyclotomic cosets constitute cyclotomic cosets modulo $2^L - 1$. There are

$$Y(N) = -1 + \frac{1}{L} \sum_{d/L} \phi(d) 2^{L/d} \quad (3.9)$$

cyclotomic cosets in all. Group elements q and r belong to the same cyclotomic coset if and only if there exists some integer j , $0 \leq j \leq n-1$, such that

$$q \equiv 2^j r \pmod{2^n - 1}. \quad (3.10)$$

The smallest member of a cyclotomic coset C_i is called the *cyclotomic coset leader*.

Let C_i and C_j be cyclotomic cosets containing q and r respectively. Then, C_i is called the *inverse cyclotomic coset* of C_j if there exists some integer k , $0 \leq k \leq n-1$, such that

$$qr \equiv 2^k \pmod{2^n - 1}. \quad (3.11)$$

Let $GF(2)$ be a finite field contained in a larger field $GF(2^L)$. Then the trace polynomial $\text{Tr}_2^{2^L}(x)$ from $GF(2^L)$ to $GF(2)$ is defined as

$$\text{Tr}_2^{2^L}(x) \equiv \sum_{i=0}^{L-1} x^{2^i}. \quad (3.12)$$

The trace in $GF(2)$ of an element α in $GF(2^L)$ is defined as $\text{Tr}_2^{2^L}(\alpha)$, i.e., the trace polynomial evaluated at α . The trace function has the following useful properties which are proved in [1, App. 5A.9]:

$$1) \alpha \in GF(2^L) \Rightarrow \text{Tr}_2^{2^L}(\alpha) \in GF(2) \quad (3.13)$$

$$2) \text{Tr}_2^{2^L}(\alpha^{2^i}) = \text{Tr}_2^{2^L}(\alpha) \text{ for all } i \quad (3.14)$$

$$3) \text{ for any } a, b \in GF(2) \text{ and } \alpha, \beta \in GF(2^L),$$

$$\text{Tr}_2^{2^L}(a\alpha + b\beta) = a\text{Tr}_2^{2^L}(\alpha) + b\text{Tr}_2^{2^L}(\beta). \quad (3.15)$$

3.4 LINEAR RECURSIONS AND CODE GENERATORS

A sequence $\{s_n\}$ of elements from $GF(2)$ is said to satisfy a *linear recursion* if there exists a relation of the form:

$$s_n = \sum_{i=1}^L g_i s_{n-i} \quad (3.16)$$

where the coefficients g_i are also from $GF(2)$. This allows each sequence element to be calculated from the L immediately preceding elements. Assuming that $g_L \neq 0$, L is called the degree of the linear recursion.

Through appropriate manipulation (see [1]), it can be obtained that the characteristic polynomial of a sequence satisfying (3.16) has the form

$$g(x) = \sum_{i=0}^L g_i x^{L-i} \quad (3.17)$$

One convention [17] used in the literature expresses the characteristic polynomials in octal notation. For example, the polynomials $x^4 + x + 1$ and $x^6 + x + 1$ are represented by the binary vectors 10011 and 1000011, respectively. The octal notation for these polynomials is determined by grouping the binary representation into sets of three bits, starting from the right, and representing these sets in octal notation. Thus, the polynomials which are represented by the binary vectors 10011 and 1000011 are expressed in octal notation as 23 and 103, respectively.

The *period* N of the sequence $\{s_n\}$ with characteristic polynomial $g(x)$ is the smallest integer N such that $f(x)$ divides $x^N + 1$. This implies that $s_i = s_{i+N}$ for all possible values of i . When $N = 2^L - 1$, the polynomial $g(x)$ is said to be primitive. A sequence generated by a primitive polynomial is called a maximal-length shift register sequence or *m-sequence* since the period of the sequence is $2^L - 1$. Henceforth, only the *m-sequences* will be considered and an analysis made on $\{s_n\}$.

M-sequences of linear span n will sometimes be termed the Rn 's. The *reciprocal sequence* is generated by the reciprocal of the characteristic polynomial whereby the sequence $\{s_n\}$ is reversed to become $\{s_{-n}\}$.

Two architectures which are typically used to generate sequences described by the characteristic polynomial $g(x)$ are the Fibonacci and Galois architectures. Note that $g(x)$ is independent of architecture although this fact is confused in much of the literature where $g(x)$ for the same sequence is different depending on the generator architecture used. The two shift register sequence generator architectures are demonstrated in Figure 3.1. The Fibonacci form has the advantage that it is easy to implement in hardware and easy to perform hand calculations with. This is because only the leftmost register value need be calculated while the other register contents are simply shifted values. The Galois form has the advantage that the state of the shift register sequence generator can be calculated using Galois field theory. For example, if the characteristic polynomial of the sequence is $g(x) = 1 + x + x^4$, then the state of the Galois shift register can be calculated using the example in Table 3.1. Thus, the sequence and the states of the sequence generator will be as seen in Table 3.2 with the corresponding sequence generator given in Figure 3.2.

As a result, it is seen that the sequence corresponding to $g(x) = 1 + x + x^4$ is $\{s_n\} = \{000100110101111\}$. Also, the correspondence between the state of the shift register generator and the polynomial representation of the elements of a Galois field can be seen by comparing Tables 3.1 and 3.2. An observation made by the author is that if the initial state of the sequence generator is $\{1000\dots00\}$, then the output sequence will be identical for both the Fibonacci and Galois architectures. Other initial conditions will give the same sequence but with the

a) Fibonacci form

-

b) Galois form

Figure 3.1: M-Sequence Generator Architectures

Table 3.2: Shift register states and output for the sequence generated using $g(x) = 1 + x + x^4$

Cycle	Shift Register State	Output
0	1 0 0 0	0
1	0 1 0 0	0
2	0 0 1 0	0
3	0 0 0 1	1
4	1 1 0 0	0
5	0 1 1 0	0
6	0 0 1 1	1
7	1 1 0 1	1
8	1 0 1 0	0
9	0 1 0 1	1
10	1 1 1 0	0
11	0 1 1 1	1
12	1 1 1 1	1
13	1 0 1 1	1
14	1 0 0 1	1

Figure 3.2: Sequence Generator for $g(x) = 1 + x + x^4$

output of one generator equal to a shifted version of the other.

Another representation of an m -sequence over $GF(2)$ generated by (3.16) is given by

$$\{s_n\} = \{ \text{Tr}_2^{2^L}(\delta \alpha^n) \} \quad (3.18)$$

where α is the root of the minimal polynomial $m_\alpha(x)$ of the sequence $\{s_n\}$ and δ is some non-zero element in $GF(2^L)$. δ in (3.18) implies that some shift has been applied to the sequence. This representation is linked closely to the superposition of cosets method of Golomb [14] since all of the elements of a coset have the same trace value.

An m -sequence $\{s_n\}$ is said to be in *natural orientation* if $s_n = s_{2n}$ for all n , indices reduced modulo $2^L - 1$. The sequence will be in natural orientation when the initial conditions of the Fibonacci generator $s_0, s_1, \dots, s_{L-1}$ are given by

$$s_n = \text{Tr}_2^{2^L}(\alpha^n). \quad (3.19)$$

The importance of natural orientation will become apparent when the decimation properties of m -sequences are discussed in Section 3.5.1.

3.5 PROPERTIES OF M -SEQUENCES

The properties of m -sequences in the sections which follow are those which have been developed in the literature. This is an attempt to put all of these properties derived from many sources into one body of literature. Section 3.5 will discuss fundamental properties of m -sequences. Subsequent sections will focus on decimation, correlation and weight distribution properties of m -sequences.

As described in Section 3.4, an m -sequence is a sequence generated by a characteristic polynomial $g(x)$ which is primitive. The sequence is generated by an

L-stage linear-feedback shift register and has period $N = 2^L - 1$. The *linear span* of an m-sequence is defined as the number of shift registers required in the sequence generator, in this case denoted L. The chip sequence composed of elements of GF(2), i.e., 0 or 1, will be denoted $S = \{s_n\}$. In order to perform correlation calculations later, it is necessary that the elements of S be mapped onto $\{-1, +1\}$. To perform this mapping, the $\chi(\cdot)$ function is defined such that $\chi(0) = +1$, $\chi(1) = -1$, or equivalently, $\chi(\alpha) = (-1)^\alpha$ for $\alpha \in \{0, 1\}$. If S denotes an arbitrary $\{0, 1\}$ -valued sequence, then $A = \chi(S)$ denotes the corresponding $\{+1, -1\}$ -valued sequence, where the n^{th} element of A is $a_n = \chi(s_n)$.

Two sequences $\{s_n\}$ and $\{s'_n\}$, each of period N, are called *cyclically equivalent* if there exists an integer τ such that $s_{n+\tau} = s'_n$ for all n. A pair of m-sequences, which are not cyclically equivalent, are termed *cyclically distinct*. The number of cyclically distinct m-sequences is equal to the number of primitive polynomials, given by (3.6). Table 3.3 shows the number of m-sequences with linear span L and the corresponding code lengths as calculated by the author, as well as the maximum cross-correlation value. Note that each of [1], [2] and [14] had errors in their equivalent tables.

The properties of m-sequences will now be discussed. As stated before, the period of an m-sequence of linear span L is

$$N = 2^L - 1. \quad (3.20)$$

The m-sequence generator cycles through all $2^L - 1$ possible non-zero states. The balance property of m-sequences states that the number of ones in the sequence exceeds the number of zeros by one. In other words, the sequence consists of 2^{L-1} ones and $2^{L-1} - 1$ zeros. This property is important because unbalanced codes give rise to irregularities in the power spectral density of the SS signal which an

Table 3.3: The number and length of m-sequences of linear span L and the maximum cross-correlation magnitude

L	$N = 2^L - 1$	$N_P(L)$	$R_{\max}$
3	7	2	5
4	15	2	9
5	31	6	11
6	63	6	23
7	127	18	41
8	255	16	95
9	511	48	113
10	1 023	60	383
11	2 047	176	287
12	4 095	144	1 407
13	8 191	630	703
14	16 383	756	5 631
15	32 767	1 800	4 927
16	65 535	2 048	-
17	131 071	7 710	-
18	262 143	7 776	-
19	524 287	27 594	-
20	1 048 575	24 000	-

interceptor can take advantage of.

The run property of m -sequences states that within one period of an m -sequence there will be

$$2^{L-J} \text{ runs of } J \text{ consecutive ones, and} \quad (3.21)$$

$$2^{L-J} - 1 \text{ runs of } J \text{ consecutive zeros, where } 1 \leq J \leq L.$$

The shift-and-add property states that for any τ , $\tau \neq 0 \pmod{2^L - 1}$, the sum of the m -sequence $\{s_n\}$ and its τ -shift $\{s_{n+\tau}\}$ is another shift $\{s_{n+\delta(\tau)}\}$ of the same m -sequence. That is,

$$s_{n+\delta(\tau)} = s_{n+\tau} + s_n \quad \text{for all } n \quad (3.22)$$

where $\delta(\tau)$ is defined for all $\tau \neq 0 \pmod{2^L - 1}$. This property is useful to an interceptor because the data can be stripped off of a chip stream by exploiting the shift-and-add property. The shift-and-add property can also be used to verify that the sequence is an m -sequence.

3.5.1 Decimation

Let S denote an arbitrary m -sequence of period N , and consider the sequence $T = \{t_n\}$ formed by taking every q^{th} bit of S (i.e., $t_n = s_{qn}$ for all $n \in \mathbb{Z}'$). The sequence T is said to be the *decimation* by q of S and will be denoted by $S[q]$. Let $\gcd(a,b)$ denote the greatest common divisor of the integers a and b . If q divides N , then T has period N/q . If q and N are relatively prime (i.e., $\gcd(q,N) = 1$), then the period of T is also N . In general, the period of T is equal to $N/\gcd(q,N)$. A decimation is termed *proper* if $\gcd(q,N) = 1$ and *improper* otherwise.

If T is generated by a proper decimation and S is an m -sequence, then T is also an m -sequence (although S and T may be the same). If S is in natural

orientation and is properly decimated, then T is also in natural orientation. If $m_\alpha(x)$ is the generating polynomial of S and $T = S[q]$, then $m_{\alpha q}(x)$ is the generating polynomial of T , where $m_{\alpha q}(x)$ is determined as discussed in [1, App. 5A.8].

If $T = S[q]$ and $S = T[r]$, then q and r are termed *inverse decimations*. In other words, decimating sequence S by q and then by r will give back sequence S . q and r are inverse decimations if there exists some integer j , $0 \leq j \leq L-1$, such that

$$qr \equiv 2^j \pmod{2^L - 1}. \quad (3.23)$$

Examples of inverse decimations are given for the R8's in Table 3.4. The author is defining *reciprocal decimation* such that the decimation gives the m -sequence which is the reverse of the original m -sequence. In order for $S[q]$ and $S[r]$ to be reciprocal decimations, it is necessary that for some $x, y \in \mathbb{Z}^*$,

$$q \cdot 2^x + r \cdot 2^y \equiv 0 \pmod{2^L - 1} \quad (3.24).$$

If $S[q]$ and $S[r]$ is a reciprocal decimation pair, then $S[q] = S'[r]$ shall be the notation to designate such. If $S[q] = S'[q]$, then $S[q]$ is termed a *self-inverse*.

Two m -sequences T and U obtained by proper decimations q and r , respectively, of the same m -sequence S in natural orientation, are identical if q and r belong to the same cyclotomic coset. If S is not in natural orientation, then T and U are shifted versions of each other. In this thesis, sequences which are shifted versions of each other will be considered identical.

The $N_p(L)$ distinct m -sequences are generated by $N_p(L)$ distinct primitive polynomials of degree L . Given one m -sequence S of linear span L , the remaining m -sequences of linear span L can be obtained from S by decimations $r_2, r_3, \dots, r_{N_p(L)}$, where the r_i constitute all of the coset leaders for $GF(2^L)$. As an example, Table 3.4 shows the set of all m -sequence characteristic polynomials for the R8's and their corresponding decimations, as well as the reciprocal decimations.

Table 3.4: Characteristic polynomials and decimations for the R8's

Decimation	Characteristic Polynomial	Octal Rep'n	Recip/Inverse Decimation
S[1]	$x^8+x^4+x^3+x^2+1$	435	S[127] / S[1]
S[7]	$x^8+x^6+x^5+x^3+1$	551	S[31] / S[37]
S[11]	$x^8+x^7+x^6+x^5+x^2+x+1$	747	S[61] / S[29]
S[13]	$x^8+x^5+x^3+x+1$	453	S[47] / S[59]
S[19]	$x^8+x^6+x^5+x^2+1$	545	S[59] / S[47]
S[23]	$x^8+x^6+x^5+x+1$	543	S[29] / S[61]
S[29]	$x^8+x^7+x^3+x^2+1$	615	S[23] / S[11]
S[31]	$x^8+x^5+x^3+x^2+1$	455	S[7] / S[91]
S[37]	$x^8+x^6+x^4+x^3+x^2+x+1$	537	S[91] / S[7]
S[43]	$x^8+x^7+x^6+x+1$	703	S[53] / S[43]
S[47]	$x^8+x^7+x^5+x^3+1$	651	S[13] / S[19]
S[53]	$x^8+x^7+x^2+x+1$	607	S[43] / S[53]
S[59]	$x^8+x^6+x^3+x^2+1$	515	S[19] / S[13]
S[61]	$x^8+x^7+x^6+x^3+x^2+x+1$	717	S[11] / S[23]
S[91]	$x^8+x^7+x^6+x^5+x^4+x^2+1$	765	S[37] / S[31]
S[127]	$x^8+x^6+x^5+x^4+1$	561	S[1] / S[127]

It has been found [42] that certain m -sequences have shorter m -sequences imbedded inside them. An m -sequence of period $n_2 = 2^L - 1$ can be obtained by decimating an m -sequence of period $n_1 = 2^M - 1$ if n_2 divides n_1 . Thus, an m -sequence is imbedded if L divides M . Therefore, it is not necessary to calculate all m -sequences of all lengths since many m -sequences will be imbedded inside longer sequences.

A chart of imbeddedness has been derived by the author based on the above and is displayed in Figure 3.3. The chart demonstrates which m -sequences can be found by decimating longer sequences. The lines pointing to the right indicate which set of m -sequences can be derived from another larger set of m -sequences by decimation. For example, all of the R2's and R11's can be obtained from the R22's. The R16's can be decimated to give the R8's, which can be decimated to give the R4's and then the R2's. The reader is referred to [42] for details.

The relationship between sequences and their decimations is summarized using decimation relations [22]. On a decimation relation graph, the decimations required to generate one sequence from another are detailed. An example for the R5's is given in Figure 3.4 which consists of six sequences. Each side of the hexagon corresponds to a decimation by 3 if traversed clockwise and to a decimation by 11 if traversed counterclockwise. The dashed lines correspond to decimation by 5 if traversed clockwise and to decimation by 7 if traversed counterclockwise. Finally, traversing a diameter corresponds to decimation by 15. These decimation relations are important because they demonstrate that knowledge of one sequence gives complete knowledge of all of the possible m -sequences of the same length including the generating polynomial. Further uses of these graphs will be demonstrated in Chapters 4 and 5.

Figure 3.3: Chart of Imbeddedness up to R31

Figure 3.4: Decimation Relations for the R5's (redrawn from [22])

3.5.2 Correlation

A defining property of m -sequences is that the periodic autocorrelation function be two-valued. The autocorrelation function for m -sequences is defined as

$$\begin{aligned} R_a(\tau) &= \sum_{n=1}^N a_{n+\tau} a_n \\ &= \begin{cases} N, & \tau = 0 \bmod N \\ -1, & \tau \neq 0 \bmod N. \end{cases} \end{aligned} \quad (3.25)$$

The autocorrelation function for m -sequences is the ideal waveform for purposes of synchronization. An example autocorrelation function is plotted in Figure 3.5.

Unfortunately, when the autocorrelation is performed over less than a full period of the m -sequence, the ideal properties quickly disappear. The partial-period autocorrelation is defined as

$$R_a(K, n, \tau) = \sum_{j=0}^{K-1} a_{n+j+\tau} a_{n+j} \quad (3.26)$$

where the correlation is performed between two blocks of K bits from $A = \{a_n\}$, one block located τ bits from the other.

Whereas there is only one full-period correlation function, the partial-period correlation values depend on the position n where the computation of the correlation begins. Hence, there will be N different partial-period correlation functions depending on the value of n . A statistical description of these functions is necessary due to the large number of calculations required to characterize the partial-period correlation functions. The time-averaged mean and variance of the partial-period autocorrelation function are given by

$$\langle R_a(K, n, \tau) \rangle = \begin{cases} -K/N, & \tau \neq 0 \bmod N \\ K, & \tau = 0 \bmod N \end{cases} \quad (3.27)$$

Figure 3.5: The Autocorrelation Function for M-Sequences

and

$$\langle |R_a(K, n, \tau)|^2 \rangle = \begin{cases} K \left[1 - \frac{K-1}{N} \right], & \tau \neq 0 \bmod N \\ K^2, & \tau = 0 \bmod N \end{cases} \quad (3.28)$$

respectively, for $K \leq N$.

The full-period cross-correlation function has been defined previously in (2.7). However, the decimation properties of m-sequences shall now be exploited requiring a new definition. The full-period cross-correlation function between two $\{+1, -1\}$ -valued m-sequences $\{a_n\}$ and $\{b_n\} = \{a_{qn}\}$ where q is a proper decimation is defined as [20]

$$R_{ab}(\tau) = R_q(\tau) = \sum_{n=1}^N a_n a_{qn+\tau}. \quad (3.29)$$

This notation can be used since any m-sequence of a given length can be generated from any m-sequence of the same length by decimating properly. Thus, any pair of m-sequences can be described as a sequence $\{a_n\}$ and a decimated version of the sequence $\{a_{qn}\}$ generated by some proper decimation q .

The cross-correlation function is periodic implying $R_q(\tau) = R_q(\tau + N)$ where N is the period of the sequences being correlated. It has been proven in [38] that for all τ , $R_q(\tau)$ satisfies the relationship that $R_q(\tau) \equiv -1 \pmod{8}$ (i.e., $R_q(\tau) + 1$ is a multiple of 8), except when $\{a_n\}$ and $\{b_n\} = \{a_{qn}\}$ are reciprocal sequences. For the reciprocal sequences, $R_q(\tau) \equiv -1 \pmod{4}$ (i.e., $R_q(\tau) + 1$ is a multiple of 4). This is a very useful property since it implies that the still undefined weight distribution function will take on only a small set of values. In fact, the number of distinct values assumed by the cross-correlation function of two m-sequences, each of period N , can never exceed $Y(N)$, the number of cyclotomic cosets modulo N [14].

Theorem 3.1: The cross-correlation function, $R_q(\tau)$, depends on the cyclotomic coset to which q belongs and not on the specific value of q . [20] [24]

Proof: The cross-correlation function is defined

$$\begin{aligned} R_q(\tau) &= \sum_{n=1}^N a_n a_{qn+\tau} \\ &= \sum_{n=1}^N a_{n-\tau} a_{qn} \end{aligned}$$

But $a_{qn} = a_{rn}$ for all n when r is in the same cyclotomic coset as q , so that $R_q(\tau) = R_r(\tau)$ for all q and r in the same cyclotomic coset.

Thus, only the cyclotomic coset leaders are required in order to characterize all of the possible cross-correlation functions $R_q(\tau)$. From Theorem 3.1 and (3.29), it would appear that to characterize all possible cross-correlation functions requires only calculating a number of cross-correlation functions equal to the number of proper cyclotomic cosets. However, this is only true if all sequences are in natural orientation. For the code intercept problem, this is not likely to be true and so it may not be possible to exploit this fact in analyzing cross-correlation functions.

Nonetheless, assuming that the sequences are in natural orientation then the decimation relations can be utilized to determine which sequence pairs will give the same cross-correlation function. Re-examining Figure 3.4, it is clear which m -sequences will give the same cross-correlation results. For example, any vertices joined by dashed lines will give the same cross-correlation function $R_5(\tau)$.

If two proper decimations q and r are inverse decimations, i.e., they satisfy (3.23), then $R_q(\tau) = R_r(q\tau)$. This means that the correlation function corresponding to decimation by q can be found by decimating by q the correlation function corresponding to decimation by r . Thus, it is not necessary to calculate

both cross-correlation functions. For example, from the decimation relations of Figure 3.3, it is seen that $R_7(\tau) = R_5(5\tau)$. Thus, it can be assumed that any vertices on a decimation relation graph which are joined by like segments (segments which correspond to same decimation or its inverse) will have the same cross-correlation function subject to the requirement of natural orientation.

If $q = 2^{n-1} - 1$, then $R_q(\tau)$ is the cross-correlation function between an m -sequence and its reverse sequence (q is a reciprocal decimation). Niho [18] discusses the bounds on $R_{-1}(\tau)$ (as the cross-correlation function of the reciprocal sequences is denoted) which are

$$-2^{(L+2)/2} + 1 \leq R_{-1}(\tau) \leq 2^{(L+2)/2} + 1 \quad (3.30)$$

and since $2^{(L+2)/2} \equiv 0 \pmod{4}$ for L even, we have

$$-2^{(L+2)/2} + 4 \leq R_{-1}(\tau) \leq 2^{(L+2)/2} + 1, \quad L \text{ even} \quad (3.31)$$

Denoting L_L as the number of distinct values that $R_{-1}(\tau)$ takes on, then the results of (3.30) and (3.31) are summarized in Table 3.5. These bounds are tight for $L \leq 18$ with the verified maximum and minimum values of $R_{-1}(\tau)$ displayed in the table. Note that the values of L_L indicate that $R_{-1}(\tau)$ takes on all of the possible values between the maximum and minimum values (all values such that $R_{-1}(\tau) \equiv -1 \pmod{4}$ in that range).

It can be shown [22] that

$$\sum_{l=0}^{N-1} R_q(\tau) = 1. \quad (3.32)$$

Thus, for long sequences the average value of $R_q(\tau)$ closely approaches zero.

Similarly, it can be demonstrated [22] that

$$\begin{aligned} \sum_{l=0}^{N-1} [R_q(\tau)]^2 &= N^2 + N - 1 \\ &= 2^{2L} - 2^L - 1 \end{aligned} \quad (3.33)$$

**Table 3.5: Cross-correlation values of reciprocal sequences
(from [18])**

L	$2^{(L+2)/2}$	$\max\{R_{-1}(\tau)\}$	$\min\{R_{-1}(\tau)\}$	L_L
3	5.7	4	-4	3
4	8	8	-4	4
5	11.3	12	-8	6
6	16	16	-12	8
7	22.6	20	-20	11
8	32	32	-28	16
9	45.3	44	-44	23
10	64	64	-60	32
11	90.5	88	-88	45
12	128	128	-124	64
13	181.0	180	-180	91
14	256	256	-252	128
15	362.0	360	-360	181
16	512	512	-508	256
17	724.1	724	-720	362
18	1024	1024	-1020	512

Note that (3.32) and (3.33) are independent of the actual m-sequences correlated and are simply functions of the code period N .

The maximum value which $R_q(\tau)$ can take has been bounded by Sidelnikov [39] such that for at least one integer τ ,

$$R_{\max} = | R_q(\tau) | > -1 + 2^{(n+1)/2}. \quad (3.34)$$

Actual values for $R_{\max}$ have been determined by the author for $L = 3$ to 15 and have been displayed in Table 3.3 previously.

3.5.3 Weight Distribution

It is not necessary in all applications to know the value of $R_q(\tau)$ at each value of τ . In cases where the sequence is received from a noisy channel and correlated, the values at individual values of τ are likely to be inconsistent. It is valuable to know the set of values taken on by $R_q(\tau)$ and the number of occurrences of each value. This is referred to as the cross-correlation spectrum [22] or as the weight distribution of the cross-correlation function [18], shortened to weight distribution. The terminology of weight distribution will be adopted here and shall be represented by the symbol $\eta[R_q(\tau)]$. An example to demonstrate notation will be given for the weight distribution of a periodic autocorrelation function which is

$$\eta[R_q(\tau)] = \begin{cases} 1, & R_q(\tau) = N \\ N - 1, & R_q(\tau) = -1 \end{cases} \quad (3.35)$$

This weight distribution function shows that $R_q(\tau) = N$ occurs once and $R_q(\tau) = -1$ occurs $N - 1$ times. Another notation which will be required later is that $\eta[R_q(\tau)]_i$ is the i^{th} element of the weight distribution corresponding to the number of occurrences of the i^{th} cross-correlation magnitude $[R_q(\tau)]_i$. These will be shortened

to n_1 and r_1 , respectively, for reasons which will become apparent. For the example above, $n_1 = \eta[R_q(\tau)]_1 = 1$ for $r_1 = [R_q(\tau)]_1 = N$ and $n_2 = \eta[R_q(\tau)]_2 = N - 1$ for $r_2 = [R_q(\tau)]_2 = -1$.

Some general properties of the weight distribution function are as follows. The weight distribution function is invariant with phase shift of the sequences being cross-correlated. The weight distribution function for two m -sequences which are related through a given decimation q is the same as the weight distribution function for any other pair of m -sequences related through the same decimation q . Thus, the weight distribution function depends only upon q and not upon individual m -sequences. However, the individual m -sequences which were used can be determined given knowledge of one of the m -sequences. Finally, inverse decimations give rise to the same weight distribution function.

Considerable effort has been expended in the literature to determine the decimations which give a weight distribution function with a specified number of values. These results will be presented here extracted from Niho [18] in brief. The interested reader is directed to the dissertation for more details. Niho presented an appendix which contains all of the weight distributions for $L = 3$ to 12. For $L = 13$ to 15, only the restricted set of decimations which lead to 7 or less distinct cross-correlation values were calculated. For $L = 16$, only those decimations which lead to 5 or less distinct cross-correlation values were determined. The author has extended this work and calculated all weight distributions for $L = 3$ to 15 with a capability to extend further given significant time and computing resources. The author also generated all of the proper decimations for $L = 3$ to 25 and calculated all of the inverse decimation pairs for these values of L .

The weight distributions which Niho discussed will now be listed.

- 1) If $q = 2^k + 1$ or $q = 2^{2k} - 2^k + 1$ and L/e is odd where $e = \gcd(L, k)$, then the weight distribution function is

$$\eta[R_q(\tau)] = \begin{cases} 2^{L \cdot e - 1} + 2^{(L \cdot e - 2)/2}, & R_q(\tau) = -1 + 2^{(L \cdot e)/2} \\ 2^L - 2^{L \cdot e} - 1, & R_q(\tau) = -1 \\ 2^{L \cdot e - 1} - 2^{(L \cdot e - 2)/2}, & R_q(\tau) = -1 - 2^{(L \cdot e)/2} \end{cases} \quad (3.36)$$

- 2) For L odd and $q = (2^{L \cdot k} + 1)/(2^k + 1)$, $\eta[R_q(\tau)]$ is given by (3.36).
 3) If $L \equiv 0 \pmod{4}$, $L = 2m$ and $q = 2^{m-1} - 1$, $\eta[R_q(\tau)]$ is a 4-valued function,

$$\eta[R_q(\tau)] = \begin{cases} (2^{2m-1} - 2^{m-1})/3, & R_q(\tau) = 2^{m-1} - 1 \\ 2^m - 1, & R_q(\tau) = 2^m - 1 \\ 2^{2m-1} - 2^{m-1}, & R_q(\tau) = -1 \\ (2^{2m} - 2^m)/3, & R_q(\tau) = -2^m - 1 \end{cases} \quad (3.37)$$

- 4) If $L = 4m$ and $q = (2^m - 1)(2^{2m} + 1) + 2$, $\eta[R_q(\tau)]$ is a 4-valued function,

$$\eta[R_q(\tau)] = \begin{cases} 2^m, & R_q(\tau) = 2^{3m} \\ 2^{4m-1} - 2^{3m-1}, & R_q(\tau) = \pm 2^{2m} \\ 2^{3m} - 2^{2m} - 1, & R_q(\tau) = -1 \end{cases} \quad (3.38)$$

- 5) If $L = 2m$ and $q = 2^m + 3$, $\eta[R_q(\tau)]$ is at most a 5-valued function,

$$\eta[R_q(\tau)] = 2^m(j-1) - 1, \quad 0 \leq j \leq 4 \quad (3.39)$$

- 6) If $L \equiv 2 \pmod{4}$, $L = 2m$ and $q = 2^m + 2^{m-1} - 1$, $\eta[R_q(\tau)]$ is at most a 6-valued function,

$$\eta[R_q(\tau)] = 2^m(j-1) - 1, \quad 0 \leq j \leq 5 \quad (3.40)$$

- 7) If $L = 2m$ and $q = 2^{m+2} - 3$, $\eta[R_q(\tau)]$ is at most an 8-valued function,

$$\eta[R_q(\tau)] = 2^m(j-1) - 1, \quad 0 \leq j \leq 7 \quad (3.41)$$

- 8) If $L = 2m$ and $q = 2^{m+2} + 2^m - 3$, $\eta[R_q(\tau)]$ is at most a 9-valued function,

$$\eta[R_q(\tau)] = 2^m(j-1) - 1, \quad 0 \leq j \leq 8 \quad (3.42)$$

- 9) If $q = (2^{5k} + 1) / (2^k + 1)$ and L/e is odd where $e = \gcd(L, k)$, then $\eta[R_q(\tau)]$ is at a most 5-valued function

$$\eta[R_q(\tau)] = -1, \pm 2^{(n+e)/2} - 1, \pm 2^{(n+3e)/2} - 1 \quad (3.43)$$

Niho [18] also made several conjectures which bounded the number of distinct cross-correlation values in a weight distribution function for given decimations but these will not be presented.

3.6 SPREAD SPECTRUM CODE INTERCEPTION

3.6.1 Introduction

In order to decode message traffic, it is necessary to detect, intercept and exploit the SS signal. The problem of detection will not be treated in this thesis. Suffice it to say that many techniques exist to detect SS signal energy such as radiometers. As well, the exploitation of SS signals will not be treated here as this topic is dependent on the information contained within the signal and the intended use of this information. The concern in this thesis will be on the interception of the SS spreading code and the determination of the specific code being used. This determination will only be performed on m-sequences at this time.

Assumptions which will be made in performing the code interception will now be detailed and justified. First of all, the SS signal bandwidth is known and the modulation scheme is known to be BPSK DS-SS. The carrier frequency has been found using a frequency multiplier [4][35]. The chip rate has been found using a chip rate detector [4]. The chip rate detector also yields the input SNR as can be seen in the example curve of Figure 3.6. As well, it will be assumed that the data

Figure 3.6: Plot of $(SNR)_{in}$ vs $(SNR)_{out}$ for Radiometer and Doubler [4]

T = integration time, W = intercept bandwidth (see p157 [4])

sequence has been removed. This can be done using the shift-and-add property of m-sequences. By recording a received sequence, shifting by one chip and adding the shifted sequence to the original received sequence, the data is effectively removed except at the chips where data transitions occurred. Finally, it will be assumed there is no interleaving, error-correction coding or encryption on the baseband data in order to ensure these cannot affect the problem. As well, it is assumed that the code length is known. Thus, the interception problem of this thesis is to determine which m-sequence is being utilized to spread the SS waveform.

Several approaches have been presented in the literature to solving the code search problem. The three families of methods which will be discussed here are the decoding methods, the null filtering technique and the autocorrelation techniques. These methods will be explained and their shortcomings will be detailed. The justification for the technique developed in this thesis will become clear upon completion of this section.

3.6.2 Decoding of Intercepted M-Sequences

Decoding techniques involve operating directly on the received sequence on a chip by chip basis. The term decoding techniques is derived from the fact that each chip must be decoded individually before code search can be performed. For BPSK modulation the probability of error in an additive white Gaussian noise (AWGN) channel is well known. This is displayed in Figure 3.7 which clearly demonstrates that decoding methods will only work if the input SNR (implies E_c/N_o) is greater than about 5 dB.

Assuming the SNR allows reception of error-free chips, then the decoding

Figure 3.7: Probability of Error for Binary Signals [10]

methods work in the following way. Assume that the linear span of the m -sequence L is known and that $2L$ consecutive sequence elements $s_j, s_{j+1}, \dots, s_{j+L-1}$ have been received. Then, the fact that the m -sequence is generated by a linear recursion (3.16) can be used to determine the generating polynomial. This is so because the recursion is linear in the L unknown coefficients a_i .

Thus the decoding techniques involve the solution of L equations in L unknowns. There are many techniques available to solve simultaneous linear equations including substitution and linear algebra methods such as Gaussian elimination and Cramer's Rule [37].

As an example of the decoding techniques, suppose that a binary sequence is known to satisfy a linear recursion of degree 5 and furthermore, a received portion of the sequence is

$$\dots 101101010001\dots \quad (3.44)$$

where the first bit is on the left and subsequent bits are on the right. Applying (3.16) to overlapping segments of the sequence yields the following set of simultaneous linear equations:

$$\begin{aligned} 1 &= a_2 + a_3 + a_5 \\ 0 &= a_1 + a_3 + a_4 \\ 1 &= a_2 + a_4 + a_5 \\ 0 &= a_1 + a_3 + a_5 \\ 0 &= a_2 + a_4 + a_5 \end{aligned} \quad (3.45)$$

Now, for binary m -sequences it is known that $a_L = 1$ implying in this case that $a_5 = 1$. Thus only $L - 1$ equations are required to be solved. In this example, the first four equations of (3.45) can be rewritten as

$$\begin{aligned}
0 &= a_2 + a_3 \\
0 &= a_1 + a_3 + a_4 \\
0 &= a_2 + a_4 \\
1 &= a_1 + a_3
\end{aligned} \tag{3.46}$$

Since addition in GF(2) is modulo 2, adding the first three equations gives $a_1 = 0$ and adding the last three gives $a_2 = 1$. Substitution of these values gives $a_3 = 1$ and $a_4 = 1$, and hence the resulting recursion is

$$s_n = s_{n-2} + s_{n-3} + s_{n-4} + s_{n-5} \tag{3.47}$$

which implies that the generating polynomial of the sequence is

$$g(x) = 1 + x + x^2 + x^3 + x^5 \tag{3.48}$$

If the assumed L is smaller than the actual linear span L , then the solution will be incorrect. To rectify this problem, simply assume that L is larger and solve using more chips.

It can be seen that the above technique becomes rather tedious for large values of L (e.g., $L > 10$). Another technique which solves for the generating polynomial in reasonable time and which is well suited for computer programming is the Berlekamp algorithm.

3.6.2.1 Berlekamp's Algorithm

The Berlekamp algorithm is normally used for the decoding of Bose, Chaudhuri, and Hocquenghem codes in an error-correction capacity. However, the algorithm has been adapted by Massey [36] and corrected and further modified for this thesis to solve for the generating polynomial of m -sequences. It will be demonstrated in its modified form to solve for the generating polynomial of

m-sequences. This technique requires any $2L$ consecutive bits of the m-sequence as input and will output the generating polynomial. The method was programmed by the author in C for the IBM PC AT and used throughout the course of this thesis. The details of the modified Berlekamp algorithm are given below.

Assume that the linear span of the intercepted m-sequence is L and that $2L$ consecutive bits have been intercepted $s_0, s_1, s_2, \dots, s_{2L-1}$. It is desired to determine the generating polynomial $g(x)$ which corresponds to the intercepted sequence. The Berlekamp algorithm yields the reciprocal of the generating polynomial which is

$$g'(x) = 1 + c_1x + c_2x^2 + \dots + c_{L-1}x^{L-1} + x^L \quad (3.49)$$

The generating polynomial can be found from $g'(x)$ using the formula

$$g(x) = x^L \cdot g'(1/x) \quad (3.50)$$

The steps of the algorithm are as follows:

STEP 1: set $g'(x) = B(x) = 1$, $n = M = 0$, $j = 1$

STEP 2: if $n = 2L$, then stop. Otherwise, compute

$$d = s_n \odot \sum_{i=1}^M c_i s_{n-i}$$

STEP 3: if $d = 0$, then $j = j + 1$, and go to 6.

STEP 4: if $d = 1$ and $2M > n$, then

$$g'(x) = g'(x) \odot x^j \cdot B(x)$$

$j = j + 1$, and go to 6.

STEP 5: if $d = 1$ and $2M \leq n$, then

$$T(x) = g'(x) \quad [\text{temporary storage of } g'(x)]$$

$$g'(x) = g'(x) \odot x^j \cdot B(x)$$

$$M = n + 1 - M$$

$$B(x) = T(x)$$

$$j = 1$$

STEP 6: $n = n + 1$, and go to 2.

The generating polynomial of the m-sequence upon completion of the algorithm is $g(x)$ which is found using equation (3.50). If the assumed linear span L is too small then the result will be incorrect. To resolve the problem, reapply the algorithm with a larger assumed L and with the larger set of $2L$ consecutive bits until a solution is achieved which is consistent with the intercepted chip stream.

Table 3.6 shows the results for the application of the Berlekamp algorithm to the binary sequence given in (3.44).

Table 3.6 indicates that the generating polynomial corresponding to the chip sequence of (3.44) is

$$g(x) = x^5 \cdot g'(1/x) \quad (3.51)$$

$$= 1 + x + x^2 + x^3 + x^5 \quad (3.52)$$

which is identical to that found using the substitution method. Although this method is tedious manually, it is clear that the use of a computer makes this algorithm very useful.

Unfortunately, the Berlekamp algorithm and other decoding techniques will encounter difficulties if any of the consecutive bits are in error or if data transitions occurred during the sequence of bits. Therefore, for SNRs less than about 5 dB it is very likely that these techniques will fail. Since intercept receivers will typically have input SNRs less than 5 dB, the decoding techniques have little value.

Table 3.6: Example of the application of the Berlekamp algorithm

n	M	$g'(x)$	j	$B(x)$	s_n	d
0	0	1	1	1	1	1
1	1	$1+x$	1	1	0	1
2	1	1	2	1	1	1
3	2	$1+x^2$	1	1	1	1
4	2	$1+x+x^2$	2	1	0	0
5	2	$1+x+x^2$	3	1	1	0
6	2	$1+x+x^2$	4	1	0	1
7	5	$1+x+x^2+x^4$	1	$1+x+x^2$	1	1
8	5	$1+x^3+x^4$	2	$1+x+x^2$	0	1
9	5	$1+x^2$	3	$1+x+x^2$	0	1
10	5	$1+x^2+x^3+x^4+x^5$	4	$1+x+x^2$	—	—

3.6.3 Null Filtering

Null filtering [43] is a variant of the decoding methods which is performed using hardware rather than software analysis as in the techniques above. The null filtering technique involves feeding the intercepted sequence into a sequence generator which has feedback taps set up in some manner. If the output is all zeros, then the intercepted sequence is generated by a sequence generator which is the same as that used in the null filter. Null filtering is likely the fastest of all decoding techniques since speed is limited by hardware rather than software.

A suggested architecture for the null filter sequence generator is given in Figure 3.8. This null filter architecture is based on the reasoning which follows. If it is assumed that the intercepted sequence is generated by the linear recursion

$$s_n = \sum_{i=1}^L g_i s_{n-i} \quad (3.53)$$

then adding the intercepted sequence to the null filter is equivalent to adding s_n to both sides of (3.53) which results in

$$0 = \sum_{i=0}^L g_i s_{n-i}, \quad (3.54)$$

so that an output of all zeros should result if the assumed linear recursion is correct. In Figure 3.7, the connections are derived from the equations above to be:

$$\begin{array}{ll} c_i \text{ is closed} & \text{if } g_i = 1 \\ c_i \text{ is open} & \text{if } g_i = 0, \end{array}$$

and the initial conditions of the null filter sequence generator should be all-zeros.

Whether or not the intercepted sequence is matched by the generator, the first L output bits will be non-zero. If the sequence matches the generator, the remainder of the output bits will be zero. Otherwise, the remainder of the output

Figure 3.8: Null Filter Sequence Generator Architecture

bits will be non-zero indicating an incorrect sequence generator. After this condition has been detected and verified, the sequence generator should be reconfigured and the intercepted sequence should be fed into the new generator. The null filtering technique involves stepping through all possible generator configurations and feeding the intercepted sequence into these generators until an all-zero condition is detected.

One problem with the null filtering technique is that of error propagation. One bit error or a data transition will generate an error pattern on the output of the null filter of $L+1$ bits. Fortunately, this can be corrected for by including detection and verification circuitry to analyze the bit stream at the output of the null filter sequence generator.

However, as with the decoding methods the effectiveness of this technique only extends down to input SNRs of about 5 dB due to an increasing probability of bit error in low SNR environments. Thus, a technique which will work in low SNR environments is still required.

3.6.4 Autocorrelation Techniques

The correlation techniques to be discussed are used for the determination of m -sequences and are based on sequentially searching through all possible codes until an autocorrelation peak is detected. Assume that a DS-SS signal has been intercepted and that all parameters have been determined except for the specific m -sequence used in the spreading operation. Then the autocorrelation technique involves correlating the unknown sequence against all possible m -sequences of the same length until the detection threshold has been crossed. The variants of this

technique are that either full-period or partial-period correlation can be performed. These will be discussed briefly.

If full-period correlation is used, then the correlator must correlate for N chips over N phases to obtain one correlation function (i.e., N^2 calculations). There are $N_p(L)$ distinct m -sequences of linear span L and length $N = 2^L - 1$. The autocorrelation peak will only occur when the reference code is equal to the unknown code and even then, only for one relative phase shift. Thus, on average the full-period correlation technique will require $N^2 \cdot N_p(L)/2$ calculations to obtain the autocorrelation peak. As stated in Section 2.3.2, autocorrelation peaks have been detected with the reference code at 0 dB and the unknown signal at -43 dB relative to the noise. So, while this technique requires a large number of calculations, it can work for very low SNR environments, unlike decoding techniques.

To alleviate the problem with this large number of calculations, partial-period correlation can be performed instead of full-period correlation. For each relative phase shift between the unknown sequence and a locally generated reference sequence, the correlator is only activated for a small percentage of a full code period, say 0.1% of the sequence length. At the correct relative phase shift, and using the reference sequence which is identical to the unknown, the resultant autocorrelation peak should be 0.1% of the full-period autocorrelation peak. However at other phase shifts of the correct sequence, the correlation magnitude will no longer be -1 but will appear to be a noise-like process. Work has been performed previously which has analyzed partial-period autocorrelation and the interested reader is referred to [19],[40] and [41] for details.

For a reference sequence which is not identical to the unknown sequence, the

results of the correlation will be noise-like as well. When the received noise is included and any noise due to the correlation circuitry is accounted for, it is clear that the partial-period correlation method is quite complex to perform an analysis on. This is probably why it is not found anywhere in the literature and will remain so. The problem for the correlation methods becomes one of setting an optimum threshold such that the autocorrelation is detected and secondary peaks are rejected. The noise on the unknown sequence will complicate this threshold setting and will likely to cause a fairly high false-alarm rate. The threshold would have to be adaptive depending on the received SNR and the false-alarm and detection probabilities would also be functions of the SNR.

Thus, a technique could be developed based on the partial-period correlation and this technique would probably require multiple-dwell threshold detection. Multiple-dwell would allow rapid rejection of incorrect correlations and allow concentration of effort on those correlation values which have exceeded the threshold. This has not been examined in the literature and could form the basis for a substantial body of work.

For simplicity, it will be assumed that the partial-period correlation technique simply employs a 3-out-of-5 detection scheme and thus requires

$$\begin{aligned} & (N^2 \cdot N_p(L)/2) \times 0.001 \times 5 \\ & = N^2 \cdot N_p(L) / 400 \end{aligned} \tag{3.55}$$

calculations.

A comparison of the amount of time required to determine the unknown code has been made and is displayed in Figure 3.9. Figure 3.9 demonstrates the amount of time required for code search as a function of code length. It is assumed that the code generators are operating at 10 MHz and that the SNR is not very low (say

Figure 3.9: Code Search Time for the Correlation Techniques
with $f_c = 10$ MHz

SNR > -5 dB). It is seen that these methods require an inordinate number of calculations for reasonable code lengths. The code search time for the partial-period correlation using the AO TIC is also displayed resulting in a down shift of the curve but still demonstrating that the code search time is quite large.

Both of the full-period and partial-period autocorrelation techniques suffer from the problem that each correlation function calculation gives a result which is independent of the others. In other words, the correlation function determined using one reference sequence does not give any information which can be used when determining the correlation function with another reference sequence. It will be seen in Chapter 5 that the interception technique developed in this thesis uses the results from one reference sequence correlation to aid with other correlations. This utilization of cross-information helps to speed the code search even further as will be demonstrated later.

CHAPTER 4

A WEIGHT DISTRIBUTION ANALYSIS TECHNIQUE

4.1 INTRODUCTION

There are clearly shortcomings with existing code interception techniques which need to be addressed. Decoding techniques will only work for high input SNRs ($\text{SNR}_{\text{in}} > 5 \text{ dB}$) and correlation techniques require a large number of calculations. It is necessary to develop an interception technique which will produce correct results in low SNR environments and which can be performed in a reasonable time. A technique which meets both requirements has been developed by the author and will now be derived.

The technique is based on an analysis of the weight distribution of the cross-correlation function between two m -sequences in noise. It has been theoretically determined that this technique will work for input SNRs down to about -12 dB which is a significant improvement over decoding techniques. As well, the technique requires orders of magnitude fewer calculations than correlation techniques. The improvement in speed is primarily due to some properties of m -sequences which have been discovered by the author. The properties which have been observed include various decimation properties and the concept of code families which will now be explained. By exploiting these properties, speed improvements of orders of magnitude can be effected.

The concept of code families has been developed by the author and will be

explained in the next section. The exploitation of this concept and the decimation properties will be discussed in Section 4.3 which applies these concepts to the interception of m -sequences using cross-correlation analysis. The final section will explain the concept of weight distribution analysis and apply all of the above to this analysis. The weight distribution analysis technique and its performance in noise will be derived as well. The application of this analysis in the form of a code search strategy will be discussed in Chapter 5.

4.2 CODE FAMILIES

The concept of code families has not appeared in the literature previously. This concept has been developed by the author and while basic in nature, it is important to the code search strategy based on weight distribution analysis.

It is known that the weight distribution function is the same for all m -sequences of the same length which are related by the same decimation. The inverse decimation also gives the same weight distribution function. These have been found by Niho [18] for m -sequences with $L = 3$ to 12, and extended by the author to $L = 15$. The set of all sequence pairs which satisfy a given decimation and its inverse will be termed an inverse pair group and will be described by the inverse decimation pair and its corresponding weight distribution function. The members of an inverse pair group also have the same cross-correlation function if both sequences in a pair are in natural orientation.

However, it has been found that different inverse pair groups have the same weight distribution in some cases. The set of all inverse pair groups which have the same weight distribution function is called a code family, i.e., the number of code

families equals the number of unique weight distribution functions for a given R_n . The concept of code families is important because these code families introduce some ambiguity into the results of a code search and will force an interceptor to perform at least two correlation calculations.

The code families have been found for the R_3 's to R_{15} 's through extensive simulation. These calculations consumed approximately one month of continuous processing time on a 20 MHz Ogivar 80386-based computer. A sample output is given in Figure 4.1 which shows the code families for the R_7 's and R_8 's. The output is formatted such that the inverse decimation pairs are given as $a_1[b_1]$, $a_2[b_2]$, ... with the corresponding weight distribution given in the form $x_1(y_1)$ $x_2(y_2)$... The x_i values correspond to the number of occurrences of the cross-correlation magnitudes y_i . Thus, for the R_8 inverse decimation pair 1[1] (corresponding to an autocorrelation), the weight distribution is 254 occurrences of -1 and 1 occurrence of 255, which is correct.

It can be seen by examination of Figure 4.1 that the R_8 's have singular code families; i.e., the code families for the R_8 's have only one inverse pair group each. On the other hand, for the R_7 's the set of ten possible inverse pair groups is reduced to only four code families. Thus, there are only four unique weight distributions which need to be searched for. The number of weight distribution functions was found in Niho [18] to be equal to the number of inverse decimation pairs. It is clear now that the number of weight distribution functions is equal to the number of code families. Table 4.1 shows the number of inverse pairs and code families for the R_3 's to the R_{15} 's as calculated using the FAMILIES software. The FAMILIES software was developed specifically for this purpose and will be described in Section 5.2.2. It is seen that there is not a large difference between the number of inverse pairs and

R7 Code Families

# 1:	1 [1]					
	126(-1)	1(127)				
# 2:	3 [43], 5 [27], 9 [15], 11 [13], 23 [29]					
	28(-17)	63(-1)	36(15)			
# 3:	7 [55], 19 [47], 21 [31]					
	1(-41)	14(-17)	28(-9)	35(-1)	28(7)	14(15)
	7(23)					
# 4:	63 [63]					
	7(-21)	7(-17)	8(-13)	21(-9)	7(-5)	14(-1)
	21(3)	7(7)	14(11)	14(15)	7(19)	

R8 Code Families

# 1:	1 [1]					
	254(-1)	1(255)				
# 2:	7 [37]					
	16(-33)	52(-17)	104(-1)	68(15)	14(31)	1(63)
# 3:	11 [29]					
	1(-65)	8(-33)	64(-17)	100(-1)	68(15)	10(31)
	4(47)					
# 4:	13 [59]					
	18(-33)	48(-17)	100(-1)	84(15)	4(47)	1(63)
# 5:	19 [47]					
	88(-17)	87(-1)	64(15)	8(31)	8(47)	
# 6:	23 [61]					
	88(-17)	89(-1)	56(15)	20(31)	2(63)	
# 7:	31 [91]					
	80(-17)	119(-1)	16(15)	40(31)		
# 8:	43 [43]					
	8(-33)	60(-17)	108(-1)	76(15)	1(63)	2(95)
# 9:	53 [53]					
	96(-17)	59(-1)	96(15)	4(63)		
#10:	127 [127]					
	8(-29)	16(-25)	8(-21)	18(-17)	24(-13)	16(-9)
	32(-5)	16(-1)	16(3)	20(7)	16(11)	16(15)
	16(19)	20(23)	8(27)	5(31)		

Figure 4.1: Code Families for the R7's and R8's

Table 4.1: The number of code families for the R3's to R15's

L	number of sequences	number of inv. prs	number of families
3	2	2	2
4	2	2	2
5	6	4	3
6	6	4	4
7	18	10	4
8	16	10	10
9	48	26	17
10	60	32	26
11	176	90	60
12	144	76	74
13	630	316	290
14	756	379	366
15	1890	904	879

the number of code families. However, as stated previously, it is important for the code search strategy that the ambiguous weight distribution functions be known.

The calculation of the code families was also performed using the FAMILIES program. Unfortunately, a theoretical proof of code families has not been realized at this time.

4.3 CROSS-CORRELATION ANALYSIS

The development of a technique to perform DS-SS code interception using cross-correlation analysis was the original intention of this thesis. However, some insurmountable difficulties were encountered which necessitated the determination of an interception technique based on weight distribution analysis. The technique based on cross-correlation analysis will now be presented and it will be shown that this technique is not feasible for code interception.

Assume that the full period of an m -sequence has been intercepted in a noiseless environment and that the sequence period is known. It is desired to determine which m -sequence has been intercepted. By correlating a locally generated reference m -sequence of the same period and the intercepted unknown m -sequence, it is possible to generate the full-period cross-correlation function between the two m -sequences. By comparing this function to a stored set of reference cross-correlation functions, the intercepted m -sequence can be determined after a few iterations. It will take several correlations due to the code families. The best way to describe the technique is to give an example which will be based upon the interception of an R8 m -sequence.

It will be assumed that the m -sequence generated by the characteristic

2

OF / DE

2

Micro-D

polynomial $g(x) = x^8 + x^4 + x^3 + x^2 + 1$, denoted S, has been intercepted. The interceptor does not have knowledge of this characteristic polynomial and wishes to determine it. The interceptor randomly selects an R8 to cross-correlate with the received sequence, in this case the m-sequence generated by the characteristic polynomial $g(x) = x^8 + x^6 + x^5 + x^3 + 1$, denoted T. The cross-correlation function is obtained and is compared to the list of all possible cross-correlation functions. The graphs of all possible unique cross-correlation functions for the R8's are given in Figures 4.2, 4.3 and 4.4. These graphs correspond to the weight distribution code families given in Figure 4.1. As well, the decimation relations graphs for the R8's is shown in Figure 4.5. As before, any pair of vertices on the same graph which are joined by a line segment of the same length, are related by the same inverse decimation pair; e.g., the sequences generated by the polynomials 435 and 747 (octal notation) are related by the inverse decimation pair 11[29] as are the polynomials 651 and 551.

If the intercepted sequence and the reference sequence are both in natural orientation, then the cross-correlation function will exactly match one of the ten possible graphs with a possible cyclic shift of the entire cross-correlation function. If either of the sequences is not in natural orientation, then the cross-correlation function between them will not be exactly the same as any of those in Figures 4.2, 4.3 and 4.4. However, the function will have the same weight distribution function; i.e., the magnitude and number of occurrences of cross-correlation values will be the same although the positions of the occurrences will not be coincident. For the sequences of this example, two of the many possible cross-correlation functions between S and T are displayed in Figure 4.5.

Upon comparison, it is seen that the cross-correlation function between the

Figure 4.2: Cross-Correlation Functions for the R8's (1)

Figure 4.3: Cross-Correlation Functions for the R8's (2)

Figure 4.4: Cross-Correlation Functions for the R8's (3)

Figure 4.6: Two Cross-Correlation Functions in the Same R8 Code Family

intercepted sequence and the reference sequence is given by the graph of Figure 4.2 which corresponds to code family #2. Since code family #2 corresponds to decimation by 7 or 37 (7 and 37 constitute an inverse decimation pair) of the reference sequence, then inspection of the decimation relations of Figure 4.5 reveals that the unknown sequence is either that generated by the polynomial 435 or 545 (octal). In other words, the possible characteristic polynomials for the intercepted sequence are $g_1(x) = x^8 + x^4 + x^3 + x^2 + 1$ or $g_2(x) = x^8 + x^6 + x^5 + x^2 + 1$. Unfortunately, an additional correlation is required to resolve the ambiguity. So, another reference sequence is selected and cross-correlated with the same intercepted sequence.

Choosing one of the possible solutions, the sequence generated by $g_2(x)$ is correlated with the unknown sequence. This time the cross-correlation function of Figure 4.3 which corresponds to code family #5 is matched to the output of the correlator circuit. Since code family #5 corresponds to decimation by 19 or 47, then inspection of the decimation relations of Figure 4.5 reveals that the unknown sequence is one of the two sequences generated by 435 or 607. Since in the first correlation, the result was that the possible solutions were 435 or 545, then it is clear that the correct solution is 435. Therefore, the intercepted sequence is generated by the m-sequence whose characteristic polynomial is given by $g_1(x) = x^8 + x^4 + x^3 + x^2 + 1$. This should be verified by correlating this solution sequence with the intercepted sequence. The expected output of an autocorrelation function corresponding to code family #1 indicates that the solution is correct.

For any noiseless m-sequences, it is true that the cross-correlation analysis technique will determine the intercepted m-sequence in only two trials (three with a verification trial). The difficulty of generating decimation relation graphs for

large R_n 's can be overcome by calculating the required generating polynomials as required. This can be performed using the techniques detailed in Simon et al [1, App. 5A.8] and could be implemented with minimal complexity. Another difficulty is the requirement that both the intercepted and the reference sequence be in natural orientation in order that all sequences related by the same inverse decimation pair have the same cross-correlation function. This will force the interceptor to employ a sophisticated pattern recognition system in order to recognize that certain cross-correlation functions have been generated.

However, the above difficulties can be overcome. Unfortunately, the presence of noise will introduce another difficulty which will cause this technique to fail. The presence of noise and its effect on the cross-correlation function will be analyzed as shown below. This derivation will preface the analyses which follow in the remainder of the chapter. As well, the failure of the cross-correlation analysis technique will be demonstrated through this analysis.

The cross-correlation of a noisy received sequence $x(t) = s(t) + n(t)$ and a noiseless reference sequence $y(t)$ will be determined, where $n(t)$ is a zero-mean Gaussian process. This will be used later to derive the form of the weight distribution of the cross-correlation function in Gaussian noise. The m -sequences are $s(t)$ and $y(t)$ are defined as periodic signals of period T consisting of $N = 2^L - 1$ $\{+1, -1\}$ -valued chips. The sequences have mean and variance which are given as follows.

Due to the balance property of m -sequences, both signals consist of $2^{L-1} - 1$ chips of amplitude $+1$ and 2^{L-1} chips of amplitude -1 . Thus, the mean of each sequence is given as:

$$E[s(t)] = E[y(t)] = \frac{-(2^{L-1}) + (2^{L-1} - 1)}{2^L - 1} \quad (4.1)$$

$$\begin{aligned} \text{So, } E[s(t)] = E[y(t)] &= -\frac{1}{N} \\ &\rightarrow 0 \quad \text{for } N \text{ large} \end{aligned} \quad (4.2)$$

The variances of the two m-sequences are equal and are found to be:

$$\begin{aligned} \sigma_s^2 = \sigma_y^2 &= \frac{\sum (X_i - E[X])^2}{N-1} \\ &= \frac{(2^{L-1} - 1)(1 + 1/N)^2 + 2^{L-1}(-1 + 1/L)^2}{N-1} \\ &= \frac{1}{N-1} \left[2^L \left(1 + \frac{1}{N^2} \right) - \left(1 + \frac{1}{N} \right)^2 \right] \\ &\rightarrow \frac{N}{N-1} \rightarrow 1 \quad \text{for } N \text{ large} \end{aligned} \quad (4.3)$$

Now, the magnitude of the cross-correlation function in noise is given by (2.1) as

$$\begin{aligned} r_{xy}(\epsilon) &= \int_0^T x(t)y(t+\epsilon) dt \\ &= \int_0^T [s(t) + n(t)] y(t+\epsilon) dt \\ &= \int_0^T s(t)y(t+\epsilon) dt + \int_0^T n(t)y(t+\epsilon) dt \end{aligned} \quad (4.4)$$

The left-hand integral can be recognized as (2.1), the ideal noiseless cross-correlation function over a full period with a relative phase shift of $\epsilon = \tau T_c$ between the two m-sequences, where T_c is the chip period of the sequences. The right-hand integral consists of noise correlated with the noiseless reference sequence and this term will superimpose a probability distribution function on the magnitudes of the noiseless cross-correlation function. The statistics of this distribution will now be determined.

The mean, or expected value $E[\cdot]$, of the cross-correlation function in additive white Gaussian noise (AWGN) at a specific phase shift ϵ is given as

$$\begin{aligned} E[r_{xy}(\epsilon)] &= E \left\{ \int_0^T s(t)y(t+\epsilon) dt + \int_0^T n(t)y(t+\epsilon) dt \right\} \\ &= \int_0^T E[s(t)y(t+\epsilon)] dt + \int_0^T E[n(t)y(t+\epsilon)] dt. \end{aligned} \quad (4.5)$$

But $n(t)$ and $y(t+\epsilon)$ are independent, implying

$$E[n(t)y(t+\epsilon)] = E[n(t)] \cdot E[y(t+\epsilon)] \quad (4.6)$$

where $E[n(t)] = 0$ is the mean of the Gaussian process. Therefore,

$$E[r_{xy}(\epsilon)] = r_{sy}(\epsilon) \quad (4.7)$$

which states that the mean of the cross-correlation function in noise at any phase shift is simply the noiseless cross-correlation magnitude at that phase shift.

The variance of the cross-correlation function in AWGN for a specific phase shift ϵ is given as

$$\sigma_{r_{xy}(\epsilon)}^2 = E[r_{xy}^2(\epsilon)] - E^2[r_{xy}(\epsilon)]. \quad (4.8)$$

The first term of (4.8) is derived to be:

$$\begin{aligned} E[r_{xy}^2(\epsilon)] &= E \left\{ \left[\int_0^T [s(t) + n(t)] y(t+\epsilon) dt \right]^2 \right\} \\ &= E \left\{ \int_0^T [s(t) + n(t)] y(t+\epsilon) dt \int_0^T [s(u) + n(u)] y(u+\epsilon) du \right\} \end{aligned} \quad (4.9)$$

where u has been introduced as a dummy variable in order to perform the integration. This decomposition will be performed many times during the derivations which follow. The inverse operation is also valid. Then, using equation (7.14) of Papoulis [11], it is obtained that

$$E[r_{xy}^2(\epsilon)] = E \left\{ \int_0^T \int_0^T [s(t) + n(t)] y(t+\epsilon) [s(u) + n(u)] y(u+\epsilon) du dt \right\}$$

$$\begin{aligned}
&= E \left\{ \int_0^T \int_0^T \left[s(t)y(t+\epsilon) + n(t)y(t+\epsilon) \right] \times \right. \\
&\quad \left. \left[s(u)y(u+\epsilon) + n(u)y(u+\epsilon) \right] du dt \right\} \\
&= E \left\{ \int_0^T \int_0^T s(t)s(u)y(t+\epsilon)y(u+\epsilon) du dt \right. \\
&\quad + \int_0^T \int_0^T s(t)n(u)y(t+\epsilon)y(u+\epsilon) du dt \\
&\quad + \int_0^T \int_0^T s(u)n(t)y(t+\epsilon)y(u+\epsilon) du dt \\
&\quad \left. + \int_0^T \int_0^T n(t)n(u)y(t+\epsilon)y(u+\epsilon) du dt \right\} \tag{4.10}
\end{aligned}$$

The expected value can be brought inside the integral signs giving

$$\begin{aligned}
&= \int_0^T \int_0^T E[s(t)s(u)y(t+\epsilon)y(u+\epsilon)] du dt \\
&\quad + \int_0^T \int_0^T E[s(t)n(u)y(t+\epsilon)y(u+\epsilon)] du dt \\
&\quad + \int_0^T \int_0^T E[s(u)n(t)y(t+\epsilon)y(u+\epsilon)] du dt \\
&\quad + \int_0^T \int_0^T E[n(t)n(u)y(t+\epsilon)y(u+\epsilon)] du dt \tag{4.11}
\end{aligned}$$

Now, since the noise is independent of the signals, then (4.6) implies that the second and third integrals of (4.11) equal zero. Thus, (4.11) becomes

$$\begin{aligned}
E[r_{xy}^2(\epsilon)] &= \int_0^T \int_0^T E[s(t)s(u)y(t+\epsilon)y(u+\epsilon)] du dt \\
&\quad + \int_0^T \int_0^T E[n(t)n(u)y(t+\epsilon)y(u+\epsilon)] du dt \Big\} \\
&= E^2[r_{xy}(\epsilon)] + \int_0^T \int_0^T E[n(t)n(u)y(t+\epsilon)y(u+\epsilon)] du dt
\end{aligned} \tag{4.12}$$

where the first integral has been solved using the inverse of (4.9). So, the variance of the cross-correlation function for a specific phase shift is:

$$\begin{aligned}
\sigma_{r_{xy}(\epsilon)}^2 &= E[r_{xy}^2(\epsilon)] - E^2[r_{xy}(\epsilon)] \\
&= \int_0^T \int_0^T E[n(t)n(u)y(t+\epsilon)y(u+\epsilon)] du dt \\
&= \int_0^T \int_0^T E[n(t)n(u)] \cdot E[y(t+\epsilon)y(u+\epsilon)] du dt
\end{aligned} \tag{4.13}$$

since the noise is independent of the sequences. But for AWGN, it is known that

$$E[n(t)n(u)] = \frac{N_o}{2} \delta(t-u) \tag{4.14}$$

where $\delta(x)$ is the Kronecker delta function defined as

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Therefore,

$$\begin{aligned}
\sigma_{r_{xy}(\epsilon)}^2 &= \int_0^T \int_0^T \frac{N_o}{2} \delta(t-u) E[y(t+\epsilon)y(u+\epsilon)] du dt \\
&= \frac{N_o}{2} \int_0^T E[y^2(t+\epsilon)] dt \\
&= \frac{N_o}{2} \int_0^T y^2(t+\epsilon) dt
\end{aligned} \tag{4.16}$$

since $y(t)$ is deterministic. Now the integral in (4.16) is the familiar autocorrelation

function for a phase shift of zero which has amplitude N , for an m -sequence of length N chips. Therefore,

$$\sigma_{r_{xy}(\epsilon)}^2 = N \frac{N_0}{2} \quad (4.17)$$

which is independent of the phase shift or the magnitude of the cross-correlation function. Note that the signal amplitude and power have been normalized to one, and so for purposes of SNR calculations, only the noise power will change. In terms of the SNR, the standard deviation is

$$\sigma_{r_{xy}(\epsilon)} = \sqrt{\frac{N}{2\text{SNR}_{\text{lin}}}} \quad (4.18)$$

where SNR_{lin} is the SNR in linear terms related to the SNR in decibels by

$$\text{SNR}_{\text{dB}} = 10 \log (\text{SNR}_{\text{lin}}) \quad (4.19)$$

It has been found that the cross-correlation function magnitude in noise at a specific phase shift has a mean equal to the noiseless cross-correlation magnitude and a variance equal to $N/2$ times $1/\text{SNR}$, where N is the length of the m -sequence. It is clear that the determination of m -sequences using cross-correlation analysis will fail. The magnitudes of the cross-correlation function in noise will vary such that any pattern recognition based on these functions will have a very low probability of detecting the correct cross-correlation pattern. Thus, the cross-correlation analysis is seen to be infeasible when noise is introduced into the problem. Hence, another technique is desired which would determine the intercepted m -sequence in a noisy environment. A technique which would solve the problem would have to be based on a statistical approach due to the noise effects, while simultaneously utilizing the characteristics of m -sequences to best advantage.

4.4 WEIGHT DISTRIBUTION ANALYSIS

The code interception technique which is developed in this thesis is based on an analysis of the weight distribution function of the cross-correlation function between two m -sequences. The intercepted m -sequence of known length N is cross-correlated with a reference m -sequence for a full period. Then, the probability distribution function of the cross-correlation function, or weight distribution, is calculated and compared to the small set of possible weight distributions to determine which sequence has been intercepted. In the noiseless case, this is a simple matter which is carried out in the same manner as for the cross-correlation analysis technique. The weight distribution in the noiseless case is simply a function consisting of a series of impulses of varying magnitudes. For example, the weight distribution of the cross-correlation functions of Figure 4.6 is shown in Figure 4.7. The vertical scale on this graph is normalized by the code length. The weight distribution calculations have been performed for all possible cross-correlations between m -sequences up to and including the R15's using the FAMILIES software.

In the case of the interception of an m -sequence in AWGN, the weight distribution function will be modified in a way which has been determined by the author. In order to determine which m -sequence has been intercepted, knowledge of the expected weight distribution function is required. The theoretical weight distribution in noise as determined by the author has been extensively simulated using the NOISEANA software developed during the course of this thesis. The NOISEANA software will be discussed in Section 5.2.3. The derivation of the weight distribution in AWGN will now be carried out. A description of the software

Figure 4.7: Noiseless Weight Distribution Function for Code Family #2 of R8's

and examples of the weight distribution in noise will be demonstrated in Chapter 5.

4.4.1 Calculation of Weight Distribution in Additive White Gaussian Noise

It is desired to derive the theoretical weight distribution of the cross-correlation function between two m-sequences, where one m-sequence is received with noise and the other is generated noise-free. This weight distribution is no longer a discrete function, as in the noiseless case, but is now a continuous function. The function which will be derived is designated $W_{xy}(\alpha)$, the continuous weight distribution function, and it is defined as

$$W_{xy}(\alpha) = \text{Prob} [r_{xy}(\epsilon) = \alpha]. \quad (4.20)$$

In other words, the value of the weight distribution at α , where α is some cross-correlation magnitude, is equal to the probability that the cross-correlation magnitude α has occurred.

Some of the simplifying notation which will be utilized to ease reading is as follows:

r_i replaces $[r_{xy}(\epsilon)]_i$,	the i^{th} distinct noiseless cross-correlation magnitude found using FAMILIES software;
n_i replaces $n[r_{xy}(\epsilon)]_i$	the number of occurrences of r_i ;
σ_r replaces $\sigma_{r_{xy}(\epsilon)}$	the standard deviation of the cross-correlation function.

As well, r_i , r_j , r_k will be used implying different magnitudes of the same cross-correlation function and n_i , n_j , n_k will be used implying numbers of occurrences of different magnitudes for the same cross-correlation function. These notations should make the derivation much more clear.

From the derivations of Section 4.3, it is clear that the weight distribution function will consist of the weighted summation of Gaussian distributions whose means are the r_i values and whose variances are σ_r^2 . It is well known that a Gaussian distribution with mean μ and variance σ^2 is given by

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-(x-\mu)^2/2\sigma^2]. \quad (4.21)$$

Thus, the weight distribution is described as

$$W_{xy}(\alpha) = \sum_{i \in S} \left[\left[\frac{n_i}{2^L - 1} \right] \frac{1}{\sigma_r \sqrt{2\pi}} \exp[-(\alpha - r_i)^2/2\sigma_r^2] \right] \quad (4.22)$$

where $S = \{\text{all distinct values of } r_{xy}(\epsilon)\}$.

Simplifying, one obtains

$$W_{xy}(\alpha) = \frac{1}{N\sigma_r\sqrt{2\pi}} \sum_{i \in S} \left[n_i \exp[-(\alpha - r_i)^2/2\sigma_r^2] \right] \quad (4.23)$$

This is the expression for the weight distribution in AWGN and it is easy to calculate since the values for n_i and r_i have been found using the FAMILIES software. It is clear that the weight distribution consists of a weighted sum of Gaussian functions, each of which will spread and lower as the SNR decreases.

It is easy to see that if there were no noise, i.e., if $\sigma_r = 0$, then (4.23) becomes the ideal noiseless weight distribution function

$$W_{xy}(\alpha) = \sum_{i \in T} \left[\frac{n_i}{N} \right] \quad (4.24)$$

where $T = \{r_{xy}(\epsilon): r_{xy}(\epsilon) \in S, r_{xy}(\epsilon) = \alpha\}$. An example has been given previously in Figure 4.7. An interesting result is that if the SNR gets very small (SNR $\ll 0$ dB), then the weight distribution function degenerates to a Gaussian distribution function,

$$W_{xy}(\alpha) = \text{Prob}[r_{xy}(\epsilon) = \alpha] \text{ as } \sigma_r \text{ becomes large}$$

$$\begin{aligned}
&= \lim_{\sigma_r^{-1} \rightarrow 0} \frac{1}{N\sigma_r\sqrt{2\pi}} \sum_{i \in S} \left[n_i \exp[-(\alpha - r_i)^2 / 2\sigma_r^2] \right] \\
&= \sum_{i \in S} \left[n_i \frac{1}{N\sigma_r\sqrt{2\pi}} \exp[-\alpha^2 / 2\sigma_r^2] \right]
\end{aligned} \tag{4.25}$$

where $\sum_{i \in S} n_i = N$ giving

$$W_{xy}(\alpha) = \frac{1}{\sigma_r\sqrt{2\pi}} \exp[-\alpha^2 / 2\sigma_r^2] \tag{4.26}$$

which is a zero-mean Gaussian distribution with a variance of σ_r^2 . This function will be denoted as $G(\alpha)$.

It is necessary that some measure for distinguishing between various weight distribution functions at the same SNR be developed. The application of neural networking, for example, could be utilized to perform some form of pattern recognition and differentiation. However, for the purposes of this thesis, a measure was desired which would distinguish between weight distributions and indicate whether a weight distribution function had degenerated to a Gaussian distribution. As well, it was desired to generate a measure whose results could be summarized concisely for ease of comparison between weight distributions.

From [22], two signals are easy to distinguish if and only if the mean-squared difference between them is large. So, the mean-squared difference and skewness functions between the weight distribution and an adaptive Gaussian function were derived. The adaptive Gaussian is the Gaussian distribution function of (4.26) which is termed adaptive because it will vary with input SNR. The mean-squared difference function calculates the mean-squared difference between the weight distribution and the Gaussian over all possible values of cross-correlation magnitude α and is effectively equivalent to a second moment

calculation between two functions. The skewness function measures the third moment between the weight distribution function and the Gaussian. This is necessary in order to compensate for the symmetry of the mean-squared difference (MSD) function being used. Used alone, the mean-squared difference function would give the same result for any weight distributions which were mirror images about the mean of the Gaussian function. The next two sections will derive these two functions which hereafter will be called the MSD and SKEW functions.

4.4.2 Derivation of the MSD Function Between a Weight Distribution and the Adaptive Gaussian

The mean-squared difference (MSD) function between the weight distribution and the reference Gaussian is

$$\begin{aligned} \text{MSD} &= \int_{-\infty}^{\infty} [W_{xy}(\alpha) - G(\alpha)]^2 d\alpha \\ &= \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \left[\frac{n_i}{N} \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right] - \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \right\}^2 d\alpha \end{aligned} \quad (4.27)$$

Extracting the common constant terms gives

$$= \frac{1}{2\pi\sigma_r^2} \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \left[\frac{n_i}{N} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right] - \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \right\}^2 d\alpha$$

Expanding terms,

$$\begin{aligned} &= \frac{1}{2\pi\sigma_r^2} \int_{-\infty}^{\infty} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] d\alpha \\ &\quad + \frac{1}{2\pi\sigma_r^2} \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \left[\frac{n_i}{N} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right] \right\}^2 d\alpha \end{aligned}$$

$$- \frac{1}{\pi \sigma_r^2} \int_{-\infty}^{\infty} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \sum_{i \in S} \frac{n_i}{N} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] d\alpha.$$

Solving the first integral, expanding the second integral and rearranging the third results in

$$\begin{aligned} \text{MSD} &= \frac{1}{2\sqrt{\pi}\sigma_r} \\ &+ \frac{1}{2\pi\sigma_r^2} \int_{-\infty}^{\infty} \sum_{i \in S} \sum_{j \in S} \frac{n_i n_j}{N} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \exp\left[-\frac{(\alpha - r_j)^2}{2\sigma_r^2}\right] d\alpha \\ &- \frac{1}{\pi\sigma_r^2} \int_{-\infty}^{\infty} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \sum_{i \in S} \frac{n_i}{N} \exp\left[\frac{-r_i^2 + 2\alpha r_i}{2\sigma_r^2}\right] d\alpha. \end{aligned}$$

Rearranging and simplifying gives

$$\begin{aligned} \text{MSD} &= \frac{1}{2\sqrt{\pi}\sigma_r} \\ &+ \frac{1}{2\pi N^2 \sigma_r^2} \int_{-\infty}^{\infty} \exp\left[-\frac{\alpha^2}{\sigma_r^2}\right] \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left[\frac{2\alpha(r_i + r_j) - (r_i^2 + r_j^2)}{2\sigma_r^2}\right] d\alpha \\ &- \frac{1}{\pi N \sigma_r^2} \int_{-\infty}^{\infty} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \sum_{i \in S} n_i \exp\left[\frac{-r_i^2 + 2\alpha r_i}{2\sigma_r^2}\right] d\alpha \quad (4.28) \end{aligned}$$

$$= A + B + C.$$

The terms of (4.28) will be treated separately and recombined later. The second term of (4.28) will now be solved. Removing the summation and constant terms of the second term gives

$$B = \frac{1}{2\pi N^2 \sigma_r^2} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left[-\frac{(r_i^2 + r_j^2)}{2\sigma_r^2}\right] \int_{-\infty}^{\infty} \exp\left[\frac{-\alpha^2 + \alpha(r_i + r_j)}{\sigma_r^2}\right] d\alpha$$

Solving the integral,

$$= \frac{1}{2\sqrt{\pi} N^2 \sigma_r} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left[\frac{-2(r_i^2 + r_j^2) + (r_i + r_j)^2}{4\sigma_r^2}\right]$$

$$= \frac{1}{2\sqrt{\pi}N^2\sigma_r} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left\{-\left[\frac{(r_i - r_j)}{2\sigma_r}\right]^2\right\} \quad (4.29)$$

which is the simplest form for the second term of (4.28). The third term can be treated similarly as follows.

$$C = -\frac{1}{\pi N \sigma_r^2} \sum_{i \in S} n_i \exp\left[-\frac{r_i^2}{2\sigma_r^2}\right] \int_{-\infty}^{\infty} \exp\left[\frac{-\alpha^2 + \alpha r_i}{\sigma_r^2}\right] d\alpha$$

Solving the integral and rearranging gives the result

$$C = -\frac{1}{\sqrt{\pi}N\sigma_r} \sum_{i \in S} n_i \exp\left[-\frac{r_i^2}{4\sigma_r^2}\right] \quad (4.30)$$

Recombining (4.29) and (4.30) into (4.28) results in

$$\begin{aligned} \text{MSD} &= \frac{1}{2\sqrt{\pi}\sigma_r} - \frac{1}{\sqrt{\pi}N\sigma_r} \sum_{i \in S} n_i \exp\left[-\frac{r_i^2}{4\sigma_r^2}\right] \\ &\quad + \frac{1}{2\sqrt{\pi}N^2\sigma_r} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left\{-\left[\frac{(r_i - r_j)}{2\sigma_r}\right]^2\right\} \\ &= \frac{1}{2\sqrt{\pi}\sigma_r} \left\{ 1 - \frac{2}{N} \sum_{i \in S} n_i \left[\exp\left[-\frac{r_i^2}{4\sigma_r^2}\right] - \frac{1}{2N} \sum_{j \in S} n_j \exp\left[-\left[\frac{(r_i - r_j)}{2\sigma_r}\right]^2\right] \right] \right\} \quad (4.31) \end{aligned}$$

This function has been implemented in the NOISEANA software and has been extensively simulated. The MSD is a function of the SNR, code length and the noiseless weight distribution between two m-sequences. The results of these simulations will be demonstrated in Chapter 5.

It can be shown that the MSD approaches zero for $\text{SNR} \ll 0$ dB (implying σ_r is large), implying that the received AWGN is dominant. Thus, a test for degradation of a weight distribution in noise is to check whether the MSD function is close to zero implying a Gaussian distribution.

As discussed previously, the MSD function cannot distinguish between

weight distributions which are mirror images of each other about zero. As well, there will be other weight distributions which give the same MSD value if their weightings are in certain orientations. Thus, it is necessary to generate the third moment, or skewness function. The skewness function will now be derived with simulation results demonstrated in Chapter 5.

4.4.3 Derivation of the SKEW Function Between a Weight Distribution and the Adaptive Gaussian

The SKEW function between a generalized weight distribution and the adaptive Gaussian is defined by the author as

$$\begin{aligned}
 \text{SKEW} &= \int_{-\infty}^{\infty} [W_{xy}(\alpha) - G(\alpha)]^3 d\alpha \quad (4.32) \\
 &= \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \left[\frac{n_i}{N} \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right] \right. \\
 &\quad \left. - \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] \right\}^3 d\alpha \\
 &= \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \frac{n_i}{N} \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right\}^3 d\alpha \\
 &\quad - 3 \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \frac{n_i}{N} \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right\}^2 \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{\alpha^2}{2\sigma_r^2}\right] d\alpha \\
 &\quad + 3 \int_{-\infty}^{\infty} \left\{ \sum_{i \in S} \frac{n_i}{N} \frac{1}{\sigma_r \sqrt{2\pi}} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \right\} \frac{1}{2\pi\sigma_r^2} \exp\left[-\frac{\alpha^2}{\sigma_r^2}\right] d\alpha \\
 &\quad - \int_{-\infty}^{\infty} \frac{1}{\sigma_r^3 (2\pi)^{3/2}} \exp\left[-\frac{3\alpha^2}{2\sigma_r^2}\right] d\alpha \quad (4.33)
 \end{aligned}$$

$$= A + B + C + D.$$

The four components of (4.33) will be treated separately and recombined later. The solution of integral D is a simple matter of integration resulting in the value:

$$\begin{aligned} D &= -\frac{1}{\sigma_r^3(2\pi)^{3/2}} \int_{-\infty}^{\infty} \exp\left[-\frac{3\alpha^2}{2\sigma_r^2}\right] d\alpha \\ &= -\frac{1}{2\pi\sigma_r^2\sqrt{3}}. \end{aligned} \quad (4.34)$$

Now, component C can be determined as

$$\begin{aligned} C &= \frac{3}{\sigma_r^3 N (2\pi)^{3/2}} \sum_{i \in S} n_i \int_{-\infty}^{\infty} \exp\left[-\frac{(\alpha - r_i)^2}{2\sigma_r^2}\right] \exp\left[-\frac{\alpha^2}{\sigma_r^2}\right] d\alpha \\ &= \frac{3}{\sigma_r^3 N (2\pi)^{3/2}} \sum_{i \in S} n_i \int_{-\infty}^{\infty} \exp\left[\frac{2\alpha r_i - 3\alpha^2 - r_i^2}{2\sigma_r^2}\right] d\alpha \\ &= \frac{3}{\sigma_r^3 N (2\pi)^{3/2}} \sum_{i \in S} n_i \exp\left[-\frac{r_i^2}{2\sigma_r^2}\right] \int_{-\infty}^{\infty} \exp\left[\frac{2\alpha r_i - 3\alpha^2}{2\sigma_r^2}\right] d\alpha. \end{aligned}$$

Solving the integral, the solution is obtained

$$C = \frac{\sqrt{3}}{2\pi N \sigma_r^2} \sum_{i \in S} n_i \exp\left[-\frac{2r_i^2}{3\sigma_r^2}\right] \quad (4.35)$$

Now, the solution to B is more complex but is found to be

$$B = \frac{-3}{\sigma_r^3 N^2 (2\pi)^{3/2}} \sum_{i \in S} \sum_{j \in S} n_i n_j \int_{-\infty}^{\infty} \exp\left[\frac{-\alpha^2 - (\alpha - r_i)^2 - (\alpha - r_j)^2}{2\sigma_r^2}\right] d\alpha$$

which can be manipulated considerably to obtain

$$B = \frac{-3}{\sigma_r^3 N^2 (2\pi)^{3/2}} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left[-\frac{(r_i^2 + r_j^2)}{2\sigma_r^2}\right] \int_{-\infty}^{\infty} \exp\left[\frac{-3\alpha^2 + 2\alpha(r_i + r_j)}{2\sigma_r^2}\right] d\alpha.$$

Integrating and simplifying, the result is

$$B = \frac{-\sqrt{3}}{2\pi N^2 \sigma_r^2} \sum_{i \in S} \sum_{j \in S} n_i n_j \exp\left[-\frac{(r_i - r_j)^2}{3\sigma_r^2}\right]. \quad (4.36)$$

The first term, A, is quite complex. The cubic is expanded to give

$$A = \frac{1}{\sigma_r^3 N^3 (2\pi)^{3/2}} \sum_{i \in S} \sum_{j \in S} \sum_{k \in S} n_i n_j n_k \int_{-\infty}^{\infty} \exp \left[\frac{-(\alpha - r_i)^2 - (\alpha - r_j)^2 - (\alpha - r_k)^2}{2\sigma_r^2} \right] d\alpha$$

$$= \frac{1}{\sigma_r^3 N^3 (2\pi)^{3/2}} \sum_{i \in S} \sum_{j \in S} \sum_{k \in S} n_i n_j n_k \int_{-\infty}^{\infty} f(\alpha) d\alpha$$

where

$$f(\alpha) = \exp \left[\frac{-3\alpha^2 + 2\alpha(r_i + r_j + r_k) - (r_i^2 + r_j^2 + r_k^2)}{2\sigma_r^2} \right].$$

Solving the integral and manipulating,

$$A = \frac{1}{2\pi\sigma_r^2 N^3 \sqrt{3}} \sum_{i \in S} \sum_{j \in S} \sum_{k \in S} n_i n_j n_k \exp \left[\frac{(r_i + r_j + r_k) - 3(r_i^2 + r_j^2 + r_k^2)}{6\sigma_r^2} \right] \quad (4.37)$$

Combining the results of (4.34), (4.35), (4.36) and (4.37) and manipulating, it is obtained that

$$\text{SKEW} = \frac{1}{2\pi\sigma_r^2 \sqrt{3}} \left\{ -1 + \frac{3}{N} \sum_{i \in S} n_i \left[\exp \left[\frac{-2r_i^2}{3\sigma_r^2} \right] - \frac{1}{N} \sum_{j \in S} n_j (\text{VAL}) \right] \right\} \quad (4.38)$$

where

$$\text{VAL} = \exp \left[\frac{-(r_i - r_j)^2}{3\sigma_r^2} \right] - \frac{1}{3N} \sum_{k \in S} n_k \exp \left[\frac{(r_i + r_j + r_k) - 3(r_i^2 + r_j^2 + r_k^2)}{6\sigma_r^2} \right] \quad (4.39)$$

This expression, while complex in appearance, is simple to implement in software. The SKEW function has been implemented in the NOISEANA software and extensive simulations were performed. Some results will be demonstrated in Chapter 5.

Several other extensive derivations have been carried out in the course of this thesis but these will be omitted since they follow the same line of reasoning as the MSD and SKEW derivations. These include the mean-squared difference between two different weight distributions and the mean-squared difference

between a weight distribution with SNR_1 and the same weight distribution with SNR_2 . The mean-squared difference between two different weight distributions could be useful in distinguishing between weight distributions and will give a result of zero if the correct weight distribution is obtained. Unfortunately, a result of zero will also occur if the SNR is very low and so this method of distinguishing weight distributions may not be useful. The mean-squared difference between a weight distribution and itself for two different SNR's is useful in dealing with errors due to incorrect determination of SNR via the chip rate detector. These results are quite complex and no simplifications were obtained in their derivations.

CHAPTER 5

RESULTS

5.1 INTRODUCTION

Substantial original results have been achieved in this thesis and these will now be presented. The software which was developed in support of the thesis will be introduced and discussed. Unfortunately, the sheer bulk of the listings preclude their inclusion in this thesis. The results of the calculations based on the weight distribution analysis of Chapter 4 will be presented and discussed. A presentation will be made of the various graphs generated in this thesis. These graphs include the cross-correlation functions for different m-sequences, ideal and noisy weight distribution functions and mean-squared difference and skewness functions. These results will be incorporated into a code search strategy based upon a combination of the analysis of weight distribution functions and decimation properties. This strategy is the basis of this thesis and its details will be explained through an example. Finally, the new code search strategy for code search will be compared with the brute force correlation techniques and its advantages and disadvantages will be highlighted.

5.2 CODE ANALYSIS SOFTWARE

Extensive software development was undertaken throughout this thesis. The software totalled roughly 6000 lines of code (not including comment lines) written in

both 80286 assembly language and the C language for the IBM PC AT. Much of this code has now been moved to 80386-processor based machines. The software was required in order to examine the interactions between m-sequences and the effects of these interactions on cross-correlation functions and weight distribution functions. Also, the weight distribution functions in noise were generated and the mean-squared difference and skewness functions between a weight distribution function and an adaptive Gaussian distribution were calculated. The various programs used in support of this thesis will now be discussed.

5.2.1 Program XCORR

XCORR was the initial software written for this thesis. Totalling about 2600 lines of code, XCORR was written using Turbo C V2.0 and Turbo Assembler for the 80286-based IBM PC AT with a 80287 Math Coprocessor, EGA graphics, an HP Laserjet printer and an HP 7470 plotter. In order to use the software, additional software was required. LINPLOT¹ converts data files into HP-GL (HP plotter control language) which allows plotting of screen graphics to a plotter. LASERPLOT² is a commercial software package which converts HP-GL files into HP Laserjet format allowing printing to an HP Laserjet printer. Thus, to plot high resolution graphs to a printer, it was necessary to generate data files which were converted to HP-GL with LINPLOT, then converted to Laserjet format with LASERPLOT, and then sent to the printer. This route was taken so that the

1 — kindly provided by Andrew Mudry of the Communications Electronic Warfare (CEW) Section at the Defence Research Establishment Ottawa, Ottawa, Ont., Canada

2 — © 1987, Insight Development Corporation

operations could be performed without requiring user interaction to obtain each plot. All of these routines are called from within XCORR and the plot generation is performed transparent to the user.

A functional hierarchy of the XCORR program can be seen in Figure 5.1 which demonstrates the interconnections between the various functions of the program. The hierarchy demonstrates the complexity of the software and gives an overview of the operations performed. Several of the functions are called by more than one function and these are denoted with an asterisk.

The XCORR program was written to generate cross-correlation plots for the m-sequences. The program is menu-driven and has the present capability of working with R8's only. The partial-period and full-period autocorrelation and cross-correlation functions can be generated for any portion of a code period. The correlation can simulate either digital or acousto-optic correlation processes. As well, the ideal weight distribution, cumulative weight distribution and the weight distribution in noise can be generated. Several of the correlation parameters can be varied and the results of these calculations are displayed on high-resolution plots to either the screen, a plotter or a laser printer. The menus available in the program will now be described. The Main menu allows selection of a new pair of sequences to work with, the capability to change the correlation parameters and access to the Display Plots menu.

The correlation parameters which can be varied are:

- 1) initial conditions of the sequence generators which are configured in the Galois architecture,
- 2) correlation time which can be varied to correlate over any range of partial-period or full-period correlations,

Figure 5.1: Functional Hierarchy of the XCORR program

3) relative phase shift between the sequences,

4) type of correlation either AO or digital.

The choices offered in the Display Plots menu are designed so that many plots can be analyzed with minimal page switching. The choices available in the Display Plots menu are as follows:

- 1) Auto Inc Phase Shift by One, allows viewing of the effects of stepping of the relative phase shift between two sequences;**
- 2) Change Relative Phase Shift, allows setting of the relative phase shift between two sequences;**
- 3) Auto Inc Integration Window by One, allows viewing of the evolution of the degradation of the correlation function as the correlation time is modified slightly;**
- 4) Change Integration Window, allows setting of the correlation time;**
- 5) Compressed Screen Plot, allows display of the entire correlation function on one screen in high-resolution colour graphics;**
- 6) Compressed Printer Plot, allows display of the entire correlation function on one page of hard copy.;**
- 7) Paged Screen Plot, spreads a correlation display over four screens to allow close examination of the function;**
- 8) Paged Printer Plot, spreads a correlation display over four pages of hard copy;**
- 9) Weight Distribution, generates the ideal weight distribution in tabular and graphical form and outputs to either the screen or to hard copy;**

- 10) Cumulative Weight Distribution, generates the ideal cumulative weight distribution and outputs to either the screen or to hard copy;
- 11) Weight Distribution in Noise, generates the weight distribution for different SNRs and displays to the screen. This function calculates the weight distribution by brute force (not decimation) and is quite slow;
- 12) Partial Period Statistics, generates the partial-period correlation magnitudes of a specific correlation start point for all relative phase shifts and finds the mean and variance;
- 13) Return to Main Menu;
- 14) Quit Program.

Note that the hard copy output from parts (6),(8) and (9) can be sent to either the plotter or HP Laserjet printer depending on the desired result. The user has the option of sending the output immediately or generating a plot file and printing later. The data file which is created by XCORR is called Dwwxxyzz.DAT where

ww = first sequence identifier

xx = second sequence identifier

zz = relative phase shift (in hex) for full-period plots

= correlation window (in hex) for partial-period plots

y = P for weight distribution plots

= N for compressed printer plots

= A,B,C,D for the pages of a paged printer plot

= E for partial-period weight distribution plots.

The .DAT file generated by XCORR is used by LINPLOT to generate a .PLT file which has the same name as the .DAT file. For plotter outputs, LINPLOT routes its output to the plotter instead of to a .PLT file. If a laser printer output is desired, then the .PLT file must be processed by the LASERPLOT software. LASERPLOT allows manipulation of magnification, line widths and many other parameters of a plot and outputs the plot to a laser printer. The option to print immediately or later allows one to display a plot immediately or to batch process many plots.

Some sample full-period correlation plots have been shown in Figures 4.2, 4.3, 4.4, and 4.6 and other correlation plots have the same general form. A sample weight distribution plot has been demonstrated in Figure 4.7. The paged printer plots will not be demonstrated.

The XCORR program initiated many of the further studies which were carried out during the thesis. The program made it possible to view many correlation plots and to recognize some of the patterns which exist with m-sequences. The link between decimations, cross-correlations and weight distributions was observed here and this led to further software development and the code search strategy.

5.2.2 Program FAMILIES

The next software to be developed was the FAMILIES software. FAMILIES consists of approximately 1400 lines of C and assembly code³. The functional hierarchy of FAMILIES is shown in Figure 5.2 demonstrating the relationship

3 — assembly code written by Lorne Gutz of the CEW Section at DREO.

Figure 5.2: Functional Hierarchy of the FAMILIES program

between the functions of the program. FAMILIES was developed to generate the code families for the R3's to the R15's and is strictly a number cruncher with no graphics or plot capability. This work is essentially an extension of Niho [18] with some added analysis to generate the code families instead of finding the inverse decimation pairs which have the same weight distribution function. A sample output from FAMILIES is shown in Figure 4.1. The program calculates all of the possible ideal weight distributions for a given R_n and sorts them into code families for output purposes. As well, data files are generated which allow the NOISEANA software to perform further processing to analyze the effect of noise on the weight distribution function. These data files are titled INVPRsxx.DAT where xx is the number corresponding to n for the R_n 's. The results of the FAMILIES software have been obtained for all m-sequences from the R3's to the R15's. The amount of processing time required on a 20 MHz OGIVAR 80386-based can be crudely approximated by presenting in terms of the ratio of time to find the families for the R_n 's to the time to find the families for the R_m 's, where m is less than n. The equation is

$$T_p(n) = 4(n-m) \frac{N_p(n)}{N_p(m)} \times T_p(m) \text{ secs} \quad (5.1)$$

where n is greater than m. Note that the $N_p(\cdot)$ terms are only approximate since the correct terms are the number of inverse decimation pairs. Thus, the R12's took 430 secs to run and the R15's took about 4.2 days to run, using the Ogivar computer. The operation of this program could be speeded up further using parallel processing or by implementing hardware to calculate the code families. Fortunately, the program need run only once for each R_n and the results are quite compact. This means a code search strategy based on these results could be implemented with existing technology. The memory requirements to store the code

families are well within present capabilities.

In order to perform pattern recognition on weight distributions in noise, it is necessary to be able to calculate these weight distributions based on the theory of Chapter 4. The calculation of the weight distributions in noise, the MSD and the SKEW functions have been implemented in software which is the topic of the next section.

5.2.3 Program NOISEANA

The NOISEANA software consists of about 1800 lines of C source code. This program was written to simulate the derivations of Chapter 4. The program provides high-resolution graphical displays to the screen or to a plotter/printer and the functional hierarchy is displayed in Figure 5.3. NOISEANA will calculate and display weight distributions in AWGN and find the MSD and SKEW functions. The program requires as input the data files INVPRSxx.DAT generated by the FAMILIES software and will generate outputs required to obtain the plots in the same manner as in the XCORR program. The program starts by reading in and listing all of the possible inverse decimation pairs whose weight distributions are available for further analysis. After selection of the inverse decimation pair to operate on, the program proceeds to the MAIN MENU whose choices will now be listed and described.

- 1) Pick New Decimation in same R_n , allows a new choice of inverse decimation whose weight distribution is to be operated on;
- 2) Weight Distribution — Screen, allows plotting of the weight distribution to the screen for a specific value of SNR which must be

Figure 5.3: Functional Hierarchy of the NOISEANA program

input at this time;

- 3) **Weight Distribution – Printer**, as (2) but to the plotter/printer. The plot can be obtained immediately or later governed by the same rules as in plotting in the XCORR software;
- 4) **MSD vs SNR – Screen**, calculates the MSD function for one weight distribution with the SNR ranging between 10 dB and -15 dB and displays the curve to the screen;
- 5) **MSD vs SNR – Printer/Plotter**, as (4) but outputs to both the screen and a printer/plotter as in XCORR;
- 6) **Skewness vs SNR – Screen**, calculates the SKEW function for one weight distribution with the SNR ranging between 10 dB and -15 dB and displays the curve to the screen;
- 7) **Skewness vs SNR – Printer/Plotter**, as (6) but outputs to both the screen and a printer/plotter as in XCORR;
- 8) **Cycle Through All MSD's**, generates the MSD function vs SNR for all possible inverse decimation pairs and displays the curves one at a time to the screen;
- 9) **Cycle Through All MSD's – Printer**, as (8) but displays the curves to the screen one at a time and generates data files for printing later;
- 10) **Cycle Through All Skewness**, generates the SKEW function vs SNR for all possible inverse decimation pairs and displays the curves one at a time to the screen;
- 11) **Cycle Through All Skewness – Printer**, as (10) but displays the curves to the screen one at a time and generates data files for printing later;
- 12) **Quit the Program.**

Some example plots of the weight distribution in noise and the MSD and SKEW functions vs SNR will given later in Section 5.3. The NOISEANA program provided a demonstration of many of the concepts derived in this thesis and was a very useful tool in establishing a theory based on the analysis of the weight distribution function in AWGN.

5.2.4 Miscellaneous Programs

Several other smaller programs were developed by the author to perform many of the tedious calculations required in this thesis. The Berlekamp algorithm of Section 3.6.2.1 has been implemented in software. The partial-period correlation values of a pair of sequences were calculated and printed out for user-defined sequences in program PCORR. Program AOCORR calculated acousto-optic correlation values for short sequences where the sequences were phase-shifted by $1/5^{\text{th}}$ of a chip at a time. This was used to examine the required detector resolution vs the chip width on a correlator.

Program FIND_DEC found all of the proper decimations for $GF(2)$ to $GF(2^{16})$ which was used in support of the FAMILIES software. Program GALFIELD generated the Galois fields from $GF(2)$ to $GF(2^{10})$ in polynomial notation given the minimal polynomial as input. An example of this was given in Table 3.1 which showed the Galois field $GF(2^4)$ where the minimal polynomial is $p(x) = 1 + x + x^4$. Program DECPOLY was used to find the generating polynomial of the sequence which is a proper decimation of the minimal polynomial. This is useful in finding the generating polynomials of a pair of m-sequences where an unknown sequence is generated by decimation of a known sequence.

It can be appreciated that software development was a significant component of this thesis. Unfortunately, due to the quantity of software generated by the author, it would be unwise to include the program listings in this thesis (upwards of 150 pages in total). This software is available to interested parties through the CEW section at DREO.

5.3 WEIGHT DISTRIBUTION ANALYSIS RESULTS

The results of the weight distribution analysis will now be presented. The weight distribution in noise was simulated for the R3's to R15's using the NOISEANA software. While extensive simulation for the R3's to the R15's was carried out, only a sampling of these will be presented here in order to demonstrate the principles learned. The sample outputs will be taken from a study of the R8's to demonstrate some of the results. The R8's have been selected because there is a small number of sequences which allows reasonable illustration of the principles. These principles can be extended to the other m-sequences with the only difference being that there are more code families to consider.

The distinct cross-correlation functions for the R8's have been displayed previously in Figures 4.2, 4.3 and 4.4 and the code families for the R8's have been given in Figure 4.1. The weight distribution functions in noise have been generated for the R8's using the NOISEANA software. An example of an ideal (noiseless) weight distribution function has been shown previously in Figure 4.7. All other noiseless weight distribution functions exhibit the same impulsive characteristic. In order to illustrate the degradation of the weight distribution functions as the SNR becomes more negative, plots were made of the weight distribution in noise for input

SNR values of 0.0 dB, -5.0 dB, -10 dB and -12.5 dB. These weight distributions correspond to each code family of the R8's and they are displayed in Figures 5.4 to 5.13. It is seen that the autocorrelation family (Family #1) has a weight distribution which is near Gaussian by the time the SNR is down to 0.0 dB. Thus, the weight distribution of the autocorrelation will not be useful in a code search strategy based on weight distribution analysis. While most of the weight distributions have degraded to Gaussians by the time that the input SNR has reached -10 dB, several of the distributions remain distinct down to SNR = -12.5 dB (R8 families #7, #8 and #9). Many of the weight distributions which have degraded still exhibit high levels of skewness at SNRs of -10 dB and may yield useful information. The effectiveness of a code search strategy based on these weight distributions will be a function of the degradation (i.e., SNR) of the distributions since there must be some way to distinguish between the weight distributions. It is important to note that as long as at least two distinct weight distributions exist, then the decimation properties of m-sequences will allow time savings to be made in a code search.

The function to distinguish between weight distributions which was developed in this thesis is based on the MSD and SKEW functions. These functions were derived in Sections 4.4.2 and 4.4.3 and have been simulated extensively using the NOISEANA software. Several plots of MSD and SKEW functions are demonstrated in Figures 5.14 to 5.16. These plots show the MSD and SKEW functions for all of the code families of the R7's, R8's and R9's to provide a sample of the results. The plots are functions of SNR and the specific code family to which a curve applies is labelled on the plot. Both functions converge to zero as the SNR decreases, which is an indication that a weight distribution has degraded to a

Figure 5.4: Code Family #1 Weight Distribution in Noise

Figure 5.5: Code Family #2 Weight Distributions in Noise

Figure 5.6: Code Family #3 Weight Distributions in Noise

Figure 5.7: Code Family #4 Weight Distributions in Noise

Figure 5.8: Code Family #5 Weight Distributions in Noise

Figure 5.9: Code Family #6 Weight Distributions in Noise

Figure 5.10: Code Family #7 Weight Distributions in Noise

Figure 5.11: Code Family #8 Weight Distributions in Noise

Figure 5.12: Code Family #9 Weight Distributions in Noise

Figure 5.13: Code Family #10 Weight Distributions in Noise

Figure 5.14: MSD and SKEW Functions for the R7's

Figure 5.15: MSD and SKEW Functions for the R8's

Figure 5.16: MSD and SKEW Functions for the R9's

Gaussian distribution function. However, for values of SNR greater than -12 dB, it is seen that these functions can provide some manner of distinguishing between weight distributions. Visual differentiation of weight distributions can still be performed without the MSD and SKEW functions, but this could prove to be very tiring. Note that the y-scales of the plots are very small and it is the relationship between the different curves of the same R_n which is important. Also, some of the curves on the same plot coincide or are very close to each other. However, there are typically distinct separations of some of the curves on these graphs. This will be useful in a code search strategy, since an error in the estimate of the input SNR (found using chip rate detector) could cause some curves to be indistinguishable. A SNR estimation error corresponds to moving up or down a curve and this translates into a MSD or SKEW magnitude error.

Nonetheless, it is seen from the curves of Figures 5.4 to 5.16 that a code search strategy based on weight distribution analysis will be useful. This will be demonstrated in the next section with an overview of the code search strategy and an example code search using weight distribution analysis.

5.4 CODE SEARCH STRATEGY BASED ON WEIGHT DISTRIBUTION ANALYSIS

The code search strategy based on weight distribution analysis is similar to that which is based on cross-correlation analysis. One of the differences is that for decreasing input SNRs, the weight distributions degrade in performance and become indistinguishable. For a code which is intercepted at a given input SNR, there will only be a certain number of distinct weight distributions with the remainder having

merged into the others. The technique will degrade to the autocorrelation methods when all of the weight distributions have taken on a Gaussian distribution. A "merge graph" can be constructed which displays the SNR values where a weight distribution has degraded into a Gaussian distribution or has become the same as another weight distribution. An example for the R8's has been developed and is displayed in Figure 5.17. The graph demonstrates when the weight distribution for a given code family (corresponding to Figure 4.1) cannot be distinguished from that of another. The values of SNR where a weight distribution has been sieved out of the code search strategy are approximate and are dependent upon the sieving criterion. The criterion which has been used in this thesis is that a weight distribution has merged with another when the MSD and SKEW function curves are both superimposed upon another. Also, a weight distribution is deemed to have merged if the MSD and SKEW functions have both gone to zero implying a Gaussian distribution. Other criteria to determine weight distribution merging which could be used include visual distinguishability or some pattern recognition system's criteria.

Thus, it is seen in Figure 5.17 that the code families 2 and 8 merge when the SNR is 0.2 dB and code families 2, 3, 4, 7, and 8 have merged when the SNR is -8.1 dB. All of the families except for the autocorrelation family have merged by the time that the SNR is down to -12.0 dB. The autocorrelation family merges with all of the rest when the SNR reaches -15.0 dB. It is seen that occasionally a merged code family will unmerge and become distinct for lower SNRs. For example, examination of the MSD graph of Figure 5.15 reveals that code families 9 and 10 are distinct down to SNRs of 7.1 dB, merge from 7.1 dB down to 4.3 dB and unmerge at 4.3 dB. These families then remerge at -12.0 dB. Thus, the code families 9 and 10

Figure 5.17: Weight Distribution Merge Graph for the R8's

have weight distributions which are distinct when the SNR is greater than 7.1 dB and between -12.0 dB and 4.3 dB. This is reflected on the merge graph of Figure 5.17.

Using the knowledge of the weight distribution functions, their decimation relationships and the merge graphs, it is now possible to formulate and conceptualize a code search strategy based on weight distribution analysis. The essence of the strategy is as follows. An m -sequence A is intercepted and recorded. After determining all parameters such as chip rate, code length, SNR and others, the unknown factor is the specific m -sequence which has been intercepted. The unknown m -sequence A is now correlated with one of the possible reference m -sequences B . The weight distribution of the cross-correlation function is then calculated by quantizing the cross-correlation function and binning the number of occurrences of each cross-correlation magnitude. The MSD and SKEW functions are then calculated between the achieved weight distribution and the adaptive Gaussian which has been described earlier in Sections 4.4.1, 4.4.2 and 4.4.3. At this point, the MSD and SKEW values are compared to the already calculated graphs of these functions. By examining the MSD and SKEW graphs and using the information provided by the merge graphs, it can be determined which weight distribution(s) correspond to the correlation of the intercepted sequence with the reference sequence. The set of possible solutions for the intercepted sequence is determined using the known relationship between the weight distributions and the decimation values. One of these is selected and the process is repeated. This process is repeated until there is only one possibility for the intercepted sequence, i.e., the intercepted m -sequence has been determined.

For SNR values higher than roughly -7 dB, the intercepted m -sequence will

typically be determined in two or three weight distribution calculations. As the SNR decreases further, more repetitions of the process will be required. For SNRs of -12 dB or lower, the number of repetitions will on average be equal to half of the number of possible codes, i.e., $N_p(L)/2$.

An example of the code search strategy will now be presented to illustrate the process. This example will be based upon the interception of an R8 and will be tried for SNRs of 5 dB, -5 dB and -10 dB. The intercepted sequence to be determined is the m -sequence generated by the characteristic polynomial $g(x) = x^8 + x^4 + x^3 + x^2 + 1$, represented by the octal notation 435 and denoted S. It is desired that the interceptor determine the m -sequence by utilizing the code search strategy based on weight distribution analysis. The decimation relation graph of Figure 4.5 will be required for the code search as will the weight distribution merge graph of Figure 5.17. The initial reference sequence will be the m -sequence which is generated by the characteristic polynomial $f(x) = x^8 + x^5 + x^3 + x + 1$, which is represented as 453 in octal notation, and denoted T.

For an SNR of 5 dB, the weight distribution for the cross-correlation function between S and T is found and the MSD and SKEW functions are calculated. The values are compared to the MSD and SKEW graphs of Figure 5.15 and found to correspond to code family #4. Examination of the merge graph of Figure 5.17 reveals that family #4 has not merged with any other family at an SNR of 5 dB. Examination of Figure 4.1 reveals that the decimations corresponding to code family #4 are 13 and 59. Therefore, it is clear by examination of Figure 4.5 that the intercepted sequence is either the m -sequence generated by 435 or 607, in octal notation. This is so because the sequence generated by 435 is the decimation by 59 of the m -sequence generated by 453 and 607 is the decimation by 13 of 453.

Using the m-sequence generated by 607 as the reference m-sequence in the second iteration, the weight distribution achieved is found to correspond to either code family #9 or #10. This means that the intercepted m-sequence is either generated by 435 (from family #9) or 703 (from family #10). Thus, by referring to the results of the first iteration, it is clear that the intercepted m-sequence is generated by the characteristic polynomial 435, in octal notation. This can be verified by cross-correlating with this new reference sequence. If an autocorrelation function results, then the determination of the intercepted m-sequence has been successful.

For the SNR value of -5 dB, the same initial reference m-sequence is used during the first iteration of the algorithm. Examination of the MSD and SKEW curves and of the merge graph reveals that the weight distribution corresponds to code families #3 and #4. Therefore, the possible solutions are 455, 717 (due to code family #3) and 435 or 607. Choosing 607 for the second iteration as before, the weight distribution of iteration 2 is found to correspond to code family #9 only. From this iteration, the only possible solution is 435. Once again, only two iterations were required to determine the intercepted m-sequence.

For an SNR of -10 dB, the initial m-sequence generated by 435 is used in the first iteration again. In this case, the possible code families are #2, #3, #4, #7 and #8, all of which have merged for the SNR = -10 dB. Therefore, the possible solutions are as follows:

- 543 and 765 due to code family #2,
- 455 and 717 due to code family #3,
- 435 and 607 due to code family #4,
- 537 and 615 due to code family #7, and

651 due to code family #8.

Choosing 607 for the second iteration again results in the weight distribution corresponding to code families #5, #6 and #9. Thus, the possible solutions from the second iteration are 545 and 651 due to code family #5, 455 and 765 due to code family #6 and 435 due to code family #9. Examination of the results of the first and second iterations reveals that the only sequences in common, and hence the possible solutions are the m-sequences generated by 651, 455, 765 and 435.

For the third iteration, one of the possible solutions is selected and utilized. With the m-sequence generated by 651, it is found that the weight distribution corresponds to code families #5, #6 and #9 again. However, since the reference sequence used in the third iteration differed from that used in the second iteration, then the list of possible solutions due to the third iteration will be different. For the third iteration, the list of possible solutions are 435 and 607 due to code family #5, 455 and 717 due to code family #6 and 545 due to code family #9. Therefore, the revised list of potential solutions is reduced to either 455 or 435. The fourth iteration will reveal the correct solution since either the autocorrelation will be achieved (right solution) or not. If not, then it is clear that 435 is the solution. So, for an SNR of -10 dB, the method required four iterations to solve for the generating polynomial of the intercepted m-sequence.

The code strategy demonstrated through these examples can be applied to m-sequences of any length for input SNRs above approximately -12 dB. The results will be achieved quickly with good performance maintained down to reasonably low SNRs.

5.5 A COMPARISON BETWEEN THE CODE SEARCH STRATEGY AND BRUTE FORCE AUTOCORRELATION TECHNIQUES

The key comparison between the code search strategy derived in this thesis and the brute force autocorrelation techniques is that of speed. The code search strategy based on the analysis of the weight distribution function has been demonstrated to work well for input signal-to-noise ratios as low as -12 dB, with some improvements available down to -15 dB. Based on requiring only two iterations to determine the intercepted m-sequence using the weight distribution analysis technique and based on a 3-out-of-5 detection scheme for the partial-period autocorrelation technique, the curves for code search time are shown in Figure 5.18. This curve amalgamates the data in Figure 3.9 with the new code search strategy data for the weight distribution analysis technique. Both of these assumptions are likely to be optimistic but reasonable for purposes of comparison. This graph shows the amount of time required to perform code search as a function of code length with a code generator chip rate of 10 MHz. The different lines correspond to the full-period autocorrelation technique of Section 3.6.4, the partial-period autocorrelation technique of Section 3.6.4, the weight distribution analysis technique of this thesis and the weight distribution analysis technique with an acousto-optic correlator. The AO correlator for the graph is assumed to have a parallelism of 1000 phases of a correlation function. The improvement due to the parallelism of the AO correlator could be applied to any of the curves. However, this was not done because it is used simply to illustrate the potential speed of the new method.

It is clear from Figure 5.18 that the new technique is orders of magnitude

Figure 5.18: Comparison of Code Search Times for Various Techniques
with $f_c = 10$ MHz

faster than the full-period autocorrelation technique for all code lengths, with the difference increasing for higher code lengths. The weight distribution analysis technique will achieve fast results on the same order of speed as the partial-period autocorrelation technique for codes with L less than 15. For L greater than or equal to 15, the new technique begins to be faster than the partial-period autocorrelation technique, with the factor growing with L . Keep in mind that the assumptions used to generate this graph are very optimistic, especially for the partial-period autocorrelation. Thus, the real difference in speed may be greater than that stated here. Nonetheless, the weight distribution analysis technique will be faster than the correlation techniques or at least as fast if the input SNR is greater than -15 dB.

CHAPTER 6

CONCLUSIONS

6.1 SUMMARY OF RESULTS

In this thesis, a technique has been developed which allows determination of an intercepted m-sequence for input SNR's down to about -12 dB. The technique is based on the analysis of the weight distribution of cross-correlation functions between two m-sequences. The technique has been shown to be a significant improvement over existing code search techniques. Decoding techniques are very fast but fail if the input SNR is less than about 5 dB. Correlation techniques work down to input SNR's of -43 dB with conditions defined earlier, but they are very slow. The weight distribution analysis technique addresses the shortcomings of both methods, providing determination of the m-sequence in less time than correlation methods and working down to input SNR's as low as -12 dB. It has also been shown that a code search technique based on the analysis of cross-correlation functions will encounter difficulties. These techniques would take as long as the correlation techniques and would not work to input SNR's as low as the correlation techniques, so the cross-correlation analysis methods have little merit. They also fail due to the problem of achieving natural orientation for both the reference and intercepted sequence.

It has been shown that the weight distribution analysis technique will work faster than correlation techniques until the SNR is lower than about -12 dB. At this point, the technique will have degraded to the speed of the correlation

techniques. A code search strategy based on the weight distribution analysis technique has been formulated and an example given.

Several misconceptions regarding spread spectrum modulation and coding have been corrected. The misconception that m-sequences are easy to intercept and determine has been shattered, since previous research hinged on the assumption that the m-sequence is received with no noise.

The weight distribution in AWGN has been derived theoretically and simulated on a computer. The mean-squared difference and skewness functions have been derived and simulated between a weight distribution in noise and an adaptive Gaussian probability distribution function. This Gaussian function has been derived as a function of SNR and is used as the ideal to compare all weight distributions to. This function was used because all of the weight distribution functions degrade to this adaptive Gaussian as the SNR decreases. Most weight distributions were found to have degraded to Gaussians by the time the SNR was down to -12 dB.

The mean-squared difference and skewness functions were utilized to provide a measure which differentiated between the weight distributions. The application of artificial intelligence techniques to the weight distribution pattern recognition problem could enhance the overall weight distribution analysis technique considerably.

In the course of this thesis, significant software development has been performed, with approximately 6000 lines of C code being generated to perform many of the required calculations. XCORR was written to generate and display cross-correlation functions for the R8's for either full or partial periods and with the potential to vary a great many of the parameters and determine their effect on

the cross-correlation function. FAMILIES was written to find all of the weight distribution families for the R3's to R15's with the capability to extend further if necessary. It is noted that FAMILIES took over 5 days to calculate all of the weight distribution families for the R15's using a 20 MHz Ogivar 80386-based computer. NOISEANA was written to simulate the weight distribution function between two m-sequences when one of the sequences is received with AWGN. The mean-squared difference and skewness functions were also calculated and displayed as measured between a weight distribution function and the adaptive Gaussian. A great number of cross-correlation functions, weight distribution functions and weight distributions in noise have been generated and plotted, as well as the mean-squared difference and skewness functions.

As well, the weight distribution functions for all possible pairings of m-sequences were calculated for the R3's to R15's, a considerable extension to the work of Niho [18]. The proper decimations were calculated and stored for the R3's to R25's and their inverse pairings were determined. Several fundamental properties of m-sequences were examined and many corrections to existing texts in the field of spread-spectrum have been made. The concepts of reciprocal decimation and code families were introduced and exploited in this thesis.

Finally, the decimation, cross-correlation and weight distribution properties of m-sequences were exploited jointly, something which has not been done in previous work. In previous work, one or two of these three properties were combined but all three have not been combined. This combination of properties enabled considerable insight to be made into the m-sequences and allowed the formulation of this code interception technique.

6.2 FUTURE WORK

There is a vast amount of work which still needs to be done in this area. A faster than brute force code intercept technique is still required for SNR's less than -12 dB. The weight distribution analysis technique requires full-period correlations to be performed. A technique based on partial-period correlation would provide speed improvements. An analysis of partial-period cross-correlations is also necessary. A code intercept technique could be developed which works on codes other than m-sequences, such as non-linear codes. The effects of carrier and chip rate deviations, interleaving, encryption, different chip waveforms and modulation should be examined. Interception techniques for FH-SS and hybrid SS should be developed. Also, the effects of non-Gaussian channel and optical noise could be examined.

Most importantly, this technique could be implemented and the work extended for higher R_n 's. The software could be enhanced to include other codes, truncation of the sequence length (to make the ratio of the chip rate to data rate a nice number) and interleaving. The decimation properties and code families could be incorporated into the correlation techniques to give vast improvements to those methods. Artificial intelligence could be applied to the pattern recognition problem associated with recognizing and matching the weight distribution functions. Finally, a measure of distinguishability between weight distributions could be developed to improve on the mean-squared difference and skewness functions which have been proposed.

REFERENCES

- [1] Simon, M.K., et al., *Spread Spectrum Communications*, Vols. 1-3, Computer Science Press, Rockville, MD., 1985.
- [2] R.C. Dixon, *Spread Spectrum Systems*, 2nd edition, John Wiley & Sons, New York, 1984.
- [3] R.E. Ziemer and R.L. Peterson, *Digital Communications and Spread Spectrum Systems*, Macmillan Publishing Co., New York, 1985.
- [4] D.L. Nicholson, *Spread Spectrum Signal Design: LPE and AJ Systems*, Computer Science Press, Rockville, MD, 1988.
- [5] J.K. Holmes, *Coherent Spread Spectrum Systems*, John Wiley & Sons, New York, 1982.
- [6] R.L. Pickholtz et al, *Theory of Spread-Spectrum Communications - A Tutorial*, IEEE Trans. on Communications, Vol. COM-30, No. 5, pp 855-883, May 1982.
- [7] S. Hatcher, *Starfix: Satellite Navigation for Maritime Users*, Microwave Systems News, Vol. 19, No. 7, pp 35-40, Jul 1989.
- [8] C.E. Jagger, *The Vulnerability of Spread Spectrum Signals to Interception*, Canadian General Electric Co., Ltd., Toronto, Ont., 12 Feb 1973.
- [9] T.J. Hall, *Intercept Vulnerability of Direct Sequence Pseudo-Noise Encoded Spread Spectrum Waveforms*, MSc Thesis, Air Force Institute of Technology, 15 Dec 1979.
- [10] J.G. Proakis, *Digital Communications*, McGraw-Hill Book Co., New York, 1983.
- [11] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd edition, McGraw-Hill Book Co., New York, 1984.
- [12] A.G. Konheim, *Cryptography: A Primer*, John Wiley & Sons, New York, 1981.
- [13] N.J. Berg and J.N. Lee, *Acousto-Optic Signal Processing*, Marcel Dekker, Inc., New York, 1983.
- [14] S.W. Golomb, *Shift Register Sequences*, Holden-Day, Inc., San Francisco, CA, 1967.
- [15] R. Lidl and H. Niederreiter, *Encyclopedia of Mathematics, Vol. 20 - Finite Fields*, Addison-Wesley Publishing Co., Reading, Mass., 1983.

- [16] S. Lin and D.J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1983.
- [17] W.W. Peterson and E.J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed, MIT Press, Cambridge, Mass., 1970.
- [18] Y. Niho, *Multi-Valued Cross-Correlation Functions Between Two Maximal Linear Recursive Sequences*, PhD Dissertation, Department of Electrical Engineering, University of Southern California, Los Angeles, CA, Jan 1972.
- [19] N.E. Bekir, *Bounds on the Distribution of Partial Correlation for PN and Gold Sequences*, PhD Dissertation, Department of Electrical Engineering, University of Southern California, Los Angeles, CA, Jan 1978.
- [20] H.M. Trachtenberg, *On the Cross-Correlation Functions of Maximal Linear Recurring Sequences*, PhD Dissertation, Department of Electrical Engineering, University of Southern California, Los Angeles, CA, Jul 1970.
- [21] H.F.A. Roefs, *Binary Sequences for Spread-Spectrum Multiple-Access Communication*, PhD Dissertation, Department of Electrical Engineering, University of Illinois at Urbana-Champaign, 1977.
- [22] D.V. Sarwate and M.B. Pursley, *Crosscorrelation Properties of Pseudorandom and Related Sequences*, Proc. IEEE, Vol. 68, No. 5, pp 593-619, May 1980
- [23] R. Gold, *Characteristic Linear Sequences and Their Coset Functions*, SIAM Journal of Applied Math, Vol. 14, No. 5, pp 980-985, Sep 1966.
- [24] N. Zierler, *Linear Recurring Sequences*, Massachusetts Institute of Technology, Lincoln Laboratory, Group Report 34-63, 21 Oct 1958.
- [25] M.B. Pursley and D.V. Sarwate, *Performance Evaluation for Phase-Coded Spread-Spectrum Multiple-Access Communication - Part II: Code Sequence Analysis*, IEEE Transactions on Communications, Vol. COM-25, No. 8, pp 800-803, Aug 1977.
- [26] S. Faulkner and N. Brousseau, *Study of the Cross-Correlation Properties of R8 for the Testing of a Time-Integrating Correlator*, Technical Note in progress, Defence Research Establishment Ottawa, Ottawa, Ont., Sep 1989.
- [27] F.J. MacWilliams and N.J.A. Sloane, *Pseudo-Random Sequences and Arrays*, Proc. IEEE, Vol. 64, No. 12, pp 1715-1729, Dec 1976.
- [28] P.F. Duvall and J.C. Mortick, *Decimation of Periodic Sequences*, SIAM Journal of Applied Math, Vol. 21, No. 3, pp 367-372, Nov 1971.
- [29] R. Gold, *Optimal Binary Sequences for Spread Spectrum Multiplexing*, IEEE Transactions on Information Theory, Vol. IT-13, No. 4, pp 619-621, Oct 1967.

- [30] R. Gold, *Maximal Recursive Sequences with 3-Valued Recursive Cross-Correlation Functions*, IEEE Transactions on Information Theory, Vol. IT-14, No. 1, pp 154-156, Jan 1967.
- [31] J.J. Stiffler, *Rapid Acquisition Sequences*, IEEE Transactions on Information Theory, Vol. IT-14, No. 2, pp 221-225, Mar 1968.
- [32] J.D. Olsen, *Nonlinear Binary Sequences with Asymptotically Optimum Periodic Cross-Correlation*, PhD Dissertation, Department of Electrical Engineering, University of Southern California, Los Angeles, CA, Dec 1977.
- [33] E.J. Groth, *Generation of Binary Sequences with Controllable Complexity*, IEEE Transactions on Information Theory, Vol. IT-17, No. 3, pp 288-296, May 1971.
- [34] E.L. Key, *An Analysis on the Structure and Complexity of Nonlinear Binary Sequence Generators*, IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp 732-736, Nov 1976.
- [35] D.J. Torrieri, *Principles of Secure Communications Systems*, Artech House, Dedham, MA, 1985.
- [36] J.L. Massey, *Shift Register Synthesis and BCH Decoding*, IEEE Transactions on Information Theory, Vol. IT-15, No. 1, pp 122-127, Jan 1969.
- [37] H. Anton, *Elementary Linear Algebra*, 3rd edition, John Wiley & Sons, New York, 1981.
- [38] T. Helleseth, *Some Results About the Cross-correlation Function Between Two Maximal Linear Sequences*, Discrete Math, Vol. 16, pp 209-232, 1976.
- [39] V.M. Sidelnikov, *On Mutual Correlation of Sequences*, Soviet Math. Dokl., Vol. 12, pp 197-201, 1971.
- [40] J.H. Lindholm, *An Analysis of the Pseudo-Randomness Properties of Subsequences of Long m-Sequences*, IEEE Transactions on Information Theory, Vol. IT-14, No. 4, pp 569-576, Jul 1968.
- [41] F.D. Schmandt, R.G. McLaughlin, J.K. Wolf, *Crosscorrelation of a Maximal Length Binary Sequence and its Subsequences*, Research and Technology Division, Rome Air Development Center, Air Force Systems Command, Griffis AFB, N.Y., Tech. Rpt. RADC-TR-67-261, Aug 1967.
- [42] B. Gordon, W.H. Mills, L.R. Welch, *Some New Difference Sets*, Canadian Journal of Mathematics, Vol. 14, pp 614-625, 1962.
- [43] private communications, M.P. Ristenbatt

END

1 3 0 9 9 0

FIN