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.

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 0-315-53249-1

Canada

**PREDETECTION FILTER DESIGN FOR PRACTICAL
DIGITAL COMMUNICATION SYSTEMS**

by

Wayne C. Groff, B.Sc.

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

April, 1989

Wayne C. Groff, Ottawa, Canada, 1989.

ABSTRACT

In the past, the design of predetection filters for linear receivers has been carried out using nominal signal waveforms to approximate the actual received waveforms, possibly with imposed constraints to make the problem of finding the optimal filter tractable. Studies involving interference in such systems have often focused on calculating the performance degradation that interference and other deviations from the nominal signal cause for such a filter. In this thesis, it is argued that if an interference source is sufficiently well-understood it should be included in the process of optimizing the predetection filter. The recently developed AOLFP algorithm provides the means for determining the optimal filter for many communication signals and sources of interference, with no significant constraints.

When applied to cases involving interference, however, the memory space and computing time required during processing can quickly be extreme. An extension to the algorithm is developed that overcomes this problem, and the extended algorithm is illustrated by application to three cases of a duobinary MSK signal suffering from some form of interference.

First studied is the effect of phase errors in carrier recovery, in which the optimal filter is seen to provide little performance improvement over the nominally optimal filter. It is concluded that the degradation in performance is a result of the structure of the linear receiver. In addition, the nominal filter is shown to be robust with respect to carrier recovery phase errors. Second, the optimal filter for duobinary MSK in the presence of adjacent channel interference is determined. Asymptotic performance gains ranging from 1 dB to 4 dB are achieved, depending on the level of distortion. Finally, the duobinary MSK signal is bandlimited at the transmitter, introducing intersymbol interference. Here, the optimal filter is seen to provide asymptotic gains on the order of 1 dB over the nominal filter.

ACKNOWLEDGEMENTS

Sincere thanks are extended to Dr. P. Galko for his guidance and many valuable suggestions over the course of this work. Thanks are also extended to the Natural Sciences and Engineering Research Council of Canada for their financial support.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	v
LIST OF TABLES	vii
1. INTRODUCTION	
1.1 A New Approach to Filter Design	1
1.2 Thesis Organization	5
2. CALCULATING THE AOF - THE AOLFP ALGORITHM	
2.1 Preliminaries	6
2.2 The Origin of the AOF	8
2.3 Properties of the AOF Vector	11
2.4 Development of the AOLFP Algorithm	13
2.4.1 Generating a Feasible Receiver Vector	14
2.4.2 Testing Feasible Vectors for Optimality	18
2.4.3 Algorithm Statement	20
2.5 Implications of the AOLFP Algorithm	22
3. PRACTICAL CONSIDERATIONS IN USING THE AOLFP ALGORITHM	
3.1 Implementing the AOLFP Algorithm	23
3.1.1 Memory Space Considerations	23
3.1.2 Solving the Quadratic Programming Problem	25
3.2 Modifying the AOLFP Algorithm for Large Signal Sets	26
3.2.1 Development of the Extended AOLFP Algorithm	26
3.2.2 Some Typical Examples	31
4. APPLYING THE AOLFP ALGORITHM TO DUOBINARY MSK	
4.1 Duobinary MSK and MSK-Type Detection	37
4.2 Nominal Filter Design	41
5. PRACTICAL APPLICATIONS OF THE AOLFP ALGORITHM	
5.1 A General Approach to Calculating the Inner Product Matrix	53

5.2 Filter Design for Nonideal Carrier Recovery	56
5.2.1 Signal Model and Calculation of the Inner Products	56
5.2.2 Results	63
5.3 Filter Design for Adjacent Channel Interference	70
5.3.1 Signal Model and Calculation of the Inner Products	71
5.3.2 Results	79
5.4 Filter Design for a Bandlimited Signal	89
5.4.1 Signal Model and Calculation of the Inner Products	89
5.4.2 Results	95
6. CONCLUSIONS	
6.1 Summary	112
6.2 Suggestions for Further Work	114
APPENDIX A COMPLETE SOFTWARE EXAMPLE USING THE AOLFP ALGORITHM	115
REFERENCES	136

LIST OF FIGURES

Figure	page
2.1 Linear receiver structure.	6
2.2 Linear receiver structure using a correlator.	8
2.3 Possible trajectories for: (a) a nonfeasible vector, and (b) a feasible vector.	17
4.1 IQ receiver.	39
4.2 Phase tree for duobinary MSK.	43
4.3 Nominal eye diagram for duobinary MSK.	44
4.4 Superimposed data signatures for nominal duobinary MSK.	45
4.5 Basic pulses comprising the nominal data signatures.	46
4.6 Data signatures for nominal duobinary MSK.	49
4.7 Correlating waveforms of the nominal filters for $L = 3T$, $L = 4T$, and $L = 5T$.	50
4.8 BER of the nominal filters for $L = 3T$, $L = 4T$, and $L = 5T$.	51
5.1 Distortion in the nominal data signatures due to a phase error of 10° .	58
5.2 Distortion in the nominal data signatures due to a phase error of 20° .	59
5.3 Distortion in the nominal data signatures due to a phase error of 45° .	60
5.4 Correlating waveforms of the AOF's and nominal filter for $L = 3T$ and several values of θ_{max} .	65
5.5 BER of the AOF's and nominal filter for $L = 3T$ and several values of θ_{max} .	66
5.6 BER of the AOF for $\theta_{max} = 20^\circ$ and the nominal filter for fixed phase errors.	68
5.7 Eye diagrams of the data signature components for $\tau = 0.5T$.	73
5.8 Time varying portion of $r_i(t)$.	77
5.9 Time varying portion of $r_i(t)$ on a time shifted observation interval.	78
5.10 Correlating waveforms of the AOF's for $L = 3T$ and ACI.	80
5.11 BER of the AOF's and nominal filter for $L = 3T$ and ACI.	82
5.12 Correlating waveforms of the AOF's for $L = 4T$ and ACI.	84

Figure	page
5.13 Correlating waveforms of the AOF's for $L = 5T$ and ACI.	85
5.14 BER of the AOF's and nominal filter for $L = 4T$ and ACI.	86
5.15 BER of the AOF's and nominal filter for $L = 5T$ and ACI.	87
5.16 Equivalent views of the transmitter for a bandlimited signal: (a) with a bandpass filter, and (b) with lowpass filters.	90
5.17 Third order Butterworth phase characteristic.	94
5.18 Eye diagram of bandlimited duobinary MSK for $f_c = \frac{.375}{T}$.	96
5.19 Responses to the basic pulses for $f_c = \frac{.375}{T}$.	97
5.20 Responses to the basic pulses for $f_c = \frac{.25}{T}$.	99
5.21 Eye diagram of bandlimited duobinary MSK for $f_c = \frac{.25}{T}$.	100
5.22 Correlating waveforms of the AOF's for $L = 3T$ and bandlimiting.	101
5.23 Correlating waveforms of the AOF's for $L = 4T$ and bandlimiting.	102
5.24 Correlating waveforms of the AOF's for $L = 5T$ and bandlimiting.	103
5.25 BER of the AOF's and nominal filter for $L = 3T$ and bandlimiting.	106
5.26 BER of the AOF's and nominal filter for $L = 4T$ and bandlimiting.	107
5.27 BER of the AOF's and nominal filter for $L = 5T$ and bandlimiting.	108

LIST OF TABLES

Table	page
3.1 Parameters of intermediate signal sets for $\epsilon = .15$ and $n_{max} = 1000$.	32
3.2 Parameters of intermediate signal sets for $\epsilon = .15$ and $n_{max} = 500$.	33
3.3 Parameters of intermediate signal sets for $\epsilon = 1.00$ and $n_{max} = 500$.	34
3.4 Parameters of intermediate signal sets for $\epsilon = 1.05$ and $n_{max} = 1000$.	35
3.5 Parameters of intermediate signal sets for $\epsilon = .95$ and $n_{max} = 1000$.	35
5.1 Asymptotic degradation of the AOF and nominal filter from the BER of MSK for several values of θ_{max} .	63
5.2 Asymptotic degradation of the AOF for $\theta_{max} = 20^\circ$ and the nominal filter from the BER of MSK for fixed phase errors.	69
5.3 Asymptotic degradation of the AOF and nominal filters for bandlimited duobinary MSK from the BER of MSK.	109

CHAPTER 1

INTRODUCTION

1.1 A New Approach to Filter Design

In many digital communication systems, signal detection is accomplished by shifting the received signal to baseband, applying the baseband signal to a linear time-invariant predetection filter, and sampling the filter output at the baud rate. Decisions for successively transmitted data symbols are then based on each successive sample value. Often, the input signal to the filter excluding noise is such that many different waveforms are possible over single baud intervals for the same transmitted data symbol. This situation results when the received signal on one baud interval is a function of several transmitted symbols, in other words, when the signal contains “memory”. In some cases this memory is intrinsic to the modulation process, such as in generalized minimum shift keying (MSK) [1] and partial response continuous phase modulation (CPM) [2], where smoother phase trajectories are achieved through overlapping frequency pulses. Even for signals without such intrinsic memory in the modulation process, effects such as phase errors in carrier recovery, intersymbol interference (ISI), adjacent channel interference (ACI), and variations in channel characteristics such as fading will produce the same behaviour of several possible received signals for a single data symbol in the signal appearing at the input to the predetection filter.

Since the problem of predetection filter design is usually carried out analytically, including this type of behaviour in the process of optimizing the impulse response can be extremely difficult, especially when the behaviour results from or is augmented by some form of interference. In such cases, the typical approach has been to simplify the problem by assuming that the nominal signal waveforms are received in the presence of additive white Gaussian noise (AWGN) and impose additional constraints that, although resulting

in a suboptimal filter, make the problem tractable. For example, filter design for pulse amplitude modulation (PAM) systems typically involve the constraint that there be no ISI at the sampling point, or alternatively, that the mean squared error (MSE) be minimized [3,Ch.8]. Results from this approach for PAM have also been applied to predetection filter design for various partial response CPM signals by approximating these signals as PAM signals [1] [4]. Although the solution is suboptimal, it is desirable as it allows nonlinearly modulated signals such as TFM and duobinary MSK to be detected with simple linear receivers.

An alternate approach to the design of the predetection filter has been suggested [5] involving simple Butterworth and Chebyshev impulse responses. This approach was investigated for duobinary MSK and TFM and while good results were obtained for these two signals, the approach is clearly suboptimal and may not provide good results in general.

Recently a new approach [6] to the design of the predetection filter has been developed that allows the above behaviour to be considered for binary communication signals. The approach considers the problem of finding the impulse response that maximizes the absolute minimum opening of the noiseless post-filter decision eye. This problem has been termed the "*Asymptotically Optimal Linear Filtering Problem*" (AOLFP) and the solution is a filter referred to as the "*Asymptotically Optimal Filter*" (AOF). The discussion in [6] demonstrates that the optimal filter response converges to the AOF impulse response for a signal corrupted by AWGN in the limit of infinitely large signal to noise ratio (SNR). That is, the AOF minimizes the probability of a decision error in the asymptotic limit of infinite SNR. The discussion also develops an efficient algorithm, involving no significant constraints, for determining the AOF and demonstrates that the AOF can be expected to be near optimal at practical SNR's.

The algorithm was originally developed to compute the impulse responses for predetection filters for the generalized MSK signals which include among others: MSK, duobinary MSK and TFM. It is based on a geometric interpretation of the various waveforms that

can possibly occur at the input to the predetection filter. These waveforms are viewed as unique vectors in signal space and the algorithm operates on a matrix of inner products calculated from the vector representations of these waveforms. The set of possible received waveforms and resulting inner products are determined directly from the model of the received signal and are completely independent of the form of modulation. Thus the so-called *AOLFP algorithm* can be used to determine the best possible predetection filter for use in virtually any binary communication system employing a linear receiver.

A necessary constraint involved in the application of the AOLFP algorithm is that the impulse response of the predetection filter be of finite duration. This is not a particularly limiting constraint however because any duration may be chosen. This freedom is especially useful when applying the algorithm to signals possessing memory, as it is reasonable to expect that a receiver that bases symbol rate decisions on the received signal over several baud intervals would provide better performance than a receiver that is limited in its observation of the received signal to a single baud interval. Since the number of received waveforms that can be processed by the algorithm is limited only by the available computing resources, the algorithm is ideally suited to finding the impulse response of the AOF for any signal possessing memory. An obvious application is therefore the use of the AOF in serial [2] or parallel MSK-type detection of generalized MSK signals, providing the best possible performance for a class of signals already important due to a constant envelope and spectral efficiency.

The AOLFP algorithm has previously been applied [1] to two such signals, specifically duobinary MSK and TFM, under the assumption that the nominal signals are received corrupted only by AWGN. The impulse response and error performance of the resulting nominal predetection filters are given in [1]. The results give the best possible performance that could be achieved with a linear filter in practice, since ISI and other effects will always cause some degradation from the nominal performance. However, if interference is present in a system and understood, one might question the use of the nominal waveforms plus

noise model for the received waveforms. If the interference is sufficiently well-understood that it can be introduced mathematically into the model of the received signal, why not exploit this knowledge by including the effect of the interference in the calculation of the inner product matrix? Since the inner products are calculated directly from the received signal model, the effect of any source of interference or distortion can be included in the model and the AOLFP algorithm can be applied to determine the impulse response of the AOF. This AOF response, to some degree, will equalize the effects of the interference included in the signal model. Provided the signal model closely approximates the actual received signal, the error performance observed in practice will closely approximate the calculated performance of the AOF when the SNR is moderately high.

This thesis will investigate the use of the AOLFP algorithm for finding optimal filters in practical signalling problems where a known perturbation affects a transmitted signal. Unfortunately this situation typically leads to cases involving very large numbers of possible received waveforms. In some cases, the memory space required to store the inner product matrix is beyond practical limits, making it impossible to apply the algorithm directly. To handle these cases, a modification to the algorithm is developed which overcomes this problem, enabling the AOF to be found regardless of the number of possible received waveforms.

The practical signalling problems considered here involve three sources of interference independently applied to a duobinary MSK signal. The AOLFP algorithm is applied to determine the best possible linear filter (AOF) for use in MSK-type detection of each resulting signal, and the corresponding error performance is calculated. The error performance that would result if the signal were detected by the AOF calculated only on the basis of the nominal waveforms, referred to as the *nominal filter*, is also calculated. From these two performance calculations it is determined whether any performance gains result from the inclusion of the interference in the signal model. Specifically, perturbations resulting from phase errors in carrier recovery, adjacent channel interference, and ISI due to bandlimiting at the transmitter are considered.

1.2 Thesis Organization

Central to this thesis is a clear understanding of the AOLFP algorithm. Consequently, Chapter 2 gives a complete development of the algorithm, including a formal statement. Chapter 3 describes the modification needed to apply the algorithm to problems involving a large set of possible received waveforms.

Since the original focus for the algorithm was the application to generalized MSK signals, it was decided to use one of these same signals to investigate the new use of the algorithm. As each case studied in this thesis pertains to MSK-type detection of duobinary MSK, Chapter 4 describes duobinary MSK and indicates how MSK-type detection of the signal is achieved. Application of the AOLFP algorithm to nominal duobinary MSK is also discussed in Chapter 4, and the impulse response and corresponding error performance of the nominal filter for three different durations is given.

Application of the AOLFP algorithm to distorted duobinary MSK is described in Chapter 5 for three types of distortion. Each case is treated independently, with Section 5.2 investigating phase errors in carrier recovery, Section 5.3 investigating ACI, and finally Section 5.4 investigating ISI caused by bandlimiting at the transmitter.

CHAPTER 2

CALCULATING THE AOF - THE AOLFP ALGORITHM

In this chapter the AOF is described and the AOLFP algorithm is developed and stated.

2.1 Preliminaries

The general form of a linear receiver used to detect a baseband signal is shown in Fig. 2.1, where $h_f(t)$ is the impulse response of the predetection filter. For binary signals, the decision device is a simple threshold detector.

Fig. 2.1 Linear receiver structure.

The interval over which the received signal is observed for the purpose of making a single symbol decision is determined by the duration of $h_f(t)$. If this interval is referred to as the *observation interval* (I_{obs}), it is clear that memory in the signal, or an I_{obs} exceeding one symbol interval results in a situation where many distinct waveforms are possible on I_{obs} for a single symbol value. Each waveform that may occur on I_{obs} for a given data symbol (exclusive of noise) is referred to as a *data signature* of the data symbol. Notice that while increasing I_{obs} may or may not result in improved error performance for a given signal, it will result in an increased number of data signatures for each symbol.

Typically the predetection filter is restricted to have an impulse response of finite duration so that each sample value will depend on the received signal over some finite interval. In particular, if the filter is also causal so that the response $h_f(t)$ is nonzero only on the interval $t \in [0, L]$ ($L > 0$), then the observation interval for the general sampling instant T_s is $I_{obs} = \{t \mid t \in [T_s - L, T_s]\}$. If $y_i(t)$ is the filter output for the input data signature $s_i(t)$, then the output sample $y_i(T_s)$ (on which a symbol decision is based) exclusive of noise is given by

$$\begin{aligned} y_i(T_s) &= s_i(t) * h_f(t)|_{t=T_s} \\ &= \int_{T_s-L}^{T_s} s_i(t) h_f(T_s - t) dt \\ &= \int_{I_{obs}} s_i(t) h_f(T_s - t) dt, \end{aligned} \tag{2.1.1}$$

where $*$ denotes convolution. By defining the *correlating waveform* $r(t)$ of the filter as

$$r(t) = h_f(T_s - t), \tag{2.1.2}$$

the output sample can be expressed as the correlation over I_{obs} between the input data signature and the correlating waveform, given by

$$\int_{I_{obs}} s_i(t) r(t) dt. \tag{2.1.3}$$

Thus the linear receiver of Fig. 2.1 can also be viewed as a correlator as shown in Fig. 2.2.

An alternate interpretation results when $s_i(t)$ and $r(t)$ are viewed as vectors in signal space over the interval I_{obs} . If $\bar{s}_i$ and $\bar{r}$ represent the vectors, the output sample can be expressed as the inner product

$$\langle \bar{s}_i, \bar{r} \rangle = \int_{I_{obs}} s_i(t) r(t) dt. \tag{2.1.4}$$

Fig. 2.2 Linear receiver structure using a correlator.

In this thesis the term *signal vector* will be used to refer to the vector form of a data signature, and the term *receiver vector* will be used to refer to the vector form of a correlating waveform. With respect to the receiver then, the received signal is simply a sequence of vector sums of signal vectors and noise vectors on successive observation intervals, and the output samples are a sequence of inner products between data signatures and the AOF vector, plus noise.

In the next Section it will be shown that for the situation discussed here (i.e., where one of a set of possible signal vectors occurs on a given observation interval for a specific symbol value), the error probability is a function of inner products of the type in equation (2.1.4), and that minimization of this function leads quite naturally to the AOF.

2.2 The Origin of the AOF

Attention is now restricted to binary communication signals with an alphabet of $+1$ and -1 where for some I_{obs} , $S^+ = \{\bar{s}_1^+, \bar{s}_2^+, \dots, \bar{s}_N^+\}$ is the set of signal vectors for the $+1$ symbol, $S^- = \{\bar{s}_1^-, \bar{s}_2^-, \dots, \bar{s}_N^-\}$ is the set of signal vectors for the -1 symbol, and $\bar{s}_i^- = -\bar{s}_i^+$ for all $i = 1 \dots N$. Since most binary signals that occur in practice can be described in this manner, this is not a particularly limiting restriction. The sets S^+ and S^- will be referred to as the *signal sets* for the $+1$ and -1 symbols respectively. With these signal sets, the received vector on each observation interval is a signal vector $\bar{s}_i \in S^+$ plus noise when a $+1$ is transmitted and a signal vector $-\bar{s}_i \in S^-$ plus noise

when a -1 symbol is transmitted. The output sample Y_i due to the data signature $s_i(t)$ including noise is the decision variable of the receiver and is given by

$$Y_i = \langle \bar{s}_i + \bar{n}, \bar{r} \rangle = \langle \bar{s}_i, \bar{r} \rangle + \langle \bar{n}, \bar{r} \rangle \quad (2.2.1)$$

where $\bar{n}$ denotes the noise vector. The noise process $n(t)$ is modelled as a zero mean, stationary, white Gaussian random process with two-sided power spectral density $\frac{N_0}{2}$ W/Hz. Since the filter described by $\bar{r}$ is linear time-invariant, the output noise sample $\langle \bar{n}, \bar{r} \rangle$ in (2.2.1) is a zero mean Gaussian random variable. Denoting this random variable by X , the variance of X is

$$\text{Var}(X) = E\{X^2\} = E\left\{ \int_{I_{obs}} n(\lambda) r(\lambda) d\lambda \int_{I_{obs}} n(\alpha) r(\alpha) d\alpha \right\}, \quad (2.2.2)$$

where E is the expectation operator. Interchanging the order of the expectation and the integration operations and noting that $r(t)$ is a deterministic function,

$$\text{Var}(X) = \int_{I_{obs}} \int_{I_{obs}} r(\lambda) r(\alpha) E\{n(\lambda) n(\alpha)\} d\alpha d\lambda. \quad (2.2.3)$$

Since $n(t)$ is a stationary process, the expectation term in (2.2.3) can be replaced by the autocorrelation function $R_{nn}(\lambda - \alpha)$, which for the white Gaussian noise process is $\frac{N_0}{2} \delta(\lambda - \alpha)$. Substituting this in (2.2.3) we find

$$\text{Var}(X) = \frac{N_0}{2} \int_{I_{obs}} \int_{I_{obs}} r(\lambda) r(\alpha) \delta(\lambda - \alpha) d\lambda d\alpha = \frac{N_0}{2} \|\bar{r}\|^2 \quad (2.2.4)$$

where the notation $\|\cdot\| = \langle \cdot, \cdot \rangle^{1/2}$ is the *norm* of a vector. Since $\langle \bar{s}_i, \bar{r} \rangle$ is a constant, the decision variable Y_i is a Gaussian random variable with mean $\langle \bar{s}_i, \bar{r} \rangle$ and variance $\frac{N_0}{2} \|\bar{r}\|^2$. Assuming equally likely data symbols, equally likely data signatures for each symbol, and a zero threshold for decisions, the probability of making a decision error is

$$P_e(\bar{r}) = \frac{1}{N} \sum_{i=1}^N Q\left\{ \frac{\langle \bar{s}_i, \bar{r} \rangle}{\|\bar{r}\|} \sqrt{\frac{2}{N_0}} \right\} \quad \text{for } \bar{s}_i \in S^+, \quad (2.2.5)$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$.

Since changing $\bar{r}$ by a scale factor changes both the inner products in (2.2.5) and $\|\bar{r}\|$ by the same factor, the receiver vector is constrained to have unit norm. With this constraint, a unique optimal impulse response can be found for the predetection filter by finding the unique unit vector that minimizes the error probability in (2.2.5). It is clear that any such optimal vector must lie in the span of the signal vectors, since any component orthogonal to the spanning set will always contribute zero to the inner product terms in (2.2.5). As a result, it is not necessary to explicitly determine a spanning set for the signal vectors to obtain an expression for the optimal vector, because any vector in the span can always be expressed as a linear combination of all the vectors in the signal set. Thus, the optimal vector can always be expressed in the form

$$\sum_i \alpha_i \bar{s}_i \quad \text{for } \bar{s}_i \in S^+ \quad (2.2.6)$$

for some set of coefficients $\{\alpha_i\}$.

Finding the coefficients $\{\alpha_i\}$ that minimize the error probability for a particular SNR requires setting the gradient of P_e with respect to each coefficient α_i to zero. Unfortunately, this results in a system of transcendental equations and the truly optimal receiver vector is difficult to find in general, even for a small number of data signatures. When N is large the problem is extremely difficult, if not impossible, and a simpler solution method that is easily applicable even when N is large is desirable.

The minimization of P_e is simpler in the limiting case of infinitely small N_0 , or equivalently, infinitely large SNR. Since $Q(x)$ for large x is well approximated by $\exp(-x^2/2)/\sqrt{2\pi}x$, the summation of (2.2.5) in this limiting case is dominated by the term with the smallest inner product between the receiver vector and any signal vector. Thus to minimize the error probability the optimal vector must be the unit vector that maximizes

$$\min_i \{ \langle \bar{s}_i, \bar{r} \rangle \} \quad \bar{s}_i \in S^+. \quad (2.2.7)$$

As stated in Section 2.1, an inner product is in reality the noiseless sampled output of the predetection filter when a specific data signature is the input. Consequently the requirement in (2.2.7) can be interpreted as finding the filter response that maximizes the eye opening of the filter output at the sampling instant. As this filter provides optimum error performance in the limiting case of infinitely large SNR, it is called the *asymptotically optimal filter* (AOF) and will be denoted as $\bar{r}_x$. In addition, because symbol decisions are based entirely on the sample values in the presence of noise, the AOF will provide the best possible performance at practical SNR's, simply because it maximizes the decision eye opening. Furthermore, the fact that the AOF maximizes the eye opening removes any importance on the assumption of equally likely data symbols and data signatures (except where equation (2.2.5) is concerned), and the AOF is the optimum filter for infinitely large SNR, independent of this assumption.

The AOLFP algorithm is an efficient method for finding the AOF vector for a given signal set S^+ and is presented fully in [6]. As the purpose of this thesis is to demonstrate the application of the AOLFP algorithm in practical cases of interest, the remainder of this Chapter is devoted to a discussion of the algorithm, and parallels the discussion given in [6].

2.3 Properties of the AOF Vector

For any receiver vector $\bar{r}$, let

$$d_{\min}(\bar{r}) = \min_i \langle \bar{s}_i, \bar{r} \rangle \quad \text{for } \bar{s}_i \in S^+. \quad (2.3.1a)$$

With this definition, the AOF vector $\bar{r}_x$ is the unit vector that maximizes $d_{\min}(\bar{r})$. In general for a vector $\bar{r}$ there will be at least one and perhaps several vectors in the signal set whose inner products with $\bar{r}$ are equal to $d_{\min}(\bar{r})$. These vectors are referred to as the *active vectors* of $\bar{r}$. In particular the active vectors of the AOF vector $\bar{r}_x$ are

indexed by the set I_x . Thus for any signal set S^+ with AOF vector $\bar{r}_x$, there exists a unique active vector set $I_x \subset \{1, 2, \dots, N\}$ such that

$$\begin{aligned} \langle \bar{s}_i, \bar{r}_x \rangle &= d_{\min}(\bar{r}_x) & \text{for } i \in I_x \\ \langle \bar{s}_j, \bar{r}_x \rangle &> d_{\min}(\bar{r}_x) & \text{for } j \notin I_x. \end{aligned} \quad (2.3.1b)$$

This is the fundamental property that, in conjunction with the following result from [6], will be used to find the AOF vector.

Theorem 1: *Let $\{\bar{s}_i\}_{i \in I}$ be a set of vectors. Let $\bar{w}$ be a unit vector with the properties that $\langle \bar{w}, \bar{s}_i \rangle$ is the same positive value for all $i \in I$ and that the value of the inner product is maximized subject to this constraint. Then $\bar{w}$ is unique and is given by*

$$\bar{w} = \sum_{i \in I} \alpha'_i \bar{s}_i, \quad (2.3.2a)$$

with the column vector $\bar{\alpha}'_I = [\alpha'_i]_{i \in I}^t$ given by $\bar{\alpha}'_I = \bar{\alpha}_I / \sqrt{\sum_{i \in I} \alpha_i}$, where for A_I the matrix of inner products $\langle \bar{s}_i, \bar{s}_j \rangle$, $i, j \in I$, and $\bar{1} = [1, \dots, 1]^t$, $\bar{\alpha}_I = [\alpha_i]_{i \in I}^t$ satisfies

$$A_I \bar{\alpha}_I = \bar{1}. \quad (2.3.2b)$$

If the active vector set $\{\bar{s}_i\}_{i \in I_x}$ of the AOF vector is known, the vector $\bar{w}$ given by the solution to the system of equations in (2.3.2b) with I replaced by I_x , will maximize the quantity $\langle \bar{s}_i, \bar{w} \rangle$ for $i \in I_x$ and will also guarantee $\langle \bar{s}_i, \bar{w} \rangle = \langle \bar{s}_j, \bar{w} \rangle$ for any $i, j \in I_x$. The vector $\bar{w}$ is therefore the AOF vector $\bar{r}_x$.

Since the active vectors are the signal vectors that result in the minimum inner product with the AOF vector, the corresponding data signatures result in the minimum sample values of the predetection filter output. Physically then, these waveforms form the inner eye of the signal at the filter output and can therefore be thought of as the “worst case” waveforms in terms of making symbol decisions with minimum error probability. Consequently, the correlating waveform of the AOF is always expressed as a linear combination of the worst case data signatures, the coefficients being determined

from (2.3.2b). The problem of finding the AOF vector can therefore be equated with finding its active vectors.

A crude procedure for finding the AOF vector is then to first determine the vector $\bar{w}_i$ from Theorem 1 for all possible index sets $I \subset \{1, 2, \dots, N\}$, then isolate the subset $\{\bar{w}_i\}_{i \in F}$ with their corresponding index sets $\{I_i\}_{i \in F}$ such that

$$\begin{aligned} \langle \bar{s}_j, \bar{w}_i \rangle &= d_{\min}(\bar{w}_i) & \text{for } j \in I_i \\ \text{and } \langle \bar{s}_j, \bar{w}_i \rangle &> d_{\min}(\bar{w}_i) & \text{for } j \notin I_i \end{aligned} \quad (2.3.3)$$

Finally, from the set F , find the one vector $\bar{w}_j$ such that

$$d_{\min}(\bar{w}_j) > d_{\min}(\bar{w}_i) \quad \text{for } i, j \in F, \quad i \neq j. \quad (2.3.4)$$

The vector $\bar{w}_j$ is then the AOF vector. The vectors in the set $\{\bar{w}_i\}_{i \in F}$ are referred to as *feasible vectors*, and the corresponding index sets $\{I_i\}_{i \in F}$ are the active vector sets of the feasible vectors. In general, a number of feasible vectors exist for a given signal set S^+ , but only the feasible vector $\bar{w}_j$ satisfying (2.3.4) is the AOF vector.

The approach to finding the AOF vector suggested above is clearly inadequate, especially when the number of signal vectors N is large, as the number of possible index sets I increases exponentially with N . The AOLFP algorithm, which makes use of the feasible vectors that exist for each signal set, can be used to find the AOF vector easily even when N is large.

2.4 Development of the AOLFP Algorithm

The AOLFP algorithm is essentially an intelligent search of the signal set for the active vectors of the AOF vector. This is accomplished by generating a sequence of feasible vectors $\{\bar{r}_f\}$, each with a greater value of $d_{\min}(\bar{r}_f)$ than the previous feasible vector, until the feasible vector with maximal $d_{\min}(\bar{r}_f)$ is found. To find the AOF in this way the algorithm repeatedly performs two fundamental tasks:

- (1) Generate a feasible vector.
- (2) Test to see if the feasible vector is optimal.

Before describing how (1) and (2) are done, it is noted that the AOF vector exists for a given signal set S^+ and is unique. The proofs of these statements can be found in [6].

2.4.1 Generating a Feasible Receiver Vector

The method of finding a feasible vector is based on the following theorem [6].

Theorem 2: Let $\bar{r}$ be a receiver vector with unit norm and active vector set $\{\bar{s}_i\}_{i \in I}$. Let $\bar{r}'$ be a distinct unit vector with the property that

$$\langle \bar{s}_i, \bar{r}' \rangle \geq d_{\min}(\bar{r}) \quad \text{for } i \in I. \quad (2.4.1a)$$

Then there exists a unit vector $\hat{r}$, expressable in the form

$$\hat{r} = \sum_{i \in I} \beta_i \bar{s}_i \quad (2.4.1b)$$

for some non-negative set of coefficients $\{\beta_i\}$, for which

$$d_{\min}(\hat{r}) > d_{\min}(\bar{r}). \quad (2.4.1c)$$

Any vector $\bar{r}'$ satisfying (2.4.1a) is referred to as an *improvement vector* of the receiver vector $\bar{r}$, since by way of Theorem 2, $\bar{r}'$ guarantees the existence of a receiver vector with $d_{\min}$ strictly larger than $d_{\min}(\bar{r})$.

The important result of Theorem 2 is that a “better” receiver vector (i.e, one with a larger $d_{\min}$) can be obtained by rotating an initial receiver vector $\bar{r}$ toward an improvement vector $\bar{r}'$. Note that $\bar{r}$ and $\bar{r}'$ are constrained to be of unit norm and that this unity norm constraint is maintained during rotation. Since the unit vector $\bar{w}$ having the largest inner product with the active vectors $\{\bar{s}_i\}_{i \in I}$ of any receiver vector

$\bar{r}$ can be found by applying Theorem 1, if $\langle \bar{s}_i, \bar{w} \rangle \geq d_{\min}(\bar{r})$ for $i \in I$, then $\bar{w}$ is an improvement vector for $\bar{r}$ provided $\bar{w} \neq \bar{r}$. If $\bar{w}$ is not distinct from $\bar{r}$, then $\bar{r}$ is in fact a feasible vector. Therefore the process of generating $\bar{r}'$ (using Theorem 1) followed by a suitable rotation of $\bar{r}$ toward $\bar{r}'$ can be repeated until $\bar{r}'$ is also a feasible vector. When this occurs $\bar{r}$ is rotated until it is equal to $\bar{r}'$ and tested for optimality.

The method of rotating $\bar{r}$ towards $\bar{r}'$ is essential to the algorithm, as it determines the sequence of receiver vectors that ultimately ends with the AOF vector. To see how a suitable rotation is determined, first note that the line joining the two vectors $\bar{r}$ and $\bar{r}'$ can be expressed by the vector

$$\bar{r}(\lambda) = \lambda \bar{r}' + (1 - \lambda) \bar{r} \quad \text{for } \lambda \in [0, 1], \quad (2.4.2)$$

where λ increasing from 0 to 1 corresponds to a movement of $\bar{r}$ along a straight line towards $\bar{r}'$ for which the unity norm constraint is not maintained. Next, let the *trajectory* of any signal vector $\bar{s}_i$ be defined as

$$T_i(\lambda) = \langle \bar{s}_i, \bar{r}(\lambda) \rangle = \lambda \langle \bar{s}_i, \bar{r}' \rangle + (1 - \lambda) \langle \bar{s}_i, \bar{r} \rangle \quad \text{for } \lambda \in [0, 1]. \quad (2.4.3)$$

When $\bar{r}$ is nonfeasible, the calculation of $\bar{r}'$ from Theorem 1 ensures that $\langle \bar{s}_i, \bar{r}' \rangle$ is constant for $\bar{s}_i$ any active vector of $\bar{r}$, and the definition of the active vectors ensures that $\langle \bar{s}_i, \bar{r} \rangle$ is also constant. As a result the trajectories of all active vectors of $\bar{r}$ are identical, and reach a maximum value of $\langle \bar{s}_i, \bar{r}' \rangle$ at $\lambda = 1$. This along with the fact that trajectories are linear functions of λ implies that any two trajectories will intersect for either a single value of λ , no value of λ , or all values of λ . Since $T_j(0) > T_i(0)$ for $\bar{s}_i$ any active vector of $\bar{r}$ and $\bar{s}_j$ a nonactive vector, $T_j(\lambda)$ and $T_i(\lambda)$ will intersect for some $0 < \lambda < 1$ only if $T_j(1) < T_i(1)$. The intersection point in this case occurs when $\langle \bar{s}_i, \bar{r}(\lambda) \rangle = \langle \bar{s}_j, \bar{r}(\lambda) \rangle$, from which the value of λ is

$$\lambda = \frac{\langle \bar{s}_j - \bar{s}_i, \bar{r} \rangle}{\langle \bar{s}_j - \bar{s}_i, \bar{r} - \bar{r}' \rangle} \quad (2.4.4)$$

The rotation of $\bar{r}$ toward $\bar{r}'$ is determined by the rotation factor λ^* , defined to be that value of λ corresponding to the first intersection of $T_i(\lambda)$ and $T_j(\lambda)$. Thus λ^* is given by the minimum value of λ satisfying (2.4.4). When $T_j(1) \geq T_i(1)$ for all nonactive vectors $\bar{s}_j$, then $\lambda^* = 1$ and the new receiver vector becomes $\bar{r}'$, which is a feasible vector with the same active vectors as $\bar{r}$. Combining the two possible cases for λ^* and noting that (2.4.4) can be written as

$$\lambda = \frac{1}{1 - \frac{\langle \bar{s}_j - \bar{s}_i, \bar{r} \rangle}{\langle \bar{s}_j - \bar{s}_i, \bar{r} - \bar{r}' \rangle}}, \quad (2.4.5)$$

the rotation factor λ^* for $\bar{s}_i$ any active vector of $\bar{r}$ and $\bar{s}_j$ a nonactive vector, is

$$\lambda^* = \left[1 - \min \left\{ \left\{ \frac{\langle \bar{s}_j - \bar{s}_i, \bar{r}' \rangle}{\langle \bar{s}_j - \bar{s}_i, \bar{r} \rangle} \right\} \mid \langle \bar{s}_j - \bar{s}_i, \bar{r}' \rangle < 0 \right\} \cup 0 \right]^{-1}. \quad (2.4.6)$$

These two possible cases for λ^* are illustrated in Fig. 2.3(a) and Fig. 2.3(b). Each figure shows the trajectories of 3 nonactive vectors T_1 , T_2 and T_3 , and the trajectory of the active vectors T_{active} . In Fig. 2.3(a), λ^* is determined by the intersection of T_3 and T_{active} , since the intersection of T_2 and T_{active} occurs for a larger value of λ . In Fig. 2.3(b), no trajectories intersect T_{active} so that $\lambda^* = 1$, resulting in a feasible vector.

Upon rotation, the new receiver vector is $\frac{\bar{r}(\lambda^*)}{\|\bar{r}(\lambda^*)\|}$ (so as to have unity norm), with an active vector set consisting of the active vectors of $\bar{r}$, plus the nonactive vector(s) whose trajectory first intersects the trajectory of active vectors. For example, in Fig. 2.3(a) the new set of active vectors would be the active vectors of $\bar{r}$ plus $\bar{s}_3$.

In determining the sequence of rotation factors that lead to the feasible vector, the algorithm essentially performs an intelligent search for the active vectors of the AOF vector. The active vector set grows and d_{min} increases for each new nonfeasible vector until a feasible vector is found when $\lambda^* = 1$. Since the number of signal vectors is finite, a feasible vector is always found after a finite number of rotations.

The algorithm can be simplified slightly by noting that a change in scale of $\bar{r}$, $\bar{r}'$ or $\bar{r}(\lambda^*)$ has no effect on the calculation of λ^* , the active vectors of $\bar{r}$, or the fact

Fig. 2.3 Possible trajectories for: (a) a nonfeasible vector, and (b) a feasible vector.

that $\bar{r}(\lambda)$ represents the family of vectors between $\bar{r}$ and $\bar{r}'$. Therefore, the unity norm constraint can be dropped for intermediate receiver vectors. This means that when solving (2.3.2b) to find $\bar{r}'$, only the column vector $\bar{\alpha}_I$ is needed and $\bar{r}(\lambda^*)$ need not be normalized after each rotation. The only receiver vector that must be normalized is the AOF vector $\bar{r}_x$.

2.4.2 Testing Feasible Vectors for Optimality

As previously stated, the optimal receiver vector is the feasible vector with maximal d_{min} . Once a feasible vector is found, it is tested for optimality by Theorem 3 which is proved in [6].

Theorem 3 (The First Optimality Criterion): *Let $\bar{r}$ be a feasible vector with $d_{min}(\bar{r}) > 0$. Then $\bar{r}$ is the AOF vector if and only if it can be expressed as a linear combination of its active vectors with non-negative coefficients.*

Since a feasible vector is always ultimately produced by Theorem 1, it can always be expressed as a linear combination of its active vectors. If all of the coefficients are non-negative, then the vector is the AOF vector by the statement of Theorem 3. There is no guarantee however that application of Theorem 1 will produce a set of non-negative coefficients, even when such a set is possible, and thus a more general optimality test is required. Fortunately, a result proved in [6] and referred to here as Corollary 1 can be used to generate an improvement vector for suboptimal feasible vectors, and in so doing test any feasible vector for optimality.

Corollary 1: *If $\bar{r}_f$ is a suboptimal feasible vector, then an improvement vector is the closest vector $\bar{r}'_f$ that can be expressed as a linear combination of the active vectors of $\bar{r}_f$ with non-negative coefficients.*

To find $\bar{r}'_f$, the set of coefficients $\{\beta_i | \beta_i \geq 0\}_{i \in I}$ are required such that

$$\bar{r}'_f = \sum_{i \in I} \beta_i \bar{s}_i \quad (2.4.7)$$

for $\{\bar{s}_i\}_{i \in I}$ the active vectors of $\bar{r}_f$. The set $\{\beta_i\}$ is found by minimizing the Euclidean distance between $\bar{r}_f$ and $\bar{r}'_f$, subject to the constraint that $\beta_i \geq 0$ for all $i \in I$. An expression for $\bar{r}_f$ in the form

$$\bar{r}_f = \sum_{i \in I} \alpha_i \bar{s}_i \quad (2.4.8)$$

is known from Theorem 1, so that the distance z between $\bar{r}_f$ and $\bar{r}'_f$ is

$$z = (\bar{\beta}_I - \bar{\alpha}_I)^t A_I (\bar{\beta}_I - \bar{\alpha}_I) \quad (2.4.9)$$

where A_I and $\bar{\alpha}_I$ are as defined in Theorem 1 and $\bar{\beta}_I$ is the column vector $[\beta_i]_{i \in I}^t$. (Note that A_I is in general a positive semi-definite matrix.) This is a quadratic programming problem and can be solved by a variety of well-known methods. If the minimum value of z is zero, the vector $\bar{r}_f$ is the AOF vector.

When $z_{\min}$ is nonzero, a new receiver vector must be obtained by rotating $\bar{r}_f$ towards $\bar{r}'_f$. The situation for this rotation is somewhat different because in finding $\bar{r}'_f$ there is no restriction that $\langle \bar{s}_i, \bar{r}'_f \rangle$ be equal for all $i \in I$, as is required for an improvement vector for a nonfeasible vector. As a result, the trajectories of the active vectors no longer coincide for all $\lambda \in [0, 1]$. However since trajectories are linear, any active vector of $\bar{r}_f$ that is not also an active vector of $\bar{r}'_f$ must have a trajectory above the trajectory of active vectors of $\bar{r}'_f$ and hence will not determine the rotation factor λ^* . Therefore, the rotation technique previously discussed can also be used here, provided the trajectory of active vectors of $\bar{r}'_f$ is used and not the trajectory of the active vectors of $\bar{r}_f$. The active vectors of $\bar{r}'_f$ can be found by directly calculating $\langle \bar{s}_i, \bar{r}'_f \rangle$ for all $\bar{s}_i \in S^+$ and choosing the vectors that minimize these inner products. It is important to note that the vector resulting from the rotation of $\bar{r}_f$ towards $\bar{r}'_f$ is not necessarily a feasible vector. In this case the algorithm begins again with this new starting vector.

All of the information needed to find the AOF vector is contained in the inner products. Therefore, although the algorithm is most easily understood through a geometric interpretation (i.e, vectors in signal space), it is never necessary to express the signal vectors in terms of basis vectors. All that is needed is the matrix of inner products A with elements

$$a_{ij} = \langle \bar{s}_i, \bar{s}_j \rangle \quad \text{for all } \bar{s}_i, \bar{s}_j \in S^+, \quad (2.4.10)$$

and this must be calculated before the algorithm can be applied.

2.4.3 Algorithm Statement

The AOLFP algorithm can be summarized as follows. First, a starting vector equal to one of the signal vectors is selected, its active vectors are found, and Theorem 1 is applied to find the improvement vector. The vector is then moved toward the improvement vector and the active vector set is updated. These steps are repeated until a feasible vector is found. If the expression for the feasible vector involves positive coefficients only then it is the AOF, otherwise the quadratic programming problem is solved, either proving the feasible vector to be optimal or generating an improvement vector. This process of rotation and generating an improvement vector repeats until some feasible vector satisfying the optimality criterion is found. That feasible vector is then normalized.

The AOLFP algorithm is formally stated below. In step 1, a nonfeasible vector is represented by $\bar{r}$ and improvement vectors for nonfeasible vectors are represented by $\bar{r}' = \sum_{i \in I} \alpha_i \bar{s}_i$. In step 2, the improvement vectors for feasible vectors are represented by $\bar{r}'_f = \sum_{i \in I} \beta_i \bar{s}_i$. The terms $\bar{\alpha}_I$, A_I and $\bar{1}$ are as defined in Theorem 1. The AOF vector is represented by $\bar{r}_x$.

Step 0: Initialization.

Set $\bar{r} = \bar{s}_j$ for some $j = \{1 \dots N\}$, and set $\rho_i = \langle \bar{s}_i, \bar{s}_j \rangle$ for all i .

Find $d_{\min} = \min_i \rho_i$ and the set of indices I where $\rho_i = d_{\min}$.

If $d_{\min} = \rho_j$, set $\bar{r}_x = \bar{s}_j / \|\bar{s}_j\|$.

Stop.

Step 1: Find a Feasible Vector.

A. Find an Improvement Vector.

Solve $A_I \bar{\alpha}_I = \bar{1}$.

Compute $\rho'_j = \langle \bar{s}_j, \bar{r}' \rangle = \sum_{i \in I} \alpha_i \langle \bar{s}_i, \bar{s}_j \rangle$ for all j , and set $d'_{\min} = \rho'_j$ for any $j \in I$.

B. Rotation of $\bar{r}$ Towards $\bar{r}'$.

Set $\lambda^* = 0$.

For $j = 1, 2, \dots, N$ $j \notin I$, compute $\Delta_j = \rho'_j - d'_{\min}$ and if $\Delta_j < 0$, set

$$\lambda^* = \min_j \{\lambda^*, \Delta_j / (\rho_j - d_{\min})\}.$$

Add to the set I the indices j for which $\lambda^* = \Delta_j / (\rho_j - d_{\min})$.

Set $\lambda^* = 1 / (1 - \lambda^*)$, and $\bar{r} = (1 - \lambda^*)\bar{r} + \lambda^*\bar{r}'$.

Compute $\rho_i = \langle \bar{s}_i, \bar{r} \rangle = (1 - \lambda^*)\rho_i + \lambda^*\bar{\rho}'_i$ for all $i = 1 \dots N$.

Find $d_{\min} = \min_i \rho_i$.

If $\lambda^* = 1$, go to Step 2.

Go to Step 1.

Step 2: Generating an Improvement Vector for a Feasible Vector.

If all coefficients describing $\bar{r}$ ($\bar{\alpha}_I$ or $\bar{\beta}_I$) are positive, go to Step 3.

Solve the quadratic program

$$\text{minimize } z = (\bar{\beta}_I - \bar{\alpha}_I)^t A_I (\bar{\beta}_I - \bar{\alpha}_I), \text{ subject to } \bar{\beta}_I \geq 0.$$

If $z_{\min} = 0$ go to step 3.

Compute $\rho'_j = \langle \bar{s}_j, \bar{r}'_f \rangle = \sum_{i \in I} \beta_i \langle \bar{s}_i, \bar{s}_j \rangle$ for all j , and set $d'_{\min} = \min_{j \in I} \rho'_j$.

Reduce the set I so that for $j \in I$, $\rho'_j = d'_{\min}$.

Set $\bar{r}' = \bar{r}'_f$.

Go to B in Step 1.

Step 3: Normalize the AOF Vector.

Set $\bar{r}_x = \bar{r} / \|\bar{r}\|$.

Stop.

Note that Step 0 includes the unlikely situation that the AOF vector is the starting vector, chosen arbitrarily as one of the signal vectors. As discussed in [6], this can only occur if the starting vector corresponds to a data signature that forms the inner eye and would not be expected to occur in practice.

2.5 Implications of the AOLFP Algorithm

The AOLFP algorithm presented in this chapter can be applied any time a received binary communication signal has the property that many possible waveforms may occur for a single data symbol on each observation interval. As discussed in Chapter 1, this can be a property of the modulated signal itself, or can arise from interference acting on the signal during transmission.

In cases where the various sources of interference acting on the transmitted signal are well-understood, it is possible to formulate a signal model that accurately describes the received signal at the input to the predetection filter. From the signal model, accurate inner product calculations can then be made for the data signatures that occur for each symbol value. By including the effect of known interference sources in the calculation of the inner product matrix, the AOLFP algorithm can be applied to determine the impulse response of the predetection filter that optimizes the error performance. Moreover, since only the inner product matrix is required, the algorithm can be applied to a sufficiently accurate signal model for any modulation technique combined with any number of different interference sources.

Essentially then, the AOLFP algorithm can be used to determine a linear filter response that provides some measure of equalization and optimizes the error performance when interference corrupts the transmitted signal. It can also be used to determine the best error performance that can be achieved with a linear receiver.

CHAPTER 3

PRACTICAL CONSIDERATIONS IN USING THE AOLFP ALGORITHM

The first Section of this chapter discusses the need to manage the available memory space in a software implementation of the AOLFP algorithm and gives details for a solution to the quadratic programming problem. In the second Section, a technique is presented for applying the algorithm when the signal set contains a large number of signal vectors.

3.1 Implementing the AOLFP Algorithm

3.1.1 Memory Space Considerations

A software implementation of the AOLFP algorithm must make efficient use of the available memory space. Since the algorithm cannot begin without the inner product matrix A containing the elements a_{ij} in (2.4.10), these values must be calculated and stored first. An important, albeit obvious point with respect to storing these inner products is that $a_{ij} = a_{ji}$ so that A is a symmetric matrix and only the upper triangular portion need be stored. As the number of signal vectors N increases, the memory required to store the $\frac{1}{2}(N^2 + N)$ elements of the upper triangle can quickly become extreme. In addition to these inner products, as the algorithm proceeds the values $\langle \bar{s}_j, \bar{r} \rangle$ and $\langle \bar{s}_j, \bar{r}' \rangle$ (ρ_j and ρ'_j in Section 2.4.3) must be stored for every signal vector. Moreover, once the AOF vector is found, calculation of the performance (using equation 2.2.5) requires that $\langle \bar{s}_i, \bar{r}_x \rangle$ be calculated and stored for every signal vector. Additional memory is required each time $\bar{r}'$ is found, as this requires the solution to a system of linear equations whose dimension increases until $\bar{r}_f$ is found. These considerations

demand the use of dynamic memory allocation in a software implementation of the algorithm.

Since the number of signal vectors in a signal set changes from application to application, the dimension of the inner product matrix changes as well. One approach to storing the upper triangular portion of A is to define several groups of dynamic memory, allocating enough memory for the top row of the matrix to the first group, enough memory for the second row (one less memory location) to the second group and so on, until the last group is allocated a single location to store the element a_{nn} . The elements of A can then be stored row by row in each group of memory locations. In this way, a change in the number of signal vectors simply requires changing the number of groups of memory locations. This type of approach is also convenient when finding a feasible vector. As discussed in Section 2.3, the number of simultaneous equations that must be solved in finding an improvement vector is equal to the number of active vectors of the nonfeasible vector. Since this number steadily increases until a feasible vector is found, new rows and columns are constantly added to the matrix A_I in (2.3.2b). As there is no way of knowing beforehand how many active vectors exist for each feasible vector, it is once again convenient to dynamically allocate memory space for each new row of A_I as it occurs, as well as increasing the memory space for the existing rows. Once the coefficients of the feasible vector are found, all the memory space allocated for the matrix A_I can be freed for other use.

The greatest benefit of using dynamic memory allocation is that it allows efficient management of the available memory space. For example, by dynamically allocating memory space for the sets of inner products $\{\rho_j\}$ and $\{\rho'_j\}$ mentioned above, this memory can be used again after $\bar{r}_x$ is found, to store the set of inner products $\{\langle \bar{s}_i, \bar{r}_x \rangle\}$ for all signal vectors needed to calculate the error performance. This managing of the available memory space is important when computing on personal computers with limited memory space, and becomes important even for mainframe computers when the number of signal vectors is large.

3.1.2 Solving the Quadratic Programming Problem

As mentioned in Section 2.4.2, a variety of methods exist for solving the quadratic programming problem. The method used in this thesis is the simplex algorithm taken from [7], for which the problem is stated in the following form. Maximize with respect to the vector $\bar{x}$ the function:

$$F(\bar{x}) = \bar{p}^t \bar{x} - \frac{1}{2} \bar{x}^t C \bar{x} \quad (3.1.1)$$

subject to the linear constraint equations

$$A\bar{x} = \bar{b}, \quad (3.1.2a)$$

and the inequalities

$$\bar{x} \geq \bar{0}. \quad (3.1.2b)$$

The vector $\bar{x}$ is a column of n variables $x_1 \dots x_n$, A is an $m \times n$ matrix such that $m < n$, $\bar{p}$ is a known column vector of m elements, and C is assumed to be an $n \times n$ symmetric positive semi-definite matrix. The problem stated in Section 2.4.2 can be restated as:

$$\begin{aligned} \text{maximize } z &= -(\bar{\beta}_I - \bar{\alpha}_I)^t A_I (\bar{\beta}_I - \bar{\alpha}_I) \\ &= -\left\{ \bar{\beta}_I^t A_I \bar{\beta}_I - \bar{\alpha}_I^t A_I \bar{\beta}_I - \bar{\beta}_I^t A_I \bar{\alpha}_I + \bar{\alpha}_I^t A_I \bar{\alpha}_I \right\}. \end{aligned} \quad (3.1.3)$$

Since the objective is to find the vector $\bar{\beta}_I$ that maximizes z , the constant $\bar{\alpha}_I^t A_I \bar{\alpha}_I$ can be removed so that the problem becomes

$$\text{maximize } z = \bar{\beta}_I^t A_I \bar{\alpha}_I + \bar{\alpha}_I^t A_I \bar{\beta}_I - \bar{\beta}_I^t A_I \bar{\beta}_I. \quad (3.1.4)$$

Now the optimality test is no longer $z_{\min} = 0$ but $z_{\max} - \bar{\alpha}_I^t A_I \bar{\alpha}_I = 0$, and the constant term must be evaluated before testing for optimality. Since $\bar{\beta}_I^t A_I \bar{\alpha}_I$ evaluates to a constant,

$$\bar{\beta}_I^t A_I \bar{\alpha}_I = \bar{\alpha}_I^t A_I \bar{\beta}_I, \quad (3.1.5)$$

and as the vector $\bar{\beta}_I$ that maximizes z also maximizes $\frac{z}{2}$, the problem can be stated in the form of (3.1.1) as

$$\text{maximize } \frac{z}{2} = \bar{\alpha}_I^t A_I \bar{\beta}_I - \frac{1}{2} \bar{\beta}_I^t A_I \bar{\beta}_I \quad (3.1.6)$$

subject to $\bar{\beta}_I \geq 0$. By comparison to (3.1.1)

$$\bar{p} = A_I \bar{\alpha}_I,$$

$$C = A_I, \quad (3.1.7)$$

$$\text{and } \bar{x} = \bar{\beta}_I.$$

The matrix $C = A_I$ is positive semi-definite as required. Note that there are no linear constraint equations in this problem so that $A = 0$ and $\bar{b} = 0$ in (3.1.2a), and only the constraints in (3.1.2b) apply. The quadratic programming problems encountered in this thesis have been solved by applying (3.1.6) to the method in [7], where the dimension n is the number of active vectors of the feasible vector.

3.2 Modifying the AOLFP Algorithm for Large Signal Sets

3.2.1 Development of the Extended AOLFP Algorithm

It was mentioned in Section 3.1 that before the AOLFP algorithm can begin, the inner product matrix A for the entire signal set S^+ must be calculated and stored. Since the upper triangular part of this $N \times N$ matrix contains $\frac{1}{2}(N^2 + N)$ elements, it becomes problematic, with respect to available memory space, to deal with signal sets for which N is large. However, many cases of interest, and indeed the cases studied in this thesis, involve signal sets that are so large, calculating the necessary inner products would require an unreasonable amount of time and memory space. It is therefore desirable to reduce the number of signal vectors the algorithm must deal with. This issue was partly addressed in [1] where the number of signal vectors was reduced by exploiting both symmetry in the received signal about the center of the observation interval and

linear dependencies among the data signatures. Such approaches are useful for filter design involving a limited number of data signatures, but in general cannot be used in cases where interference effects destroy the symmetry and result in signal sets that are so large that linear dependencies would be too time consuming and difficult to find. A new approach allowing the AOLFP to be solved regardless of the number of signal vectors is clearly needed for such cases.

Recall from Chapter 2 that the problem of finding the AOF vector is equivalent to the problem of finding its active vector set I_x , since the AOF vector can then be found simply by applying Theorem 1. This is true regardless of the size of the signal set. This simple fact makes it possible to extend the search method for I_x provided by the AOLFP algorithm so that I_x (which is typically a much smaller set than the signal set) and hence the AOF vector can be found even when the signal set is large.

In this extended search method, the AOLFP algorithm is applied to a sequence of subsets of the signal set $\{G^{(i)} | G^{(i)} \subset S^+\}$, each subset small enough that the AOLFP algorithm can be applied to it with the computing resources available. The key to this search method is that each successive subset $G^{(i)}$ contains at least as many members of the active vector set I_x as the previous subset $G^{(i-1)}$. In this way, the AOF vector calculated for each subset $G^{(i)}$, referred to as $\bar{r}^{(i)}$, is a better approximation to the AOF vector of the signal set $\bar{r}_x$ than the previous solution $\bar{r}^{(i-1)}$. This is because each $\bar{r}^{(i)}$ is expressed as a linear combination of more members of I_x than the previous vector $\bar{r}^{(i-1)}$. After a finite number of iterations, some subset $G^{(k)}$ is identified that includes the entire active vector set I_x . Application of the AOLFP algorithm to $G^{(k)}$ produces the vector $\bar{r}^{(k)}$ which must be identical to the AOF vector $\bar{r}_x$ because $\bar{r}_x$ is expressed in terms of the vectors in I_x , and the AOLFP algorithm will always identify the active vectors of the given signal set and determine the linear combination that defines $\bar{r}_x$, regardless of how many nonactive vectors are included.

As a general and completely arbitrary example of how a typical solution might proceed, consider a case where the active vector set I_x consists of 10 vectors. Finding $\bar{r}_x$

could take 4 iterations, with $G^{(1)}$ containing 2 of the active vectors, $G^{(2)}$ also containing 2 of the active vectors, $G^{(3)}$ containing 6 active vectors, and $G^{(4)}$ finally containing all 10 members of I_x .

Instead of choosing the initial set $G^{(1)}$ at random, it is logical to choose $G^{(1)}$ in such a way that it is representative of the signal set as a whole. If no signal vectors are completely omitted in finding $\bar{r}^{(1)}$, then $\bar{r}^{(1)}$ will be an approximation to the AOF vector of the entire signal set, and not an approximation to the AOF vector of a subset of the entire signal set. Essentially, this means that every signal vector in the signal set must be included in some way in the process of finding $G^{(1)}$. Toward this end, the subset $G^{(1)}$ is found by comparing the distance between various pairs of signal vectors. From the magnitudes of various difference vectors, the signal set is partitioned into many subsets, such that all signal vectors in a particular subset are at most a distance 2ϵ from any other signal vector in the subset. The initial subset $G^{(1)}$ is then comprised of one signal vector from each subset of the partition. In this way a smaller signal set can be identified that is representative of the entire signal set. How "close" the vectors in a particular subset are to one another depends on the parameter ϵ .

Consider the difference between $\langle \bar{s}_1, \bar{r} \rangle$ and $\langle \bar{s}_2, \bar{r} \rangle$ for any signal vectors $\bar{s}_1$ and $\bar{s}_2$, and any receiver vector $\bar{r}$ with $\|\bar{r}\| = 1$. From the Cauchy-Schwarz inequality

$$|\langle \bar{s}_1, \bar{r} \rangle - \langle \bar{s}_2, \bar{r} \rangle| \leq \|\bar{r}\| \|\bar{s}_1 - \bar{s}_2\| = \|\bar{s}_1 - \bar{s}_2\|. \quad (3.2.1)$$

If the norm of the difference vector $\bar{s}_1 - \bar{s}_2$ is known to be less than some positive value ϵ , then $|\langle \bar{s}_1, \bar{r} \rangle - \langle \bar{s}_2, \bar{r} \rangle| < \epsilon$. Since the two inner products involved can be viewed as samples of the output of the predetection filter defined by $\bar{r}$, the difference between these two sample values is less than ϵ . In this case the receiver sees $\bar{s}_1$ and $\bar{s}_2$ as essentially the same signal vector. If $\bar{s}_1$ is not an active vector and ϵ is small, then it is likely that $\bar{s}_2$ is not an active vector, and only one of the pair of vectors needs to be included in $G^{(1)}$. An initial signal set of any size can therefore be determined simply by changing the precise definition of "close", i.e., by changing the value of ϵ .

The process of constructing $G^{(1)}$ begins with adding the first signal vector $\bar{s}_1$ to $G^{(1)}$ and choosing a value for ϵ . If $\|\bar{s}_2 - \bar{s}_1\| > \epsilon$, then $\bar{s}_2$ is also added to $G^{(1)}$. The magnitude of the difference vector $\bar{s}_i - \bar{s}_j$ is then calculated for each remaining signal vector $\bar{s}_i \in S^+$ for every $\bar{s}_j \in G^{(1)}$. If $\|\bar{s}_i - \bar{s}_j\| > \epsilon$ for all $\bar{s}_j \in G^{(1)}$ then $\bar{s}_i$ is included in $G^{(1)}$, otherwise it is neglected. This process continues until all signal vectors $\bar{s}_i \in S^+$ have been tested in this way. Since there is no way to determine in advance how many signal vectors $G^{(1)}$ will contain, and the overall objective is to reduce the number of signal vectors the AOLFP algorithm must deal with, the maximum number of signal vectors in any subset $G^{(i)}$ is limited to some integer n_{max} . The value for n_{max} is chosen in accordance with the available computing resources. The process for finding $G^{(1)}$ therefore stops either when $G^{(1)}$ contains n_{max} vectors, or when all signal vectors $\bar{s}_i$ have been tested.

Each time the AOLFP algorithm is applied to a subset $G^{(i)}$ a receiver vector $\bar{r}^{(i)}$ results with active vectors $I^{(i)}$. It is clear that if

$$\langle \bar{s}_j, \bar{r}^{(i)} \rangle \geq \langle \bar{s}_k, \bar{r}^{(i)} \rangle \quad \text{for all } j \notin I^{(i)} \text{ and } k \in I^{(i)}, \quad (3.2.2)$$

then $\bar{r}^{(i)}$ must be the AOF vector for the entire signal set. This fact is an obvious result of the fundamental property of the AOF stated in (2.3.1), however since this provides the optimality test for each intermediate solution it is stated as a formal Theorem, the proof of which is trivial.

Theorem 4. (The Second Optimality Criterion:) *Let G be a set of vectors that is a subset of a larger set S^+ , and let $\bar{r}$ be the AOF vector of G . If $\{\bar{s}_i\}_{i \in I} \subset G$ are the active vectors of $\bar{r}$, then $\bar{r}$ is the AOF vector of the set S^+ if and only if*

$$\langle \bar{s}_j, \bar{r} \rangle > d_{min}(\bar{r}) \quad \text{for all } \{\bar{s}_j | \bar{s}_j \in S^+ \cap \bar{s}_j \notin G\}, \quad (3.2.3)$$

where $d_{min}(\bar{r}) = \langle \bar{s}_i, \bar{r} \rangle$ for $i \in I$.

In most cases $\bar{r}^{(1)}$ will not be the AOF vector. This results when some or all of the active vectors of $\bar{r}_x$ are discarded in forming the set $G^{(1)}$. If $\bar{r}^{(1)}$ is not the AOF

vector, then at least one vector $\{\bar{s}_k | k \notin I^{(1)}, k \in I_x\}$ will cause the optimality test for $\bar{r}^{(1)}$ to fail. As a result, any vector that does not satisfy (3.2.3) is a good candidate for an active vector of $\bar{r}_x$. Therefore, the second subset $G^{(2)}$ is formed by the set $I^{(1)}$, plus all such vectors $\bar{s}_k$ up to the maximum allowable number of signal vectors n_{max} . If the number of signal vectors in $G^{(2)}$ is less than n_{max} , then $G^{(2)}$ will almost always contain more members of I_x than $G^{(1)}$. (See Tables 3.1-3.6). If $G^{(2)}$ contains n_{max} signal vectors, then $G^{(2)}$ contains at least the same number of members of I_x as $G^{(1)}$, but a different vector $r^{(2)}$ will result with a different active vector set $I^{(2)}$. By continuing the procedure, eventually a subset is found containing the entire set I_x , so that the resulting vector satisfies the second optimality criterion and is in fact the AOF vector.

In any application where the extended version of the AOLFP algorithm must be used, the choice of n_{max} is more important than the choice of ϵ . The purpose of ϵ is mainly to obtain a subset of the signal set that is small enough to be handled by the algorithm, and at the same time is representative of the signal set as a whole. It is impossible to know for a given signal set what a good choice for ϵ would be. A value for ϵ that is too large will result in a small signal set for $G^{(1)}$, and reduce the number of active vectors of the AOF vector in $G^{(1)}$. In this case, more iterations will likely be required to find a subset $\{G^{(i)} | I_x \subset G^{(i)}\}$, and the size of $G^{(2)}$ is often greater than the size of $G^{(1)}$. Conversely, setting a small value for ϵ produces a better approximation to the signal set and hence a better starting point for the algorithm, but unfortunately this generally requires that $G^{(1)}$ contain more than n_{max} signal vectors, defeating the purpose of using small subsets. Restricting the size of each subset to n_{max} compensates for poor choices for ϵ , allowing the algorithm to proceed within the limits of the available memory space.

Tradeoffs exist for the choice of n_{max} as well. Making n_{max} as large as possible will definitely result in fewer iterations, however the overall process of finding the AOF

vector is likely to be slower, as larger inner product matrices must be calculated and stored for each iteration, and each application of the AOLFP algorithm must search through more signal vectors. A smaller value of n_{max} will result in more iterations, however since the AOLFP algorithm itself runs very quickly, a small value for n_{max} may result in a quicker convergence to the AOF vector because the inner product matrices are smaller for each iteration, requiring less time for calculation and storing, and each application of the AOLFP algorithm has a smaller signal set to deal with. It turns out that many of these valid considerations are merely academic, as the extended algorithm typically converges quickly for various combinations of n_{max} and ϵ in practice, some typical cases being given in Tables 3.1 to 3.6. The only requirement is that n_{max} be greater than the number of active vectors of the AOF vector, otherwise no subset $G^{(i)}$ will ever contain the entire set I_x , and the algorithm will never converge. Unfortunately, the number of vectors in I_x cannot be predicted, however during the course of this thesis I_x was rarely found to contain more than 100 signal vectors.

The extended AOLFP algorithm discussed in this Section can be seen to be similar to the crude method of finding the AOF vector described in Section 2.3. In the crude approach, subsets of vectors are chosen at random until the set I_x is found. In the extended AOLFP algorithm, intelligent guesses are made for vectors that are likely to be active vectors. This approach, combined with the AOLFP algorithm, allows the AOF vector to be found for any size signal set, given sufficient computing time.

3.2.2 Some Typical Examples

In this Section, details of the convergence of the extended AOLFP algorithm are considered for a few of the practical cases to be presented later. The results in Tables 3.1 to 3.3 correspond to the solution for a duobinary MSK signal corrupted by an adjacent channel interference signal with spectral separation given by ΔfT , observed over 3 bit intervals. The number of data signatures for this case is 2816. Direct application of

the AOLFP algorithm would require more than 31 Mb of memory to store the inner product matrix (using double precision), a rather extreme requirement. In addition to the memory space, excessive computing time would also be required to calculate the 3,996,336 inner products that comprise the upper triangular portion of the inner product matrix. The extended AOLFP algorithm has been used to solve this problem for three cases of ϵ and n_{max} . In each Table, n represents the number of vectors in each subset $G^{(i)}$ and m represents the number of active vectors of the AOF (members of I_x) included in $I^{(i)}$ for i the iteration index. From the results in Table 3.1 for $\epsilon = .15$ and $n_{max} = 1000$, the largest intermediate signal set is one of 936 signal vectors for $\Delta fT = .862$, requiring roughly 3.4 Mb of memory for the inner product matrix. Note that the number of members of I_x increases with each iteration. It appears that $\epsilon = .15$ is a good choice, as the initial subset in each case contains two members of I_x , and successive subsets rapidly decrease in size allowing the AOLFP algorithm to run more quickly with each iteration.

iteration index	$\Delta fT = .554$		$\Delta fT = .654$		$\Delta fT = .862$	
	n	m	n	m	n	m
1	780	2	802	2	936	2
2	11	4	11	4	14	5
3	6	5	8	6	9	6
4	6	6				
5	7	7				

Table 3.1 Parameters of intermediate signal sets for $\epsilon = .15$ and $n_{max}=1000$.

In Table 3.2, similar results are given for $n_{max} = 500$. Here, less than 1 Mb is required to store the inner product matrix, but the initial subsets $G^{(1)}$ in each case are no longer representative of the signal set, because the algorithm stopped looking at

iteration index	$\Delta fT = .554$		$\Delta fT = .654$		$\Delta fT = .862$	
	n	m	n	m	n	m
1	500	1	500	0	500	1
2	12	6	31	2	34	5
3	7	7	15	2	7	6
4			7	3		
5			8	6		

Table 3.2 Parameters of intermediate signal sets for $\epsilon = .15$ and $n_{max} = 500$.

difference vectors once 500 signal vectors had accumulated. From the results however, it can be seen that this does not result in an increase in the number of iterations, except when $G^{(1)}$ contains no members of I_x as for $\Delta fT = .654$. In fact, two fewer iterations are required for $\Delta fT = .554$ than previously. Most importantly though, the computing time for each case was reduced by approximately one third due to the smaller size of the initial subsets. Since each $G^{(1)}$ in Table 3.2 contains n_{max} vectors, it is impossible to increase the number of members of I_x in $G^{(1)}$ (and thereby reduce the number of iterations) by decreasing ϵ . Therefore gaining the benefit of a small value of ϵ in general requires a large value of n_{max} so that all signal vectors are considered in finding $G^{(1)}$. Unfortunately, this increases the memory space required to store the inner product matrix for each subset, and increases the time required for each iteration.

The problem was solved again for $\epsilon = 1.0$ and $n_{max} = 500$. The results are given in Table 3.3 and indicate that the large value of ϵ has little effect on both the number of iterations in each case and how many members of I_x are contained in $G^{(1)}$. The most important result here is that the computing time required for these cases was roughly one tenth of the time required for the cases in Table 3.1. This results from the fact that the large value of ϵ produced a small subset $G^{(1)}$ representative of the entire signal set, and the subsequent subsets each contained successively fewer signal vectors. Note also

from Table 3.3 that despite the fact that the signal set involves 2816 signal vectors, the extended algorithm is only concerned with a maximum of 89 signal vectors at any one time.

iteration index	$\Delta fT = .554$		$\Delta fT = .654$		$\Delta fT = .862$	
	n	m	n	m	n	m
1	76	1	85	1	89	2
2	42	4	52	6	60	6
3	15	5				
4	6	6				
5	7	7				

Table 3.3 Parameters of intermediate signal sets for $\epsilon = 1.00$ and $n_{max} = 500$.

From the results of Table 3.1 to 3.3, it appears that the best approach is to set ϵ to a large value $\epsilon \approx 1$, and select a value for n_{max} based primarily on available memory space, but also on available computing time. Note that since the value of $\langle \bar{s}_i, \bar{r}_x \rangle$ for any $\bar{s}_i$ depends on the maximum value of $s_i(t)$ and the symbol time T , an appropriate value for ϵ depends on these values. The value of $\epsilon = 1$ stated here refers to inner products calculated for $T = 1$ and the components comprising each data signature (see Section 5.3) scaled to have a maximum value of 1.

This approach was applied to the same signal as the above cases, but for observation intervals lasting 4 and 5 bit intervals. These cases involve 45,056 data signatures, and the inner product matrix contains over one billion entries. This would require almost 8 Gb to store and could not even be attempted without the extended version of the AOLFP algorithm. The solutions were found with $\epsilon = 1.05$ and $\epsilon = .95$, and are given in Table 3.4 and Table 3.5 respectively. In each case $n_{max} = 1000$ so that a maximum of 4 Mb are required to store an inner product matrix.

iteration index	$\Delta fT = .554$		$\Delta fT = .654$		$\Delta fT = .862$		$\Delta fT = 3.620$	
	n	m	n	m	n	m	n	m
1	336	0	357	1	363	1	350	0
2	346	20	204	12	722	29	1000	3
3	108	24					944	62
4	32	28						

Table 3.4 Parameters of intermediate signal sets for $\epsilon = 1.05$ and $n_{max} = 1000$.

iteration index	$\Delta fT = .554$		$\Delta fT = .654$		$\Delta fT = .862$	
	n	m	n	m	n	m
1	906	0	890	2	868	1
2	280	22	464	30	301	41

Table 3.5 Parameters of intermediate signal sets for $\epsilon = .95$ and $n_{max} = 1000$.

In Table 3.4, even though $\epsilon = 1.05$ is sufficient to ensure for each case that all signal vectors are considered when finding $G^{(1)}$, $G^{(2)}$ contains more signal vectors than $G^{(1)}$ in three of the four cases. In contrast, the similar value for ϵ used in Table 3.5 results in successively smaller subsets and few iterations. It therefore appears to be impossible to predict how the solution will proceed for a given value of ϵ , and n_{max} must be used as a “safety feature” to limit the size of each intermediate subset, as seen by the second iteration for $\Delta fT = 3.62$ in Table 3.4.

From the typical cases presented it appears that when applying the extended AOLFP algorithm it is best to use a value for ϵ large enough that all signal vectors are considered in finding $G^{(1)}$, that is, so that $G^{(1)}$ contains fewer than n_{max} signal vectors. This ensures that $G^{(1)}$ is representative of the entire signal set. It also appears that n_{max} should be made small so as to reduce the memory space and computing

time requirements, however it must be greater than the number of active vectors of the AOF vector. In each case presented here, this number was less than 100. When computing on a mainframe computer, the choice of ϵ and n_{max} are less important as there is generally sufficient memory available for $n_{max} \geq 500$, and computing time is not generally excessive even when several iterations are required due to a poor choice of ϵ . When computing on a personal computer however, some trial and error may be necessary before the AOF vector can be found. As a general rule of thumb, $\epsilon \approx 1$ and $n_{max} \geq 100$ can be used.

CHAPTER 4

APPLYING THE AOLFP ALGORITHM TO DUOBINARY MSK

In this and the remaining chapters, the AOLFP algorithm is applied to examples that demonstrate its use as a method of filter design for practical signalling problems. The examples in Chapter 5 deal with MSK-type detection of duobinary MSK corrupted by AWGN and a specific source of distortion that increases the number of data signatures at the receiver. This chapter introduces the duobinary MSK signal and explains why MSK-type detection is possible and useful. The AOLFP algorithm is applied to the “nominal” (undistorted) duobinary MSK data signatures over three different observation intervals and the error performance of each nominal AOF is calculated for comparison with the results for distorted duobinary MSK in Chapter 5.

4.1 Duobinary MSK and MSK-Type Detection

Duobinary MSK is a specific instance of the class of signals referred to as partial response continuous-phase modulated (CPM) signals [2]. This class of signals has received a great amount of attention in recent years, mainly due to the properties of constant envelope, efficient power spectrum, and good error performance. These signals can be expressed as in [2], or alternatively as given in [1] as

$$s(t) = A_c \cos(\omega_c t + \phi(t) + \phi_0), \quad (4.1.1)$$

where $\phi(t)$ is the “excess-phase” given by

$$\phi(t) = \frac{\pi h}{T} \sum_n \alpha_n \int_0^t g(\tau - nT) d\tau, \quad (4.1.2)$$

ω_c is the angular carrier frequency, ϕ_0 is the initial carrier phase, h is the modulation index, and T is the symbol time. The digital information is transmitted by the

uncorrelated M -ary data symbols $\{\alpha_i\}$, occurring every T seconds and taking values $\pm 1, \pm 3, \dots \pm (M-1)$. The pulse $g(t)$ is nonzero only on the interval $t \in [0, aT]$, and determines the instantaneous frequency of the transmitted signal $s(t)$. Typically a is an integer. The slope of $g(t)$ determines how the data symbols modulate the excess-phase, and therefore has a direct impact on the power spectrum of $s(t)$. By varying the shape and duration of $g(t)$, as well as the parameters h and M , many modulated signals are possible, each with different spectral characteristics. For example, $a = 1$, $M = 2$, $h = \frac{1}{2}$ and the pulse

$$g_{MSK}(t) = \begin{cases} 1, & t \in [0, T], \\ 0, & \text{otherwise,} \end{cases} \quad (4.1.3)$$

produce the signal commonly known as minimum shift keying (MSK). Signals such as these for which $a = 1$ are referred to as “full response” CPM [2] and result in each data symbol contributing $\pm\pi h$ to the excess-phase over a single symbol interval. From equation (4.1.2) the excess-phase for MSK at any time $t = mT$ is

$$\phi_{MSK}(mT) = \frac{\pi}{2} \sum_{n=0}^{(m-1)} \alpha_n \quad (4.1.4)$$

where m may be negative and the notation $\sum'$ is defined as

$$\sum_{n=i}^j = \begin{cases} \sum_{n=i}^j, & j \geq i, \\ 0, & j = i - 1, \\ -\sum_{n=j+1}^{i-1}, & j \leq i - 2. \end{cases} \quad (4.1.5)$$

As discussed in [1][8], MSK is equivalent to OQPSK so that this nonlinearly modulated signal can be optimally demodulated by the IQ receiver in Fig. 4.1, with a predetection filter having impulse response

$$h_f(t) = \begin{cases} \cos \frac{\pi}{2T}(t - T), & t \in [0, 2T], \\ 0, & \text{otherwise.} \end{cases} \quad (4.1.6)$$

When MSK is processed by the IQ receiver with perfect carrier recovery, the signal $\frac{A_s}{2} \cos \phi_{MSK}(t)$ appears at the input to the predetection filter in the upper arm, and

Fig. 4.1 IQ receiver.

$\frac{A_c}{2} \sin \phi_{MSK}(t)$ appears in the lower arm. The polarity of the samples of the filtered versions of these signals determines the received data symbols $\{\pm \alpha_i\}$.

Since each arm of the IQ receiver is in fact a linear receiver of the type shown in Fig. 2.1, the AOF for each arm of the IQ receiver can be found by applying the AOLFP algorithm to the signals appearing at the input to the predetection filters. For MSK, the AOF for each arm is identical to $h_f(t)$ in (4.1.6).

It is well-known [9] that by extending the duration a of the pulse $g(t)$, while preserving the property that

$$\int_{-\infty}^{\infty} g(t) dt = T, \quad (4.1.7)$$

the spectral characteristics of $s(t)$ can be greatly improved. The resulting signals are referred to as "partial response" CPM and each data symbol contributes $\pm \pi h$ rad to the excess-phase over a symbol intervals. Duobinary MSK occurs for $a = 2$, $M = 2$, $h = \frac{1}{2}$ and the pulse

$$g(t) = \begin{cases} \frac{1}{2}, & t \in [0, 2T], \\ 0, & \text{otherwise.} \end{cases} \quad (4.1.8)$$

At any time $t = mT$, the excess-phase of duobinary MSK from (4.1.2) can be expressed

as

$$\begin{aligned}\phi(mT) &= \frac{\pi}{2} \sum_{n=0}^{m-1} \alpha_n - \frac{\pi}{4} \alpha_{m-1} + \frac{\pi}{4} \alpha_{-1} \\ &= \phi_{MSK}(mT) - \frac{\pi}{4} \alpha_{m-1} + \frac{\pi}{4} \alpha_{-1}.\end{aligned}\tag{4.1.9}$$

As discussed in [1][2], partial response CPM signals such as duobinary MSK can not be optimally demodulated by a linear receiver, and the truly optimal receiver for these signals is the highly complex and nonlinear Viterbi receiver. Fortunately, it is possible, under certain conditions, to suboptimally demodulate these signals with the linear IQ receiver. To see how this is possible for duobinary MSK, note that the last term in (4.1.9) is independent of the general time instant mT and hence can be incorporated into the carrier phase ϕ_0 . The transmitted signal $s(t)$ can then be viewed as

$$s(t) = A_c \cos(\omega_c t + \hat{\phi}(t) + \hat{\phi}_0)\tag{4.1.10}$$

where

$$\hat{\phi}(mT) = \phi_{MSK}(mT) - \frac{\pi}{4} \alpha_{m-1}\tag{4.1.11}$$

and

$$\hat{\phi}_0 = \phi_0 + \frac{\pi}{4} \alpha_{-1}.\tag{4.1.12}$$

Since all data symbols are either +1 or -1,

$$|\hat{\phi}(mT) - \phi_{MSK}(mT)| = \frac{\pi}{4}.\tag{4.1.13}$$

Thus $\cos \hat{\phi}(t)$ and $\cos \phi_{MSK}(t)$ are simply offset by $\frac{\pi}{4}$ and the two signals have the same polarity at the sampling instants. (Similarly for $\sin \hat{\phi}(t)$ and $\sin \phi_{MSK}(t)$). As a result, data transmitted by a duobinary MSK signal can also be recovered by the linear IQ receiver, provided ϕ_0 in Fig. 4.1 is replaced by $\hat{\phi}_0$. This will result in $\frac{A_c}{2} \cos \hat{\phi}(t)$ and $\frac{A_c}{2} \sin \hat{\phi}(t)$ appearing at the input to the predetection filters so that the polarity of the output samples (which determine the data symbols) is unchanged compared to MSK. Thus, linear detection of the spectrally superior duobinary MSK signal is also

possible (although clearly with degraded performance). This type of linear detection of nonlinear modulation signals is possible for many other partial response CPM signals with modulation index $h = \frac{1}{2}$ and is commonly called "MSK-type detection". A general criterion for when MSK-type detection is possible for a given partial response CPM signal can be found in [1].

It is clear that MSK-type detection is suboptimal for any partial response CPM signal, however, the question arises as to the choice of predetection filter that will provide the best suboptimal error performance. Since each arm of the IQ receiver is comprised of a linear filter, sampler, and level slicer; and partial response CPM signals possess memory, the problem of predetection filter design for MSK-type detection of partial response CPM signals readily lends itself to the AOLFP algorithm. Therefore by applying the algorithm to the signals that appear at the predetection filter input for various CPM schemes, the impulse response that maximizes the post filter eye opening can be found. The error performance of this filter will be the best attainable with the linear IQ receiver.

In the next Section, the AOF for duobinary MSK is determined for three observation intervals from the nominal data signatures, that is, for ideal transmission except for AWGN.

4.2 Nominal Filter Design

To find the impulse response of the nominal AOF, we assume that undistorted duobinary MSK affected only by AWGN is received, and ideal carrier recovery exists for coherent detection. In this case $\cos(\omega_c t + \hat{\phi}_0)$ and $\sin(\omega_c t + \hat{\phi}_0)$ are available at the receiver so that $\frac{A_c}{2} \cos \hat{\phi}(t)$ appears at the input to the predetection filter in the upper (I) arm of the IQ receiver, and $\frac{A_c}{2} \sin \hat{\phi}(t)$ appears at the filter input in the lower (Q) arm. (Note that due to the lowpass nature of the predetection filters, the high frequency components that also appear at the filter input have been neglected.)

Finding the AOF for each arm of the receiver requires the set of data signatures that result from these signals on some observation interval I_{obs} . These data signatures can be constructed from the eye diagram of each signal, which is derived from the excess-phase function $\hat{\phi}(t)$. Since $g(t)$ for duobinary MSK (equation (4.1.8)) is flat, the phase trajectory of $\hat{\phi}(t)$ on any bit interval $t \in [mT, mT+T]$ between the phase states $\hat{\phi}(mT)$ and $\hat{\phi}(mT+T)$ is linear. This fact coupled with equation (4.1.11) results in the phase tree for duobinary MSK shown in Fig. 4.2.

For simplicity, the phase tree shown corresponds to:

$$\alpha_n = 0 \quad \text{for } n < -1, \quad \phi_{MSK}(0) = 0, \quad \text{and } \alpha_{-1} = -1. \quad (4.2.1)$$

That is, the time origin is chosen as some instant $t = kT$ such that $\phi_{MSK}(kT) = 0$, with a -1 symbol occurring on the interval $[kT-T, kT]$. After five bit intervals, some phase states begin to repeat modulo 2π , and the phase tree provides enough information to determine the eye diagram for $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t)$. The eye diagram for $\cos \hat{\phi}(t)$ is given in Fig. 4.3 over a generic $3T$ interval. This eye is found by plotting $\cos \hat{\phi}(t)$ for all possible paths through the phase tree over any interval $t \in [nT, nT+3T]$ centered at the maximum eye opening. It is easy to see from Fig. 4.2 that a phase shift in $\hat{\phi}(t)$ of $\pm \frac{\pi}{2}$ translates into a time shift of $\pm T$ in the eye diagram. This means that the eye diagrams for $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t \pm T)$ are identical. Since the data signatures for each signal are derived from the eye diagram, these two signals have identical signal sets and identical AOF's. It is therefore sufficient to find the AOF for $\cos \hat{\phi}(t)$ only, and use two identical filters in the IQ receiver.

At the transmitter, the $\frac{1}{T}$ rate data signal is demultiplexed into two $\frac{1}{2T}$ rate bit streams, each modulating the signal in either the I or Q channel. The eye diagram has periodicity $2T$ and all possible realizations of $\cos \hat{\phi}(t)$, superimposed on each other, appear as a sequence of concatenated eye diagrams. Data symbols are transmitted on specific $2T$ intervals corresponding to an open eye, and the value of the data symbol

Fig. 4.2 Phase tree for duobinary MSK.

Fig. 4.3 Nominal eye diagram for duobinary MSK.

transmitted (± 1) determines the polarity at the center of the eye opening. The $2T$ periodicity of $\cos \hat{\phi}(t)$ implies that the detection problem on a specific $2T$ interval when the eye is open is identical to the detection problem on any other such $2T$ interval. Therefore, the calculation of the AOF can be based on a $+1$ symbol being transmitted on any specific $2T$ interval. Since in the nominal case this same signal (multiplied by $\frac{A_c}{2}$) appears at the input to the predetection filter in the I channel, the received data signatures corresponding to a $+1$ symbol can be found directly from the phase tree in Fig. 4.2. However, as many of the paths through the phase tree result in the same waveform, it is easiest to determine the data signatures directly from the eye diagram for $\cos \hat{\phi}(t)$. For observation intervals of duration greater than $2T$, the data signatures can be found using the $2T$ periodicity of the eye diagram.

In the analysis, the time origin was chosen as the beginning of some $2T$ interval on which a $+1$ symbol was transmitted. The data signatures that exist for the $+1$ symbol are shown superimposed in Fig. 4.4 over a $3T$ duration. Note that the time scale in Fig. 4.4 is arbitrary and is not the same as the time scale in Fig. 4.2. Since the AOLFP algorithm operates only on inner products, the time scale used to calculate the inner products is irrelevant and is chosen simply for convenience.

Fig. 4.4 Superimposed data signatures for nominal duobinary MSK.

Distinct paths through the phase tree are reflected by distinct pieces of the waveform in Fig. 4.4 on each T interval. Therefore, any waveform in Fig. 4.4 that is continuous except at $t = mT$ (for m an integer) is a data signature transmitting the $+1$ symbol and is used to find the nominal AOF. Each data signature can be viewed as being comprised of a sum of pulses on successive T intervals, each pulse being a time shifted and possibly inverted replica of one of the three basic pulses $p_1(t)$, $p_2(t)$, and $p_3(t)$ shown in Fig. 4.5.

With these pulse definitions, the behaviour of the signal can be fully described by four properties:

- (1) a $\pm p_1(t)$ pulse can be followed by a $\pm p_2(t)$ or $\pm p_3(t)$ pulse.
- (2) a $\pm p_2(t)$ pulse can be followed by a $\pm p_1(t)$, $\pm p_2(t)$ or $\pm p_3(t)$ pulse.
- (3) a $\pm p_3(t)$ pulse can be followed by a $\mp p_1(t)$ or $\mp p_2(t)$ pulse.
- (4) if $\pm p_1(t)$ ($\pm p_3(t)$) occurs on a particular T interval in a particular data signature then $\pm p_3(t)$ ($\pm p_1(t)$) cannot occur on the same interval in any other data signature.

Since the possibilities for the pulse occurring on a particular interval depend only on the pulse occurring on the previous interval, this signal can be described by a Markov chain and can be easily simulated in software. Each data signature can be generated

Fig. 4.5 Basic pulses comprising the nominal data signatures.

as a sequence of integers from the set $\{\pm 1, \pm 2, \pm 3\}$, indicating the sequence of pulses that form the waveform on the observation interval. The number of pulses depends on the number of T intervals, in whole or in part, contained in the observation interval. Each integer indicates the pulse and polarity that describe the data signature on one T interval. This type of approach can be applied to any CPM signal, and was exactly the technique used in [5]. The number and shape of the pulses $\{p_i(t)\}$ that can occur for a specific CPM signal varies with the choice of $g(t)$ in (4.1.2).

To calculate the inner products, an observation interval of arbitrary length L was symmetrically chosen about the center of the eye opening at $t = \frac{3}{2}T$. This was the most obvious choice for the observation interval as this is where the signal contains the information about the +1 data symbol. Therefore, it will enable the calculation of the best possible error performance. With this orientation of the observation interval, the data signatures for $L = 3T$ are exactly those superimposed in Fig. 4.4.

A point of clarification is necessary with regards to the impulse response of the AOF. Since the correlating waveform is expressed as a linear combination of specific data signatures, it will be described on the chosen observation interval. The impulse response however is always derived from $r_x(t)$ by reflecting it about the right end point of the observation interval and time shifting it so as to be nonzero only for $t \in [0, L]$. As a result, the arbitrary interval $t \in [0, 2T]$ chosen to transmit a +1 symbol and the arbitrary selection of the observation interval will have no effect on the causality of the AOF.

Since any data signature for $L = 3T$ is comprised of a sequence of three basic pulses, the easiest method of finding the inner product between any two data signatures is to first calculate the inner products amongst the three basic pulses that can occur on the same T interval. Then from the sequence of pulses that comprise the data signatures on each T interval, add the appropriate inner products on successive T intervals to form the entire inner product. In this way, extending the observation interval merely requires more additions of known values. Stated explicitly, each data signature $s_1(t)$ can be expressed as:

$$s_1(t) = \sum_{i=1-L/2}^{1+L/2} a_{1,i} p_{1,i}(t - iT) \quad (4.2.2)$$

where L is the duration of the observation interval, $p_{1,i}$ is the basic pulse describing $s_1(t)$ on the i^{th} T interval $t \in [iT, iT + T]$, $a_{1,i} \in \{\pm 1\}$ defines the polarity of the pulse, and $L/2$ refers to integer division. The inner product between $\bar{s}_1$ and $\bar{s}_2$ can therefore be expressed as:

$$\langle \bar{s}_1, \bar{s}_2 \rangle = \sum_{i=1-L/2}^{1+L/2} a_{1,i} a_{2,i} \langle \bar{p}_{1,i}, \bar{p}_{2,i} \rangle. \quad (4.2.3)$$

Note that when the duration of the observation interval L is even, the $(1 - L/2)^{th}$ and $(1 + L/2)^{th}$ intervals involve integrations over $\frac{1}{2}T$ intervals. For the basic pulses in

Fig. 4.5 the relevant inner products for L an odd integer are:

$$\begin{aligned}\langle \bar{p}_1, \bar{p}_1 \rangle &= \left(\frac{1}{2} + \frac{1}{\pi} \right) T & \langle \bar{p}_1, \bar{p}_2 \rangle &= \frac{2}{\pi} T & \langle \bar{p}_2, \bar{p}_2 \rangle &= \frac{1}{2} T \\ \langle \bar{p}_2, \bar{p}_3 \rangle &= 0 & \langle \bar{p}_3, \bar{p}_3 \rangle &= \left(\frac{1}{2} - \frac{1}{\pi} \right) T,\end{aligned}\tag{4.2.4a}$$

and for L even are:

$$\begin{aligned}\langle \bar{p}_1, \bar{p}_1 \rangle &= \frac{1}{2} \left(\frac{1}{2} + \frac{1}{\pi} \right) T & \langle \bar{p}_1, \bar{p}_2 \rangle &= \frac{1}{\pi} T \\ \langle \bar{p}_2, \bar{p}_2 \rangle &= \frac{1}{4} T & \langle \bar{p}_3, \bar{p}_3 \rangle &= \frac{1}{2} \left(\frac{1}{2} - \frac{1}{\pi} \right) T \\ \langle \bar{p}_2, \bar{p}_3 \rangle &= \left(\frac{1 - \sqrt{2}}{\pi} \right) T & \text{for } t \in \left[\left(\frac{3}{2} - \frac{L}{2} \right) T, \left(2 - \frac{L}{2} \right) T \right] \\ \langle \bar{p}_2, \bar{p}_3 \rangle &= \left(\frac{\sqrt{2} - 1}{\pi} \right) T & \text{for } t \in \left[\left(1 + \frac{L}{2} \right) T, \left(\frac{3}{2} + \frac{L}{2} \right) T \right].\end{aligned}\tag{4.2.4b}$$

Consider the observation interval for $L = 3T$ specified by $t \in [0, 3T]$. In this case, eight data signatures exist that satisfy the requirement of positive polarity for $t \in [0, 2T]$ and are shown in Fig. 4.6. These data signatures form the signal set S^+ for nominal duobinary MSK on this observation interval.

After calculating the inner product matrix for these data signatures from the component inner products in (4.2.4a), the nominal AOF for $L = 3T$ was calculated. The correlating waveform of the AOF is shown in Fig. 4.7, and the corresponding error curve is given in Fig. 4.8. These curves were originally given in [1]. The two cases of $L = 4T$ and $L = 5T$ each involve 32 data signatures similar to those in Fig. 4.6, but with longer duration. The AOF's and error curves for these cases are also shown in Fig. 4.7 and Fig. 4.8. For comparison, the error curve for MSK is included as well.

The filters can be compared on the basis of their asymptotic degradation from the MSK curve, given by $-20 \log(d_{\min}(\bar{r}_x))$ dB. The values are 1.15471 dB, .99923 dB, and .961120 dB for the AOF on $L = 3T$, $L = 4T$, and $L = 5T$ respectively. Notice that the differential in the asymptotic degradation for $L = 3T$ and $L = 4T$ is larger than that of $L = 4T$ and $L = 5T$. This differential continues to diminish as L increases so

Fig. 4.6 Data signatures for nominal duobinary MSK.

Fig. 4.7 Correlating waveforms of the nominal filters for $L = 3T$, $L = 4T$, and $L = 5T$.

Fig. 4.8 BER of the nominal filters for $L = 3T$, $L = 4T$, and $L = 5T$.

that there is a point after which increasing L does not improve the error performance appreciably. This is typical of signals involving memory.

Nominally speaking then, there is roughly a 1 dB penalty in SNR for duobinary MSK as compared to MSK, which is of course the tradeoff for the spectrally superior duobinary signal. Note that for the $L = 3T$ case, the sampling point is at $t = 3T$, making the impulse response of the nominal AOF from equation (2.1.2) identical to the correlating waveform in Fig. 4.7. To obtain the impulse response of the AOF for $L = 4T$ and $L = 5T$, the correlating waveforms must be shifted so as to be nonzero for $t \in [0, 4T]$ and $t \in [0, 5T]$ respectively.

The error curves presented in Fig. 4.8 give the best possible performance for MSK-type detection of duobinary MSK. In practice, this performance could never be achieved due to the many sources of distortion that inevitably occur during transmission. The remainder of this thesis considers the presence of three such sources of distortion, and determines their effect on the detection of the +1 symbol. In each case, a new AOF is determined by taking into consideration the effect of the distortion. The performance of the new AOF is calculated as well as the performance that would be achieved by the nominal AOF's in Fig. 4.7. These are then compared with the "ideal" performance in Fig. 4.8. To make clear the distinction between the two filters, the nominal AOF's in Fig. 4.7 will be referred to as *nominal filters* in the remainder of this thesis, and the new AOF will be referred to simply as the AOF.

CHAPTER 5

PRACTICAL APPLICATIONS OF THE AOLFP ALGORITHM

In this chapter the ability of the AOF to compensate for known signal perturbations is investigated. Three different cases are studied in which the signal model includes a specific source of distortion common in practical communication systems. The distortion significantly increases the number of signals in each signal set and affects the values in the inner product matrix. Due to the size of the signal sets, each solution requires that the method discussed in Chapter 3 be employed. In addition, the size of the signal sets requires that the process of calculating the inner product matrix in each case be automated. As this is the case in any nontrivial application of the algorithm, the same basic approach is taken in each signalling problem and is discussed briefly in Section 5.1.

The specific problems considered deal independently with phase errors in carrier recovery, adjacent channel interference from a second duobinary MSK signal, and intersymbol interference resulting from bandlimiting at the transmitter. Since the process for finding the AOF is the same for both the I-channel and the Q-channel, only the design problem for the I-channel is considered in each case.

5.1 A General Approach to Calculating the Inner Product Matrix

In any application of the AOLFP algorithm, a signal model of the received signal is required from which the received data signatures can be determined and the inner product matrix calculated. In the nominal case, the signal model was quite simple as it was assumed that an undistorted signal was received. The number of parameters needed to describe a data signature was equal to the number of symbol intervals in the observation interval. When the signal model includes interference effects, information

about the interference must be added, increasing the number of parameters needed to fully describe a data signature. From the allowable values of each parameter, the size of the signal set can be determined. In software, a data structure must be defined whose members are these parameters, for the purpose of storing information that describes a data signature. In addition, each data signature must be assigned a distinct index, and some means (subroutine) for generating the appropriate set of parameters given a specific index must be provided.

In addition to describing the data signatures, it is also necessary to calculate the inner product between any two data signatures given the parameters that describe them. Typically, the inner products are calculated numerically and there are two approaches to finding the inner product matrix. The simplest way is to directly calculate each element of the inner product matrix by evaluating numerical integrals over the entire observation interval for every pair of data signatures. Although this method is easily achieved in software it is inefficient. While no two data signatures have the same description over the entire observation interval, many have identical descriptions on various symbol intervals within the observation interval. Therefore, in the course of numerically calculating every inner product, many integrations are repeated. When the signal set contains a large number of signal vectors, this approach needlessly and severely increases the required computing time and is unacceptable in general.

A superior approach is to use the signal model to break up a general inner product into a summation of integrations done on successive symbol intervals. These integrations are referred to as *component inner products*. Since various data signatures have identical descriptions on various symbol intervals, each inner product can be found through some suitable combination of the component inner products. The required combination can be determined from the parameters describing the two data signatures involved. Therefore it is more efficient to numerically evaluate only the component inner products and develop rules that, from the description of any two data signatures,

combine the component inner products in an appropriate manner to form the overall inner product.

The component inner products are the minimum number of numerical integrations that must be evaluated to enable any inner product to be found. These integrations can be ascertained from the signal model, as can the rules needed to combine them into inner products. Although these two steps are conceptually quite simple, they often require a fair amount of ingenuity and time to accomplish in software for any reasonably complex signal model. Since practical applications typically involve very large signal sets, the need to keep the time spent computing the inner product matrix within reasonable limits means that this approach is essential.

To summarize, any application of the AOLFP algorithm requires five essential steps. They are:

- (1) Determine the parameters needed to describe each data signature.
- (2) (a) Assign an index to each data signature and,
(b) provide some means for generating the parameter values for a given index.
- (3) Calculate and store the component inner products.
- (4) Develop rules to calculate overall inner products from component inner products.
- (5) Calculate the inner product matrix.

This is one of the reasons duobinary MSK was repeatedly chosen as the transmitted signal in each case studied. The software used to generate the nominal data signatures was reused in step (2b) for each case involving interference.

5.2 Filter Design for Nonideal Carrier Recovery

The first signalling problem considered involves relaxing the previous restriction of ideal carrier recovery at the receiver. Recall that with ideal carrier recovery $\cos(\omega_c t + \hat{\phi}_0)$ is available at the receiver so that $\frac{A_c}{2} \cos \hat{\phi}(t)$ is the input to the predetection filter in the I channel. It is now assumed that some phase error θ results from the carrier recovery process so that the waveform used for carrier recovery is $\cos(\omega_c t + \hat{\phi}_0 + \theta)$. The objective is to determine whether an AOF that outperforms the nominal filter can be found by including the effect of θ in the received data signatures.

5.2.1 Signal Model and Calculation of the Inner Products

It is assumed that the transmitted signal

$$A_c \cos(\omega_c t + \hat{\phi}(t) + \phi_0) \quad (5.2.1)$$

reaches the receiver, accompanied by $n(t)$ an AWGN process with two-sided power spectral density $\frac{N_0}{2}$. The distortion occurs due to the phase error in the carrier recovery, resulting in the signal

$$\frac{A_c}{2} (\cos \theta \cos \hat{\phi}(t) + \sin \theta \sin \hat{\phi}(t)) = \frac{A_c}{2} r(t) \quad (5.2.2)$$

appearing at the input to the predetection filter, plus some higher frequency signals and noise. As in the nominal filter case in Chapter 4, the AOF's are low pass type filters which essentially filter away the high frequency components, and they need not be considered in finding the AOF. Hence, the AOF is found from the data signatures of $r(t)$ in (5.2.2). (Since the correlating waveform describing the AOF is normalized, the scale factor $\frac{A_c}{2}$ can be neglected provided it is included in calculating the error performance). Note that for $\theta = 0$ (perfect synchronization), the orthogonality of the I and Q channel signals $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t)$ is preserved, and the AOF is the nominal filter from Chapter 4 since $r(t) = \cos \hat{\phi}(t)$. However, when $\theta \neq 0$ cross coupling occurs

between the channels, destroying the orthogonality of the signals, and the AOF is no longer identical to the nominal filter.

The effect of the phase error θ on the pre-filter eye diagram as compared to the nominal eye diagram is illustrated in Fig. 5.1 for $\theta = \pm 10^\circ$, Fig. 5.2 for $\theta = \pm 20^\circ$ and Fig. 5.3 for $\theta = \pm 45^\circ$. These plots indicate that both the eye opening and the eye width decrease as $|\theta|$ increases. Note that the two nominal data signatures that are DC waveforms at $\pm \frac{1}{\sqrt{2}}$ are seen to split into two DC waveforms above and below the nominal level. As $|\theta|$ increases, the two smaller magnitude waveforms approach each other, reaching a level of zero when $|\theta| = 45^\circ$. At this point the eye closes and the transmitted data cannot be recovered.

Recall from Chapter 4 that the time axis used to determine the inner products is unimportant. It is only required that the data signatures on a specific observation interval be valid on any similar observation interval. Then the detection problem for a +1 symbol transmitted on a particular $2T$ interval is identical for any other similar $2T$ interval on which a +1 symbol is transmitted. Therefore as in the nominal case it is assumed here that a +1 symbol is transmitted on the interval $t \in [0, 2T]$, and the AOF is found from the data signatures of $r(t)$ on observation intervals centered at the maximum eye opening.

To describe the data signatures on a generic observation interval requires some consideration of the phase-error angle θ . While it is theoretically possible for θ to take on any value from $[-\pi, \pi]$, the probability of large phase errors occurring is very small for any practical synchronizer. Most phase errors that occur will lie within certain limits so that θ can be viewed as a continuous random variable uniformly distributed over some smaller interval. Although θ is in fact a slowly varying function of time, it was assumed in this analysis that $\frac{d\theta}{dt} \ll \frac{1}{2T}$ so that θ can be treated as a constant on any observation interval, as was done in [10]. Unfortunately, the fact that θ may take on any constant value in a given range presents a severe problem from the point of view of the

Fig. 5.1 Distortion in the nominal data signatures due to a phase error of 10°

Fig. 5.2 Distortion in the nominal data signatures due to a phase error of 20.0°

Fig. 5.3 Distortion in the nominal data signatures due to a phase error of 45°

AOLFP algorithm because this translates into an infinite number of data signatures for $r(t)$. To handle this problem, θ was modelled as a discrete random variable uniformly distributed over the range $\theta \in [-\theta_{max}, \theta_{max}]$, taking on n_t values equally spaced over the entire range. The signal set was formed by repeating the data signatures that result for a specific value of θ a total of n_t times, each group identical to the next except for distinct values of θ . The AOF's and resulting error performance were calculated for steadily increasing values for n_t until there was no significant change in the results. In this way the signal set represents a generic observation interval by including many values of θ . Note that in calculating the error performance it was assumed that all values of θ were equally probable. In reality this is not the case since phase errors of $\theta \approx \pm\theta_{max}$ are less likely to occur than phase errors of $\theta \approx 0$. Since the former degrades the overall performance more than the latter, the error curves presented in this Section are for a worst case situation, and better performance would be expected in practice.

In addition to the parameter θ , the $\sin \hat{\phi}(t)$ term in (5.2.2) must be considered in determining each data signature. To see what happens to the Q-channel signal $\sin \hat{\phi}(t)$ when $\cos \hat{\phi}(t)$ changes, consider a T interval on which $\cos \hat{\phi}(t)$ is described by the pulse preceeding $p_1(t)$ in Fig. 4.5. This pulse corresponds to a linear phase change in $\hat{\phi}(t)$ either from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$ or from $\frac{\pi}{4}$ to $-\frac{\pi}{4}$. As a result, the signal $\sin \hat{\phi}(t)$ can be described by either $p_3(t)$ or $-p_3(t)$ depending on the pulse that occurred on the preceeding T interval. Similarly, if $\cos \hat{\phi}(t)$ is described by $p_2(t)$ on a particular T interval, corresponding to $\hat{\phi}(t)$ remaining constant at either $\frac{\pi}{4}$ or $-\frac{\pi}{4}$, then $\sin \hat{\phi}(t)$ can be described by $\pm p_2(t)$ on the same T interval. Thus in general, for each realization of $\cos \hat{\phi}(t)$ on an observation interval there are two possible realizations for $\sin \hat{\phi}(t)$ on the same observation interval. For a specific value of θ , the number of possible data signatures for $r(t)$ is therefore twice the number of data signatures for $\cos \hat{\phi}(t)$. If this number is n_s , then the total number of data signatures in the signal set is $2n_s n_t$. The parameters required to describe a single data signature are: a value for θ , a sequence

of pulses describing the $\cos \hat{\phi}(t)$ waveform, and a similar sequence of pulses describing the $\sin \hat{\phi}(t)$ waveform. Note that the $\cos \hat{\phi}(t)$ waveforms are the nominal waveforms in Chapter 4 and the $\sin \hat{\phi}(t)$ waveforms are found directly from the $\cos \hat{\phi}(t)$ waveforms in the manner described above. These same signal descriptions are then simply repeated in groups, with a different value of θ assigned to each group, to form the signal set.

The calculation of the inner products for the signal model just described is fairly straightforward and can be done using the method (and software routines) from the nominal case. Consider two data signatures

$$\begin{aligned} r_1(t) &= \cos \theta_1 \cos \hat{\phi}_1(t) + \sin \theta_1 \sin \hat{\phi}_1(t) \\ r_2(t) &= \cos \theta_2 \cos \hat{\phi}_1(t) + \sin \theta_2 \sin \hat{\phi}_2(t). \end{aligned} \tag{5.2.3}$$

The inner products that form the matrix are of the form

$$\begin{aligned} \langle r_1(t), r_2(t) \rangle &= \cos \theta_1 \cos \theta_2 \langle \cos \hat{\phi}_1(t), \cos \hat{\phi}_2(t) \rangle + \cos \theta_1 \sin \theta_2 \langle \cos \hat{\phi}_1(t), \sin \hat{\phi}_2(t) \rangle \\ &\quad + \sin \theta_1 \cos \theta_2 \langle \sin \hat{\phi}_1(t), \cos \hat{\phi}_2(t) \rangle + \sin \theta_1 \sin \theta_2 \langle \sin \hat{\phi}_1(t), \sin \hat{\phi}_2(t) \rangle \end{aligned} \tag{5.2.4}$$

where the notation $\langle r_1(t), r_2(t) \rangle$ is equivalent to $\langle \bar{r}_1, \bar{r}_2 \rangle$.

The four inner product terms in (5.2.4) indicate the component inner products needed to determine the overall inner product between two data signatures. Each of the four inner products are identical to the inner products required in the nominal case, with a minor adjustment to allow the pulses $p_1(t)$ and $p_3(t)$ to occur on the same interval (this is necessary to evaluate the $\langle \cos \hat{\phi}_1(t), \sin \hat{\phi}_2(t) \rangle$ and $\langle \sin \hat{\phi}_1(t), \cos \hat{\phi}_2(t) \rangle$ terms). Clearly then, since θ_1 and θ_2 are constants, $\langle r_1(t), r_2(t) \rangle$ can easily be calculated by finding the four component inner products in (5.2.4) first, then evaluating (5.2.4) directly.

5.2.2 Results

The observation interval was centered at $t = \frac{3}{2}T$ and the AOF was found for several values of θ_{max} over a duration of $L = 3T$. A useful measure for comparing the AOF and nominal filters for each value of θ_{max} is the asymptotic degradation of the bit error rate (BER) of each filter from the BER of MSK. The performance improvement provided by each AOF over its nominal counterpart is therefore given by the difference between the degradation of each filter from the MSK curve. These results are given in Table 5.1 and indicate that the performance improvement afforded by the AOF as compared to the nominal filter is negligible for $\theta_{max} \leq 10^\circ$. To see if the AOF can provide significant performance gains when severe distortion results from a large phase error, $\theta_{max} = 10^\circ, 20^\circ$ and 30° were considered.

θ_{max}	AOF (dB)	Nominal Filter (dB)
0°	1.15	1.15
1°	1.32	1.32
2.5°	1.57	1.59
5°	2.03	2.06
10°	3.07	3.15
20°	5.32	6.11
30°	10.18	11.09

Table. 5.1 Asymptotic degradation of the AOF and nominal filter from the BER of MSK for several values of θ_{max} .

To calculate the BER of each AOF, the bandpass AWGN process $n(t)$ at the receiver is expressed in terms of the quadrature component processes $n_c(t)$ and $n_s(t)$ as

$$n(t) = n_c(t) \cos(\omega_c t + \psi) + n_s(t) \sin(\omega_c t + \psi) \quad (5.2.5)$$

where $n_c(t)$ and $n_s(t)$ are independent baseband noise processes with power spectral density N_0 . Since the expression is valid for any value of ψ [11,pg.433] choosing $\psi = \hat{\phi}_0$ results in the baseband Gaussian noise process $n_1(t)$ at the input to the AOF given by:

$$n_1(t) = \frac{1}{2} \left(\cos \theta n_c(t) - \sin \theta n_s(t) \right). \quad (5.2.6)$$

Repeating the analysis in Section 2.2 with $n(t)$ and $r(t)$ replaced by $n_1(t)$ and $r_x(t)$ respectively, the decision variable of the AOF when the data signature $\frac{A_c}{2} r_i(t)$ is received is a Gaussian random variable with mean

$$\mu = \frac{A_c}{2} \langle \bar{r}_i, \bar{r}_x \rangle = \frac{A_c}{2} \int_{I_{ob}} r_i(t) r_x(t) dt \quad (5.2.7)$$

and variance

$$\sigma^2 = \frac{N_0}{4}. \quad (5.2.8)$$

Therefore the conditional probability of a bit error when $r_i(t)$ is received is

$$P_e = Q \left\{ \frac{\mu}{\sigma} \right\} = Q \left\{ \frac{A_c}{\sqrt{N_0}} \langle \bar{r}_i, \bar{r}_x \rangle \right\}. \quad (5.2.9)$$

Since each of the N data signatures are equally likely, and the received energy per bit is $E_b = \frac{A_c^2 T}{2}$, the bit error probability becomes

$$P_e = \frac{1}{N} \sum_{i=1}^N Q \left\{ \sqrt{\frac{2E_b}{N_0}} \gamma_i \right\}, \quad (5.2.10)$$

where

$$\gamma_i = \frac{\langle \bar{r}_i, \bar{r}_x \rangle}{\sqrt{T}} \quad (5.2.11)$$

is independent of T . With this definition, the asymptotic degradation from the BER of MSK in Table 5.1 is given by $-20 \log \gamma_{min}$, where $\gamma_{min} = \gamma_i$ for $\bar{r}_i$ any active vector. The correlating waveforms and resulting BER curves for $\theta_{max} = 10^\circ$, 20° and 30° are shown in Fig. 5.4 and Fig. 5.5. Note that the "ideal" curve in Fig. 5.5 is the BER curve for the nominal filter of Chapter 4, or equivalently, the AOF for $\theta_{max} = 0^\circ$.

Fig. 5.4 Correlating waveforms for the AOF's and nominal filter for $L = 3T$ and several values of $\theta_{\max}$.

Fig. 5.5 BER of the AOF's and nominal filter for $L = 3T$ and several values of θ_{max} .

From the BER curves for $\theta_{max} = 10^\circ$ and $\theta_{max} = 20^\circ$ the AOF and nominal filter performance are seen to be identical for all practical error rates, and show a significant degradation from the ideal performance. The AOF for $\theta_{max} = 20^\circ$ provides an asymptotic gain of only .29 dB over the nominal filter. For $\theta_{max} = 30^\circ$ the error curve shows a noticeable improvement over the nominal filter with the improvement in asymptotic performance increasing to .91 dB. Clearly though, carrier recovery with phase errors as high as 20° could never be tolerated. Since the AOF is the best filter that can be used in a linear receiver it can be concluded that the performance degradation that occurs due to nonideal carrier recovery is a result of the structure of the linear receiver. Therefore, simply optimizing the predetection filter impulse response, as has been done here, will not provide satisfactory protection from phase errors in carrier recovery. This explains the near equivalent performance of the AOF and nominal filter for reasonable values of θ_{max} .

Further justification for why the AOF does not provide significant performance gains over the nominal filter is the fact that all of the active vectors of the AOF vector have a parameter $\theta = \pm\theta_{max}$. All other signal vectors are completely disregarded by the algorithm. Therefore the AOF would only be expected to outperform the nominal filter when $\theta \approx \pm\theta_{max}$. Since the BER curves in Fig. 5.5 were calculated by averaging the conditional error probability of each data signature, the differences between the two filters in terms of error performance is masked. The true behaviour of the filters is shown in Fig. 5.6 which shows the error performance of the AOF calculated for $\theta_{max} = 20^\circ$ and the nominal filter, when θ is fixed at 0° , 10° and 20° . The nominal filter is seen to outperform the AOF when $\theta = 0^\circ$, while the AOF provides slightly superior performance when $\theta = 20^\circ$. For the intermediate value of $\theta = 10^\circ$ the filter's perform identically. These results are more clearly seen by the asymptotic degradation of each error curve from the BER of MSK, given in Table 5.2.

Since the performance gain provided by the AOF under the best circumstances (when $\theta = 20^\circ$) amounts to only .29 dB, there appears to be little motivation for

Fig. 5.6 BER of the AOF for $\theta_{\max} = 20^\circ$ and the nominal filter for fixed phase errors.

θ	AOF (dB)	Nominal Filter (dB)
0°	1.36	1.15
$\pm 10^\circ$	3.18	3.15
$\pm 20^\circ$	5.82	6.11

Table. 5.2 Asymptotic degradation of the AOF for $\theta_{max} = 20^\circ$ and the nominal filter from the BER of MSK for fixed phase errors.

including the effect of phase errors in the calculation of the predetection filter. This is further emphasized by the fact that the data signatures with a θ parameter of θ_{max} are least likely to occur in practice.

As a final observation, note that the nominal filter from Chapter 4 provides near optimal performance for small phase errors. When $\theta \in [-1^\circ, 1^\circ]$, the asymptotic performance of the nominal filter is degraded by only .17 dB compared to the ideal BER, as seen in Table 5.1. For $\theta \in [-2.5^\circ, 2.5^\circ]$ this degradation increases only slightly to .44 dB and reaches only .91 dB for $\theta \in [-5^\circ, 5^\circ]$. It can therefore be stated that the nominal filter is robust with respect to phase errors in carrier recovery.

5.3 Filter Design for Adjacent Channel Interference

A second source of distortion common in digital communication is adjacent channel interference (ACI). This type of interference is inherent in frequency division multiple access (FDMA) systems where the available bandwidth is divided into separate transmission channels at different carrier frequencies. Since compact spectra enables more efficient use of the available bandwidth, CPM signals such as MSK, TFM, and duobinary MSK are good candidates for use in these systems. Consequently, there has been much interest in the error performance of various CPM schemes, as well as various filters used in MSK-type detection of these schemes, in the presence of ACI [12][13][14]. However, the approach has typically been to calculate the performance of specific predetection filters, designed either on the basis of nominal received signals or simple approximations to such filters.

Regardless of the spectral separation between adjacent carriers, some degree of interference always exists between channels. In this section the potential for performance gains resulting from the inclusion of ACI in the signal model is investigated using the AOLFP algorithm. The effect of ACI, no matter how minute, is included in the calculation of the inner product matrix, and the truly optimal filter (AOF) is then calculated. Since ACI dominates over noise at moderate to high SNR's, asymptotically optimal filters would normally be of interest in cases where ACI is present, because they are near optimal at practical SNR. Therefore, if significant gains are possible the approach could find widespread application.

The problem considered here is the transmission of a duobinary MSK signal with a second duobinary signal transmitted on the same medium in an adjacent channel above the desired signal. The spectral separation of the two carrier frequencies ΔfT is varied, and the AOF is calculated for each case.

5.3.1 Signal Model and Calculation of the Inner Products

As with any application of the AOLFP algorithm, the first step is to obtain a suitable signal model from which the information necessary to describe a single data signature on a generic observation interval can be determined. Since nonideal carrier recovery was studied in the previous section, it is assumed here that perfect carrier phase recovery exists for coherent detection at the receiver. Additionally, it is assumed that no other channel impairments exist other than AWGN.

The received signal consisting of a transmitted signal, interference signal, and noise is expressed as:

$$s(t) = A_c \cos(\omega_c t + \hat{\phi}_d(t) + \hat{\phi}_0) + A_c \sqrt{IC} \cos((\omega_c + \Delta\omega)t + \hat{\phi}_i(t - \tau) + \hat{\phi}_0 + \theta) + n(t) \quad (5.3.1)$$

where $\hat{\phi}_d$ is the excess-phase function of the desired signal derived from the data sequence that is to be recovered, $\Delta\omega$ defines the spectral separation between the desired and the interfering carriers, θ is the random phase offset between the two carriers, IC is the ratio of the received power in the interferer to the received power in the desired signal, $\hat{\phi}_i(t - \tau)$ is the excess-phase function of the interferer derived from some irrelevant data sequence whose bit intervals are offset in time by τ from the bit intervals of the desired data sequence, and $n(t)$ is a bandpass AWGN process with power spectral density $\frac{N_0}{2}$. Upon multiplication by $\cos(\omega_c t + \hat{\phi}_0)$ and neglecting the high frequency components that result, the signal model for the input to the I channel predetection filter is

$$\frac{A_c}{2} \left[\cos \hat{\phi}_d(t) + \sqrt{IC} \cos(\Delta\omega t + \hat{\phi}_i(t - \tau) + \theta) \right] + \frac{1}{2} n_c(t). \quad (5.3.2)$$

The problem to be studied is the detection of a +1 symbol transmitted by the $\cos \hat{\phi}_d(t)$ signal on a particular interval of duration $2T$. The inner product matrix is calculated from the data signatures of the expression in (5.3.2) (excluding noise) on some observation interval. The constant $\frac{A_c}{2}$ can be incorporated into the error performance so that for $IC = 1$ (equal strength carriers) the data signatures are found from

$$\begin{aligned} r(t) &= \cos \hat{\phi}_d(t) + \cos a(t) \cos \hat{\phi}_i(t - \tau) - \sin a(t) \sin \hat{\phi}_i(t - \tau) \\ &= r_d(t) + r_i(t) \end{aligned} \quad (5.3.3)$$

where $a(t) = (\Delta\omega t + \theta)$, $r_d(t)$ represents the component of $r(t)$ due to the desired signal, and $r_i(t)$ represents the component due to the interferer.

From the expression for $r(t)$, the information needed to fully describe a data signature can be ascertained. In fact, the signal in (5.3.3) is very similar to the signal in (5.2.2), except that the parameter θ in (5.2.2) is replaced by the time varying function $-a(t)$, and the $\cos \hat{\phi}_i(t)$ and $\sin \hat{\phi}_i(t)$ signals in (5.2.2) have been time shifted by τ . These two changes, however, drastically increase the difficulty involved in calculating the inner product matrix, in addition to increasing the size of the signal set.

As a first step toward identifying the parameters that determine each data signature, consider the eye diagrams of the signals $\cos \hat{\phi}_d(t)$, $\cos \hat{\phi}_i(t - \tau)$ and $\sin \hat{\phi}_i(t - \tau)$ shown in Fig. 5.7. On any specific T interval $[nT, nT + T]$, the $\cos \hat{\phi}_d(t)$ component of a specific data signature is equivalent to one of the nominal data signatures from Chapter 4, and can therefore be described by a sequence of basic pulses $p_i \in \{\pm p_1, \pm p_2, \pm p_3\}$ in Fig. 4.5. However, due to the time shift τ , two such pulses are required to describe each of $\cos \hat{\phi}_i(t - \tau)$ and $\sin \hat{\phi}_i(t - \tau)$ on each T interval, one pulse on $t \in [nT, (nT + \tau)]$ and a second on $t \in [nT + \tau, nT + T]$. As a result, many more data signatures exist as compared to the case when $\tau = 0$, which requires only a single pulse on each T interval. Although this does not present a problem in terms of a larger signal set because of the approach discussed in Chapter 3, many more component integrations must be calculated and stored. Moreover, the rules for combining the component inner products to form the inner product for a specific pair of data signatures becomes much more complex. To make this step as painless as possible, it has been assumed that $\tau = 0$ so that the phase transitions in the desired signal and the interferer occur simultaneously. This simplifies the application of the algorithm by essentially eliminating τ from the discussion. The $\cos \hat{\phi}_i(t)$ and $\sin \hat{\phi}_i(t)$ components of each data signature can then be described by a single pulse for each T interval in the observation interval. This eases the task of calculating the component inner products and simplifies the rules for combining them into the inner product between data signatures. The assumption itself is

Fig. 5.7 Eye diagrams of the data signature components for $\tau = .5T$.

valid in practice as it is reasonable in FDMA systems for the input data streams to be clocked by a common clock (given that the data streams operate at the same rate).

Having addressed the parameter τ in (5.3.3), the remaining parameters that describe a data signature are now considered. The $\cos \hat{\phi}_i(t)$ and $\sin \hat{\phi}_i(t)$ components of a data signature are each duobinary MSK signals and hence can be described by a set of pulses generated in the same manner as the $\cos \hat{\phi}_d(t)$ component. In addition, the two possible $\sin \hat{\phi}_i(t)$ components that exist for each $\cos \hat{\phi}_i(t)$ component can be found as described in section 5.2.1. Thus the description of each data signature involves three sets of integers, each describing one of the three signal components $\cos \hat{\phi}_d(t)$, $\cos \hat{\phi}_i(t)$, and $\sin \hat{\phi}_i(t)$ on the observation interval. Since the spectral separation $\Delta\omega$ is a fixed parameter, only θ remains to be considered in the description of a data signature.

The parameter θ is the random phase offset of the interferer's carrier waveform with respect to the desired signal's carrier waveform, and may therefore take on any value between 0 and 2π . Once again this situation presents a problem for the AOLFP algorithm as it results in an infinite number of received data signatures for any observation interval. As in the previous case, this situation requires that the AOLFP algorithm be applied to an approximate signal set. As before, the approach was to model θ as a uniformly distributed discrete random variable, taking on n_t values equally spaced from 0 to 2π . The signal set is then formed in the same manner as in section 5.2, by repeating the set of data signatures that may occur for a specific value of θ in n_t groups, each group corresponding to a different value for θ . A suitable value for n_t was determined by finding the AOF and corresponding BER curve for successively increasing values of n_t . It was observed that the difference between the results for $n_t = 11$ and $n_t = 15$ were insignificant, and $n_t = 11$ was deemed to be satisfactory. This result agrees with a similar finding in [12].

To determine the number of data signatures that comprise the signal set for a given observation interval, let n_d be the number of possible waveforms for $\cos \hat{\phi}_d(t)$ on

the observation interval for a +1 symbol being transmitted. (Note that this is exactly the number of data signatures in the signal set for the nominal filters in Chapter 4.) Since the interferer is not restricted to transmitting a +1 on the observation interval, the possible $\cos \hat{\phi}_i(t)$ waveforms consist of the same waveforms as $\cos \hat{\phi}_d(t)$, plus the negatives, for a total of $2n_d$ waveforms. Since there are two $\sin \hat{\phi}_i(t)$ waveforms possible for every $\cos \hat{\phi}_i(t)$ waveform there are a total of $4n_d$ possible waveforms for the interferer. Thus there are $4n_d^2$ received data signatures for each value of θ and hence $4n_d^2 n_t$ data signatures in the signal set for any observation interval.

The calculation of the inner products for these data signatures is more difficult than the previous case due to the time varying function $a(t)$ in (5.3.3). To find the set of component inner products, note that the inner product for two data signatures $r_1(t)$ and $r_2(t)$ can be expressed as:

$$\langle \bar{r}_1, \bar{r}_2 \rangle = \langle \bar{r}_{d_1}, \bar{r}_{d_2} \rangle + \langle \bar{r}_{d_1}, \bar{r}_{i_1} \rangle + \langle \bar{r}_{i_1}, \bar{r}_{d_2} \rangle + \langle \bar{r}_{i_1}, \bar{r}_{i_2} \rangle. \quad (5.3.4)$$

The first term in (5.3.4) is merely an inner product between two realizations of the $\cos \hat{\phi}_d(t)$ signal (nominal data signatures) and hence are simply elements of the inner product matrix calculated for the nominal case. The remaining three terms involve θ and the time varying function $a(t)$, and therefore the component inner products for these terms must be found numerically. To perform these integrations the time origin was chosen (as usual) to be the beginning of a $2T$ interval corresponding to the transmission of a +1 symbol by $\cos \hat{\phi}_d(t)$, and the observation interval was centered at $t = \frac{3}{2}T$. The second and third terms in (5.3.4) involve the inner product between a $\cos \hat{\phi}_d(t)$ waveform (described completely by a sequence of pulses), and an interferer defined by a value for θ , one sequence of pulses for $\cos \hat{\phi}_i(t)$, and a second sequence for $\sin \hat{\phi}_i(t)$. In similar fashion to the phase error case, the set of component inner products must be found for all possible combinations of pulses that can occur, and then repeated for all possible values of θ . Furthermore, due to the time varying nature

of $a(t)$, the component inner products differ from T interval to T interval and must be evaluated and stored for each such interval in the observation interval. The last term in (5.3.4) is an inner product between two interferers and hence involves four sets of pulses and two values of θ . Here considerably more component inner products must be computed compared to the second and third terms since more combinations of pulses exist, as well as more combinations of θ values. Again, the presence of the time varying function $a(t)$ requires that these values be computed for all T intervals in the observation interval. Finally, rules must be developed that, from the description of any two data signatures, allow the overall inner product to be found by appropriately combining specific component inner products. The reader is spared these details.

The component inner products were calculated numerically using the Romberg integration technique. In this method, the number of subintervals in the range of integration is increased in a nonlinear fashion until the change in successive approximations to the integral is less than some predetermined amount. For the integrations calculated here, an accuracy of 10^{-6} was required, typically being achieved in less than 8 iterations, and never more than 12 iterations.

One final but important note must be made with regards to the parameter θ . Recall that the signal set must be comprised of all possible data signatures that may occur on a generic observation interval. In other words, the same signal set must result regardless of a shift of $2kT$ in both the observation interval and the interval assumed to transmit a $+1$, for k any integer. In the two previous cases studied, there was no time varying function similar to $a(t)$ involved in the description of the data signatures and as a result, the signal set could be found from the eye diagram on any arbitrary interval on which a $+1$ was assumed to be transmitted. The time origin was not important as the same signal set would be constructed if the observation interval was shifted by an arbitrary number of $2T$ intervals to some other interval on which a $+1$ was transmitted.

In the ACI case however, the time varying function $a(t)$ prevents this from being the case. To see why, recall that any interferer $r_i(t)$ in (5.3.3) can be expressed as

$$\begin{aligned} r_i(t) &= \cos(\Delta\omega t + \hat{\phi}_i(t) + \theta) \\ &= \cos a(t) \cos \hat{\phi}_i(t) - \sin a(t) \sin \hat{\phi}_i(t). \end{aligned} \quad (5.3.5)$$

The waveforms $\cos a(t)$ and $\sin a(t)$ are shown in Fig. 5.8 for $\Delta\omega = \frac{\pi}{2T}$ and $\theta = 0$ on the observation interval of duration $L = 3T$.

Fig. 5.8 Time varying portion of $r_i(t)$.

Since the $\cos \hat{\phi}_i(t)$ and $\sin \hat{\phi}_i(t)$ descriptions for each data signature are derived from the nominal eye diagrams, the time origin has no effect on these descriptions and they are found simply by centering an open eye diagram at $t = \frac{3}{2}T$ and constructing all possible waveforms over the observation interval $t \in [0, 3T]$. If the +1 symbol was assumed to be transmitted on the next signalling interval $t \in [2T, 4T]$ in Fig. 5.8, then the observation interval would be $t \in [2T, 5T]$ and the $\cos \hat{\phi}_i(t)$ and $\sin \hat{\phi}_i(t)$ components of the data signatures would become the possible waveforms for an open eye centered at $t = \frac{7}{2}T$, resulting in exactly the same descriptions. However, as seen in Fig. 5.9, the $\cos a(t)$ and $\sin a(t)$ portions of the data signatures on this observation interval are

phase shifted by π as compared to the observation interval in Fig. 5.8. Therefore, the group of data signatures having $\theta = 0$ in the signal set constructed on the observation interval in Fig. 5.8 are mapped to the group of data signatures having $\theta = \pi$ in the signal set constructed on the observation interval in Fig. 5.9.

Fig. 5.9 Time varying portion of $r_i(t)$ on a time shifted observation interval.

This situation is a direct result of the need to have a definite time origin to compute the component inner products numerically. Only when $\Delta\omega = \frac{\pi}{T}$ would each interferer appear without a phase shift on successive observation intervals. Thus for any other value of $\Delta\omega$, a specific interferer described by (5.3.5) would appear to correspond to one value of θ on one observation interval and another value of θ on a different observation interval. However, by expanding the signal set to allow several values of θ , essentially the same signal set would be constructed on any observation interval regardless of the choice of time origin. There always exists a one to one mapping of the data signatures having one value of θ on one observation interval to the data signatures having another value of θ on a second observation interval. In effect, the signal vectors in the two signal

sets are simply reordered and both sets describe the signal for a generic observation interval, as required by the AOLFP algorithm. Note that for an arbitrary value of $\Delta\omega$, mappings that exactly reproduce the interferer (like $\Delta\omega = \frac{\pi}{2T}$ in Fig. 5.8 and Fig. 5.9) do not always exist because only a finite number of θ values are considered. This will not affect the results however, as 11 values for θ were found to provide sufficient accuracy. Increasing the number of θ values to make the mappings more exact would have no significant effect on the results.

5.3.2 Results

Specific values were chosen for the spectral separation ΔfT , each value corresponding to a specific fractional power bandwidth of duobinary MSK. Since the desired signal and interferer have identical power spectra, the spectral separation needed for a specific percentage of spectral overlap can be easily found from the power spectra of duobinary MSK given in [15]. For example, from [15] 90% of the total power in the duobinary MSK signal lies in the range $f \in [f_c - .554/2T, f_c + .554/2T]$. Therefore alignment of the 90% fractional power bandwidths of the two spectra requires $\Delta fT = .554$. Similarly, spectral separations equal to the 95%, 99%, 99.9%, and 99.99% fractional power bandwidths require $\Delta fT = .654, .862, 1.80, \text{ and } 3.62$ respectively. The very poor error performance that results for $\Delta fT = .554$ limits this to the minimum value considered.

The first calculations were done for an observation interval of duration $L = 3T$, resulting in a signal set containing 2816 signal vectors. The correlating waveforms describing the AOF calculated on this interval for each of the above values of ΔfT are shown in Fig. 5.10. These curves illustrate clearly how the calculated AOF approaches the nominal filter as ΔfT increases, i.e., as the received signal approaches the nominal signal.

Since the baseband noise process $n_c(t)$ at the input to the predetection filter has a power spectral density of N_0 , the error probability is given by equation (5.2.10).

Fig. 5.10 Correlating waveforms of the AOF's for $L = 3T$ and ACl.

In this case however, the factor E_b represents the received energy in only the desired duobinary MSK signal during one bit interval. The performance of each AOF and the performance that would be achieved by the nominal filter is given in Fig. 5.11. By comparing these results it is evident that a significant improvement in performance is possible when the effect of ACI is included in the calculation of the AOF. The largest performance gains result for smaller values of ΔfT where the distortion due to ACI is greatest.

To see what gains are possible at practical SNR's, consider the curves in Fig. 5.11 for $\Delta fT = .654$. At a BER of 10^{-5} , the AOF requires a SNR that is 3.6 dB below that required by the nominal filter, a significant improvement in performance. This improvement increases as SNR increases, reaching 4.6 dB in the asymptotic limit of infinitely large SNR. For the larger spectral separation of $\Delta fT = .862$ and an error probability of 10^{-6} , the AOF is seen to provide a gain of 1.5 dB over the nominal filter. This increases to 2.4 dB asymptotically. Thus, even for a fairly large spectral separation, in this case equal to the 99% fractional power bandwidth of MSK, significant performance gains are possible when the effect of the ACI is included in the calculation of the AOF. These gains are also possible when the spectral separation is increased, further diminishing the amount of distortion due to ACI. For example, with $\Delta fT = 3.62$ the calculated AOF still provides an asymptotic gain of .3 dB over the nominal filter. This fact is further illustrated by comparing the error curves for the AOF calculated for $\Delta fT = 1.8$ and the nominal filter for $\Delta fT = 3.62$. Although the nominal filter sees a signal suffering a smaller degree of distortion, the AOLFP algorithm is able to exploit the information about the ACI present when $\Delta fT = 1.8$ and calculate an AOF that provides virtually equivalent performance, despite a larger level of distortion. This is demonstrated by the slightly smaller degradation from the ideal curve for the AOF for $\Delta fT = 1.8$ as compared to the degradation of the nominal filter for $\Delta fT = 3.62$. Asymptotically, the calculated AOF results in a .2 dB improvement over the nominal

Fig. 5.11 BER of the AOF's and nominal filter for $L = 3T$ and ACl.

filter, despite the smaller spectral separation. While these smaller gains would not likely be pursued in practice, the results clearly indicate that gains are possible through the use of the AOLFP algorithm and the method for applying it discussed in Chapter 3.

Similar calculations were made for observation intervals of duration $L = 4T$ and $L = 5T$. The signal sets that result in these cases are quite large, consisting of 45,056 signal vectors. Note that had the assumption that $\tau = 0$ not been made, enormous signal sets would result. Although the method of Chapter 3 provides the means for finding the AOF in this case, the required computing time would be far too extreme for the purpose of this study.

The correlating waveforms describing the AOF for $\Delta fT = .554$, $.654$, and $.862$ are shown for $L = 4T$ and $L = 5T$ in Fig. 5.12 and Fig. 5.13 respectively. The performance of the calculated AOF's and the nominal filters on these observation intervals are given in Fig. 5.14 and Fig. 5.15. These results indicate that the performance improvement provided by the calculated AOF over the nominal filter decreases as the observation interval becomes longer. For example, for $\Delta fT = .654$ and a BER of 10^{-5} the AOF of duration $4T$ provides a .8 dB improvement over the nominal filter while the AOF of duration $5T$ provides an improvement of .4 dB. This trend is reflected in the asymptotic gains as well, decreasing from 4.6 dB for the AOF of duration $3T$ to 1.9 dB and 1.1 dB for the AOF's having duration $4T$ and $5T$ respectively. This behaviour is also reflected in the curves for other values of ΔfT in Fig. 5.14 and Fig. 5.15. These results suggest that for duobinary MSK and perhaps other signals as well, filter design based only on nominal signals may be acceptable when ACI is present, provided the impulse response duration exceeds $5T$. If however it is desirable to have a short duration impulse response, it is definitely advantageous to apply the extended AOLFP algorithm to a more accurate signal model.

One can also see from Fig. 5.14 and Fig. 5.15 that the incremental performance improvement between AOF's calculated on various observation intervals decreases as

Fig. 5.12 Correlating waveforms of the AOF's for $L = 4T$ and ACL.

Fig. 5.13 Correlating waveforms of the AOF's for $L = 5T$ and ACl.

Fig. 5.14 BER of the AOF's and nominal filter for $L = 4T$ and ACl.

Fig. 5.15 BER of the AOF's and nominal filter for $L = 5T$ and ACL .

longer observation intervals are chosen. This is seen for example by the AOF for $\Delta fT = .862$. Increasing the observation interval duration from $3T$ to $4T$ results in a .61 dB performance gain asymptotically, while increasing the observation interval from $4T$ to $5T$ results in only a .28 dB gain. This was the case for the nominal signals in Chapter 4 and is always the case, regardless of the complexity of the signal model.

One final fact to note is that the correlating waveforms for $\Delta fT = .554$, .654, and .862 do not tend to zero at the edges of the observation interval. This indicates that further gains are possible if the observation interval is extended. This analysis was not done due to the already small performance gains provided for $L = 5T$ and the large computation time required for durations of $4T$ and $5T$. Extending the observation interval to $6T$ alone would result in an enormous signal set, although it could be handled by the extended algorithm. Consequently only observation intervals for $L < 6T$ were considered in this analysis, as was the case in [13]. Furthermore, the main objective of this thesis has been to demonstrate the ability of the AOLFP algorithm to calculate predetection filters that provide improved performance when accurate signal models are used, which has been amply demonstrated for $L = 3T$, $L = 4T$, and $L = 5T$.

To summarize, it has been shown that by providing the AOLFP algorithm with accurate inner products, an AOF capable of significant performance gains over the nominal filter can be calculated. In fact, for every combination of ΔfT and L considered, the error curve of the AOF lies below the corresponding error curve for the nominal filter at practical and high SNR's, indicating a performance gain of some degree.

5.4 Filter Design for a Bandlimited Signal

In the final problem studied, the transmitted duobinary MSK signal is bandlimited prior to transmission over the channel. This is done in practice to reduce out of band emissions that would interfere with the bandwidth allocated to nearby signalling channels, for example on a microwave link. If the details of the bandpass filter acting on the transmitted signal

$$\begin{aligned} s(t) &= A_c \cos(\omega_c t + \hat{\phi}(t) + \hat{\phi}_o) \\ &= A_c \cos(\omega_c t + \hat{\phi}_o) \cos \hat{\phi}(t) - \sin(\omega_c t + \hat{\phi}_o) \sin \hat{\phi}(t) \end{aligned} \quad (5.4.1)$$

are known, then the equivalent lowpass filtering effect on the baseband signals $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t)$ can be determined. Due to the fundamental relationship between time and frequency, the bandlimiting process results in a “spreading out” of each signal in time, causing the effect of a single data symbol to extend beyond one symbol interval. This additional intersymbol interference (ISI) augments the memory already inherent in the modulation process and increases the number of data signatures that may be received on a given observation interval. In this final section the effect of the additional ISI is included in the calculation of the inner product matrix in the hope achieving performance gains over the nominal filter.

5.4.1 Signal Model and Calculation of the Inner Products

Conceptually, the signalling problem under consideration can be thought of as shown in Fig. 5.16(a) where the transmitted signal $s(t)$ passes through a bandpass filter prior to being applied to the channel. The channel adds only AWGN noise to the signal as in previous cases and it is assumed that ideal carrier recovery exists at the receiver. An equivalent view of the problem is shown in Fig. 5.16.(b) where the filtering process has been shifted to baseband so that $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t)$ are filtered directly. Since the $\cos \hat{\phi}(t)$ and $\sin \hat{\phi}(t)$ signals are filtered in exactly the same way, the

(a)

(b)

Fig. 5.16 Equivalent views of the transmitter for a bandlimited signal: (a) with a bandpass filter, and (b) with lowpass filters.

same AOF results in both channels, therefore only the I channel is considered in the analysis.

As in previous cases, the waveforms that can occur for the nominal signal $\cos \hat{\phi}(t)$ can be described by a sequence of integers indicating which of the three basic pulses in Fig. 4.5 describes the waveform on successive symbol intervals. In this way, all possible inputs to the lowpass filter $H_{LP}(\omega)$ can be expressed as

$$\sum_{m=-\infty}^{\infty} \alpha_m p^{(m)}(t - mT), \quad (5.4.2)$$

where $p^{(m)}$ is the basic pulse occurring on the interval $t \in [mT, mT + T]$ and each $\alpha_m \in \{\pm 1\}$ indicates the polarity of the pulse. If $y_j(t)$ for $j \in \{1, 2, 3\}$ are the filter responses to the basic pulses $p_j(t)$, then the output signal $r(t)$ in Fig. 5.16(b) can be expressed as

$$r(t) = \sum_{m=-\infty}^{\infty} \alpha_m y^{(m)}(t - mT) \quad (5.4.3)$$

where $y^{(m)}(t)$ is the response to the basic pulse occurring in the input on the interval $t \in [mT, mT + T]$. With ideal carrier recovery at the receiver, $r(t)$ is the signal, exclusive of noise, appearing at the input to the predetection filter. Therefore the AOF is determined from the data signatures of this signal.

As in previous cases, the time origin was chosen such that a +1 symbol was transmitted on the interval $t \in [0, 2T]$. To see what information is needed to describe a single data signature of $r(t)$, consider the case of $H_{LP}(\omega) = 1$ for all ω . In this case, the three output pulses $\{y_j(t)\}$ are identical to the three pulses $\{p_j(t)\}$, making $r(t)$ equivalent to $\cos \hat{\phi}(t)$. As a result, describing a data signature simply requires one integer giving the basic pulse and its polarity for each symbol interval in the observation interval. For any practical filter however, the three output pulses $\{y_j(t)\}$ have infinite duration due to the discontinuities in the pulses $\{p_j(t)\}$. As a result, an infinite number of symbol intervals prior to the observation interval must be considered in calculating the inner product matrix. Thus it appears, as in the previous cases involving interference,

that the signal set involves an infinite number of data signatures and some form of approximation is necessary to apply the AOLFP algorithm. Fortunately, in this case the approximation is straightforward since, for any practical filter $H_{LP}(\omega)$, each pulse can realistically be considered zero after some finite duration. It is therefore sufficient to consider only a finite number of symbol intervals prior to the observation interval. If N_1 is large enough that it can be assumed that

$$\begin{aligned} y_i(t) &\neq 0 & 0 \leq t \leq N_1 T \\ y_i(t) &= 0 & t > N_1 T \end{aligned} \quad (5.4.4)$$

without affecting the inner product matrix significantly, then the number of pulses required to describe a data signature is $N_1 + N_2$, where N_2 is the number of symbol intervals in the observation interval. The signal set is therefore described by all possible sequences of $(N_1 + N_2)$ pulses having positive polarity on $t \in [T, 2T]$, where the pulses belong to the set $\{\pm y_j(t)\}_{j \in \{1,2,3\}}$. If N_1 is such that the sequence of pulses for each data signature span the interval $t \in [-pT, kT]$, then each data signature $r_i(t)$ can be expressed as

$$r_i(t) = \sum_{m=-p}^k \alpha_{m,i} y_i^{(m)}(t - mT). \quad (5.4.5)$$

The expression in (5.4.5) constitutes the signal model for each received data signature. The pulses $\{y_i^{(m)}(t)\}$ for $m = -p \dots -1$ are transmitted prior to the +1 symbol at $t = 0$ and contribute significant ISI on the observation interval. The appropriate value of k is determined by the observation interval duration. The value of p required in each application is found by increasing p until there is no significant change in the results for further increases. This amounts to improving the accuracy of the approximation in (5.4.5) with respect to the actual description in (5.4.3) until the discrepancy is negligible for finding the inner products.

Given the description of a data signature in (5.4.5), the inner product between any

two data signatures $r_1(t)$ and $r_2(t)$ can be expressed as

$$\langle r_1(t), r_2(t) \rangle = \sum_{m=-p}^k \sum_{n=-p}^k \alpha_{m_1} \alpha_{n_2} \langle y_{i_1}^{(m)}(t - mT), y_{i_2}^{(n)}(t - nT) \rangle. \quad (5.4.6)$$

The component inner products needed to calculate the inner product matrix can be determined from the inner product term in (5.4.6). For m and n fixed, the term represents an integration over the entire observation interval of the product of two pulses at different (possibly) offsets from the origin. Therefore, the set of component inner products is found by calculating this integral for the set of integrands arising from all possible combinations of m and n and for all possible pulse shapes $y_{i_1}^{(m)}$ and $y_{i_2}^{(n)}$ that can occur on the intervals $[mT, mT + T]$ and $[nT, nT + T]$.

Since the polarity information (α_{m_1} and α_{n_2} in (5.4.6)) can be considered when the summation in (5.4.6) is taken, only integrands involving positive pulses are required for finding the component inner products. Therefore the integrands necessary to find the component inner products for the interval $[-pT, -pT + T]$ involve all combinations of the positive pulses on this interval and all successive intervals up to and including $[kT, kT + T]$. The component inner products for the next interval $[-pT + T, -pT + 2T]$ are found in the same way and the procedure continues for the next $k + p - 1$ intervals, culminating in the component inner products for the interval $[kT, kT + T]$. Note that the number of possible integrands decreases with each step, with only three possible integrands for the component inner products on the final interval $[kT, kT + T]$. Having integrated each of these integrands over the observation interval, the value of every inner product in (5.4.6) can be calculated given the actual pulses and associated polarities that describe any two data signatures. This approach enables the inner product matrix to be calculated while performing a minimum number of numerical integrations.

Fig. 5.17 Third order Butterworth phase characteristic.

Since the component inner products are found numerically the expressions for $y_1(t)$, $y_2(t)$ and $y_3(t)$ are required. Therefore the responses to $p_1(t)$, $p_2(t)$ and $p_3(t)$ must be calculated for each filter response $H_{LP}(\omega)$ of interest. In this analysis, a third order Butterworth frequency response was chosen so that

$$H_{LP}(\omega) = \frac{\omega_c^3}{(s + \omega_c)(s^2 + s\omega_c + \omega_c^2)}, \quad (5.4.7)$$

where ω_c is the 3 dB cutoff frequency of the filter. The phase characteristic for the filter is given in Fig. 5.17 and indicates that the phase is approximately linear over the frequency range $\frac{\omega}{\omega_c} \in [0, 1.5]$. This property is desirable because the filter tends not to augment the ISI it produces in the output with additional distortion due to a nonlinear phase characteristic. As the group delay τ of the filter is given by

$$\tau = -\frac{d\theta(\omega)}{d\omega}, \quad (5.4.8)$$

the filter will simply introduce a constant delay in the output, provided a large percentage of the power spectrum of the duobinary MSK input is contained in the linear region. This constant delay is also important with respect to recovery of the $\frac{1}{2T}$ clock in the receiver. Since the clock is recovered from a difference of the received I and Q channel signals, the constant delay in the signals will be reflected in the recovered clock. It is therefore necessary to shift the observation interval used to calculate the inner products so as to be centered at the center of the eye opening at $t = \frac{3}{2}T + \tau$.

From Fig. 5.17, the group delay over the linear region is found to be approximately given by

$$\tau = \frac{.369}{f_c} T. \quad (5.4.9)$$

Using this result and the calculated responses $y_1(t)$, $y_2(t)$ and $y_3(t)$ for a given cutoff frequency f_c , the inner product matrix can be calculated and the AOLFP algorithm applied.

5.4.2 Results

The AOLFP algorithm has been applied for two values of f_c . The first value considered was $f_c = \frac{.375}{T}$ which results in 4.8% of the power in the duobinary MSK input signal being filtered out. The eye diagram for the resulting 95.2% power bandwidth signal is shown in Fig. 5.18, indicating the ISI that results from the bandlimiting. The eye diagram was generated by plotting the possible data signatures for a +1 or -1 symbol occurring in the transmitted duobinary MSK signal on $t \in [0, 2T]$. In the nominal case, the eye opening would be centered at $t = \frac{3}{2}T$. In Fig. 5.18 however, the eye opening is centered at $t = 2.48T$ which is consistent with a group delay of $.98T$ as predicted by equation (5.4.9). The data signatures themselves are simply summations of shifted (and possibly inverted) versions of the responses $y_1(t)$, $y_2(t)$ and $y_3(t)$ shown in Fig. 5.19.

Fig. 5.18 Eye diagram of bandlimited duobinary MSK for $f_c = .375/T$.

Fig. 5.19 Responses to the basic pulses for $f_c = .375/T$.

The observation interval was centered at $t = 2.48T$, which is optimum for the detection of the +1 symbol transmitted on $[0, 2T]$. With a duration of $L = 3T$, the observation interval is $t \in [.98T, 3.98T]$. From Fig. 5.19 the effective duration of the pulses $y_i(t)$ is $5T$. Therefore, pulses occurring on $t \in [-3T, -2T]$ are required to accurately describe the received data signatures, requiring that $p = 3$ in (5.4.5). Since the pulse occurring for $t \in [4T, 5T]$ does not fall in the observation interval, the proper value for k is 3. Thus each data signature is described as a summation of seven pulses occurring on the T intervals $[-3T, -2T]$ to $[3T, 4T]$, resulting in a signal set of 128 signal vectors. The signal sets for $L = 4T$ and $L = 5T$ were found similarly.

To increase the distortion due to ISI, the cutoff frequency of the lowpass filter was decreased to $f_c = \frac{25}{T}$. This results in 15.9% of the power in the input duobinary MSK signal being removed. The narrower filter bandwidth increases the spreading effect that causes the ISI, as seen by the increased duration of the responses $y_1(t)$, $y_2(t)$ and $y_3(t)$ to $6T$ as shown in Fig. 5.20. The increased amount of ISI that results is evident in the eye diagram for $f_c = \frac{25}{T}$ shown in Fig. 5.21. The eye is seen to be centered at approximately $t = 3T$, corresponding to a group delay of $\tau = 1.5T$. Although 13% of the input signal power is outside the linear range of the filter phase characteristic, the observed delay agrees well with the value $\tau = 1.47T$ predicted by equation (5.4.9). The observation intervals considered for this case were centered at $t = 3T$ and covered durations of $L = 3T$, $L = 4T$ and $L = 5T$. For the cases of $L = 3T$ and $L = 4T$, four symbol intervals prior to $t = 0$ were sufficient to accurately describe the resulting data signatures. One additional symbol interval was required for the case of $L = 5T$.

The correlating waveforms for each value of f_c are shown in Figs. 5.22-5.24,. Although the correlating waveforms were calculated on time shifted observation intervals, this has no effect on the impulse response of the AOF which is always found by reflecting the correlating waveform about the sampling point and shifting it so as to be nonzero only on $t \in [0, L]$. The delay involved in calculating the inner products is

Fig. 5.20 Responses to the basic pulses for $f_c \approx .25/T$.

Fig. 5.21 Eye diagram of bandlimited duobinary MSK for $f_c = .25/T$.

Fig. 5.22 Correlating waveforms of the AOF's for $L = 3T$ and bandlimiting.

Fig. 5.23 Correlating waveforms of the AOF's for $L = 4T$ and bandlimiting.

Fig. 5.24 Correlating waveforms of the AOF's for $L = 5T$ and bandlimiting.

incorporated into a constant delay in the sampling instants at the receiver. Thus the time axis on which the correlating waveform is calculated is irrelevant and the correlating waveforms are plotted on the observation interval over which the nominal filter of equal duration is found.

Upon comparison of the calculated and nominal correlating waveforms, one fact is immediately apparent. The correlating waveforms (and therefore the impulse responses) are not symmetric about the center of the observation interval, unlike the nominal filter which is symmetric. This result is due to the fact that the eye diagrams in Fig. 5.18 and Fig. 5.21 are not symmetric about the center of the eye opening, a property resulting from the filtering process.

To calculate the error performance of the AOF's, recall from equation (5.2.9) that for any signal $r_i(t)$ at the input to the predetection filter, the BER is given by

$$P_e = \frac{1}{N} \sum_{i=1}^N Q \left\{ \frac{A_c}{\sqrt{N_0}} \langle \bar{r}_i, \bar{r}_x \rangle \right\}. \quad (5.4.10)$$

To obtain a more useful expression for P_e , let P' be the average power in the signal $r(t)$ at the input to the multipliers in Fig. 5.16(b). The average power at the output of the multipliers in each channel is $\frac{A_c^2 P'}{2}$ and hence the average power in the transmitted signal is $A_c^2 P'$. Since the channel is assumed ideal, this is also the average power in the received signal, making the average received energy in one bit time $E_b = A_c^2 P' T$. Thus in general, the BER is given by

$$P_e = \frac{1}{N} \sum_{i=1}^N Q \left\{ \sqrt{\frac{E_b}{P' N_0}} \frac{\langle \bar{r}_i, \bar{r}_x \rangle}{\sqrt{T}} \right\}. \quad (5.4.11)$$

In the context of the bandlimiting problem, the problems studied in Section 5.2 and Section 5.3 correspond to a filter response $H_{LP}(\omega) = 1$ for all ω , so that P' is the average power in the $\cos \hat{\phi}(t)$ signal. Recall from the Markov nature of duobinary MSK that $\cos \hat{\phi}(t)$ may be described on two successive T intervals by two pulses p_i and p_j where, if $p_i \in \{\pm p_1, \pm p_2\}$ then $p_j \in \{\pm p_2, \pm p_3\}$. This pattern repeats on

successive pairs of T intervals so that the possible behaviour of the signal over any two T intervals is given by the eight pairs of pulses $\{p_1, p_2\}, \{p_1, p_3\}, \{p_2, p_2\}, \{p_2, p_3\}$, plus the negatives of these pairs. The average power can therefore be found by calculating the average power for these pairs of pulses. The result is $P' = \frac{1}{2}$ for the nominal signal $\cos \hat{\phi}(t)$ so that $E_b = \frac{A_c^2 T}{2}$ and (5.4.11) reduces to (5.2.10) when $H_{LP}(\omega) = 1$. When $H_{LP}(\omega) \neq 1$, P' can be found by substituting the output pulses y_j in place of the pulses p_j , and calculating the average energy over any $2T$ interval numerically. Naturally, to obtain an accurate calculation of P' it is necessary, as with the integrations for the component inner products, to consider pulses on several T intervals prior to the integration interval. The result for $f_c = \frac{375}{T}$ is $P' = .476$, while $P' = .4205$ results for $f_c = \frac{25}{T}$. Using these values, the BER for the AOF's in Figs. 5.22-5.24 have been calculated and are given in Figs. 5.25-5.27. To see if the inclusion of ISI in the inner products provides improved performance, the Figures include the BER that results when each bandlimited signal is detected by the nominal filter. The "ideal" curve is the BER that results when the nominal signal is detected by the nominal filter.

Consider first the results in Fig. 5.25 for the observation interval $L = 3T$. At a BER of 10^{-5} , the AOF for the 95.2% power bandwidth (PBW) signal ($f_c = \frac{25}{T}$) provides an improvement of only .25 dB over the nominal filter. The AOF for the 84.1% PBW signal provides no improvement. As the BER decreases however, the improvement provided by each AOF as compared to the nominal filter increases. At a BER of 10^{-7} the gain provided by the AOF for the 95.2% PBW signal increases to .32 dB and reaches .74 dB in the asymptotic limit of infinitely large SNR. The AOF for the 84.1% PBW signal begins to show an improvement over the nominal filter at a BER of 10^{-7} , with a .1 dB performance gain. This gain increases to .42 dB asymptotically.

Due to the spreading effect of the signal in the time domain, it is reasonable to expect that the performance gain provided by the AOF for a particular bandlimited signal would increase as the duration of the observation interval is increased. The

Fig. 5.25 BER of the AOF's and nominal filter for $L = 3T$ and bandlimiting.

Fig. 5.26 BER of the AOF's and nominal filter for $L = 4T$ and bandlimiting.

Fig. 5.27 BER of the AOF's and nominal filter for $L = 5T$ and bandlimiting.

results in Fig. 5.26 for $L = 4T$ illustrate that this is in fact correct. At a BER of 10^{-5} the AOF for the 95.2% PBW signal provides a .35 dB improvement over the nominal filter, while the AOF for the 84.1% PBW signal provides a .55 dB improvement. These performance gains increase to .88 dB and 1.12 dB respectively as SNR becomes infinitely large, compared to the asymptotic gains of .74 dB and .42 dB provided by the AOF's calculated on $L = 3T$. Increasing the observation interval further to $L = 5T$ shows a continuation of the trend. At a BER of 10^{-5} the AOF for the 95.2% PBW signal provides a .39 dB gain over the nominal filter and the AOF for the 84.1% PBW signal provides a gain of .78 dB. Asymptotically, the improvement of these filters increases to .91 dB and 1.28 dB respectively, as seen in Table 5.3 which gives the degradation in performance of each filter compared to the BER of MSK in the limit as SNR becomes infinite.

L	$f_c = .375/T$		$f_c = .25/T$	
	AOF (dB)	Nominal (dB)	AOF (dB)	Nominal (dB)
$3T$	1.66	2.41	3.34	3.76
$4T$	1.30	2.16	2.38	3.50
$5T$	1.19	2.10	2.18	3.46

Table. 5.3 Asymptotic degradation of the AOF and nominal filters for bandlimited duobinary MSK from the BER of MSK.

The superiority of the AOF's over the nominal filters is also illustrated by the performance degradation of each filter as compared to the ideal curve, which is the best possible performance attainable for duobinary MSK using any filter under any circumstance. For each bandlimited signal considered, the BER curve of the calculated AOF's approach the ideal performance much more quickly than the nominal filters as L increases. For example, for the 84.1% PBW signal at a BER of 10^{-6} , the performance

of both the nominal filter and AOF on $L = 3T$ are degraded by 1.8 dB from the ideal BER curve. When L is increased to $4T$, the AOF performance degradation is reduced to 1.0 dB, while the degradation of the nominal filter remains virtually unchanged at 1.7 dB. A further increase to $L = 5T$ results in reducing the degradation in the AOF performance to .7 dB, while the degradation for the nominal filter again falls only slightly to 1.6 dB. The same behaviour is observed for the 95.2% PBW signal, with the AOF for $L = 5T$ actually matching the ideal performance for BER's down to 10^{-8} with negligible degradation for smaller BER's as seen in Fig. 5.27. Thus, although the performance of the nominal filter for $L = 5T$ degrades from the ideal BER, it is still possible to achieve virtually ideal performance by including the known behaviour of the ISI in the calculation of the inner product matrix.

From the results presented in this Section it is clear that at practical error rates, if the effect of ISI caused by a known source of bandlimiting is included in the application of the AOLFP algorithm, some performance improvement over the more easily calculated nominal filter can be achieved. Thus, the AOLFP algorithm is capable of providing some measure of equalization against bandlimiting, providing the details of the bandlimiting are sufficiently well-understood. This claim is substantiated by the fact that the AOF performance exceeded the nominal filter performance at practical error rates (or less) for every case studied, as was the case when ACI was the source of the distortion.

Although the results have been calculated assuming a third order Butterworth bandlimiting characteristic, the same analysis can be performed with the known bandlimiting characteristics of any practical signalling channel. The problem can be viewed as a cascade of a bandlimiting filter $H(\omega)$ and an ideal AWGN channel where $H(\omega)$ models the bandlimiting effect of the channel. In cases where the channel characteristic is closely approximated by a third order Butterworth filter, the results in this Section are directly applicable.

As a final note, although the performance improvements provided by the AOF filters in this Section are sometimes small, the intent has been to demonstrate that the AOLFP algorithm can use information that describes distortion in a received signal to achieve improved error performance. Whether that performance gain is large enough to warrant choosing the AOF over the nominal filter in practice is not the primary issue. In any case, the results provided in this Section can always be used as performance benchmarks for systems involving duobinary MSK, as the error curves give the best performance attainable with a linear receiver when the signal is bandlimited.

CHAPTER 6

CONCLUSIONS

6.1 Summary

Since the entire thesis involves the use of the AOLFP algorithm, a thorough discussion of the algorithm was given in Chapter 2. It was noted that due to the completely general nature of the inner product matrix, the algorithm can be applied to many binary communication signals, but is most suited to signals with the property that many distinct waveforms may be received for a single transmitted symbol. Since various sources of interference can produce this behaviour, it was suggested that improved error performance may result when interference effects are included in the inner product matrix.

Chapter 3 discussed the advantage of using dynamic memory allocation in a software implementation of the algorithm, and provided a solution to the quadratic programming portion of the algorithm. The problem presented by large signal sets in terms of computing time and memory space was also addressed. An extension to the algorithm was developed, resulting in the memory requirements being determined by the number of active vectors of the AOF vector, as opposed to the number of signal vectors in the signal set as is the case with the original statement of the algorithm. This extension vastly increases the number of signal vectors the algorithm can process.

Since the practical signalling cases studied involve MSK-type detection of duobinary MSK, these topics were discussed in Chapter 4. Nominal results were obtained by applying the algorithm to the signal in the absence of interference, for the purpose of judging the results in Chapter 5.

Chapter 5 dealt with the application of the extended algorithm to practical signalling problems and began with a general approach to applying the algorithm. In Section 5.2, the effect of a random phase error in carrier recovery was included in the inner product

matrix. The AOF and nominal filter were seen to suffer essentially equivalent performance degradation from the ideal BER of duobinary MSK. It was concluded that the degradation occurs as a result of the structure of the linear receiver and can not be reduced substantially by simply optimizing the predetection filter. In addition, the performance of the nominal filter was found to be near optimal, indicating that the nominal filter is robust with respect to nonideal carrier recovery.

In Section 5.3, the effect of adjacent channel interference was included in the inner product matrix. The AOF was seen to provide significant performance gains over the nominal filter at practical error rates for $L = 3T$. The performance gain was seen to rapidly diminish as the observation interval was extended, suggesting that nominal filter design with $L \geq 5T$ may be acceptable in practice.

In Section 5.4, the transmitted signal was bandlimited prior to transmission, introducing intersymbol interference at the receiver. This effect was included in the inner product matrix. The AOF was seen to provide performance gains on the order of 1 dB over the nominal filter.

This thesis has presented a method for greatly increasing the usefulness of the AOLFP algorithm by allowing signal models that result in many more data signatures than was previously possible to be considered in the process of filter design. The algorithm can now be easily used to find the predetection filter response that provides the best attainable performance with a linear receiver for many communication signals suffering from a known source of distortion.

It is recognized that an AOF designed through the use of the extended AOLFP algorithm may not be implemented in practice, either because the filter response can not be easily realized in hardware or the improvement over the nominal filter does not justify its implementation. However, when the AOF can be easily realized, its implementation is worthwhile regardless of the magnitude of the performance improvement. In any case, the algorithm will always provide performance benchmarks that can be used to determine

whether the effect of a specific source of distortion can be neglected when designing a filter, and can be used to judge the performance of other filters designed through simpler approaches.

6.2 Suggestions for Further Work

As the approach to filter design presented in this thesis can be applied to virtually any binary communication signal, similar analysis could be performed for many other modulation techniques, for example other CPM signals. Alternate sources of distortion could also be considered, as well as variations in the channel characteristics such as fading. In addition, multiple sources of interference could be considered. For example, since FDMA systems commonly employ bandlimiting in each subchannel, determining the AOF for the combined effect of ACI and bandlimiting may be of interest.

The software implementation of the extended algorithm could also be improved by allowing the value of ϵ in Chapter 3 to vary during the determination of the initial approximate signal set. A more desirable approach would be to make n_{max} the only input parameter and have the program automatically increase the value of ϵ , starting over once the initial signal set contains n_{max} signal vectors. In this way, all intermediate signal sets are restricted to at most n_{max} vectors, and the initial signal set remains representative of the entire signal set. The possibility of finding a suitable combination of ϵ and n_{max} through trial and error would be removed.

APPENDIX

A COMPLETE SOFTWARE EXAMPLE USING THE AOLFP ALGORITHM

To clarify the approach to using the AOLFP algorithm discussed in Section 5.1, and to aid those interested in using the algorithm; the software, written in C, used to produce the results in Section 5.4 is included here. The portion of the program that implements the AOLFP algorithm begins with the calculation of the inner products on page 130. To apply the software to any other signal set, two subroutines specific to the application must be provided. First, "get1signal" must return a user defined data structure which contains the appropriate parameters describing an indexed data signature. Second, "get_ip" must return the inner product, given the data structures describing any two data signatures.

```

/* THIS PROGRAM COMPUTES THE INNER PRODUCT MATRIX FOR DUOBINARY MSK
BANDLIMITED BY A THIRD ORDER BUTTERWORTH FILTER PRIOR TO TRANSMISSION.
CALCULATIONS ARE DONE FOR A NORMALIZED BIT INTERVAL OF T=1.
THE AOF IS CALCULATED USING THE EXTENDED AOLFP ALGORITHM.
THE DEGREE OF BANDLIMITING IS DETERMINED BY THE PARAMETER K WHICH
DETERMINES THE 3 dB CUTOFF FREQUENCY OF THE FILTER AS GIVEN BY

```

CUTOFF FREQ. = $(K * \pi) / (2T)$

*/

```

#include <stdio.h>
#include <math.h>
FILE *outfile;
extern double fvscllr();
extern double DERFC();
extern fvsinit();
extern printf(),fprintf(),realloc(),calloc(),free();
#define K 1.5
#define delay 0.97 /* Group delay of the filter */
#define Ebt 0.9519 /* Average power after bandlimiting */
#define I 5.0 /* Observation interval duration */
#define d 5 /* number of intervals prior to t=0 */
#define f 4 /* number of T intervals after t=0 */
#define NUMWAVE 1024 /* total # of data signatures */
#define NSMAX 50 /* max. size of intermediate sets */
#define checklim 512
#define epsilon 0.0025
#define runlimit 10 /* max # intermediate solutions */
#define start_vect 0 /* any integer x: 0 <= x <= NS-1 */
#define errorlimit 1.0e-6 /* for numerical integration */
#define ACTIVELIM 32 /* max # active vectors */
#define TR_SIZE 20
#define EVEN 1
#define ODD 0
#define B 1.732050808
#define YES 1
#define NO 0
#define U 1
#define X 2
#define inner_prod(x,y) *(data[min(x,y)] + mag(x-(y)))

int *ISET,*ISETF,*prewave,*postwave,row_count=1,NS,feasible=NO;
double *A[ACTIVELIM],*data[NSMAX],dmin,dmin_im;
double *rho,*rho_im,*rvect,wm,wc,D;

```

```

double *CIPS[f+d+1];          /* Component inner products      */
struct transition
{
    int signal;
    struct transition *ptr0;
    struct transition *ptr1;
} ET[4],OT[4];

int intpow(x,y)
{
    int x,y;
    { int i,z=1;
      for(i=1; i <=y; ++i)
          z *= x;
      return(z);
    }
}

int min(x,y)
{
    int x,y;
    { int z;
      return(z = x<=y ? x:y);
    }
}

int mag(x)
{
    int x;
    { int z;
      return(z = x>0 ? x:-x);
    }
}

int even_or_odd(x)
{
    int x;
    { int z;
      z = x % 2;
      z = (z == 0) ? EVEN : ODD;
      return(z);
    }
}

/* FUNCTION TO FIND Dmin */

double get_dmin(x)
{
    double x[];
    { int i;
      double min = x[0];
      for(i=1; i<NS; ++i)
          if(x[i] < min)
              min = x[i];
      return(min);
    }
}

double get_y1(t)
{
    double t;
    { double y1,y2,y=0.0;
      if( (t>0.0) || (fabs(t) < 1.0e-8) )
          { y1 = (1-K)*exp(-wc*t) + (K-1)*cos(wm*t) + (K+1)*sin(wm*t);

```

```

    y1 /= (K*K+1);
    y2 = (K*K+K-1)*cos(wm*t) + (K-K*K+1)*sin(wm*t)
        + (1-K-K*K)*exp(-wc*t/2.0)*cos(B*wc*t/2.0)
        + (2*K*K*K-K*K-K-1)/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0);
    y2 /= D;
    y = y1 - y2; }
if( t >= 1.0 )
{ t -= 1.0;
  y1 = -(1+K)*exp(-wc*t) + (K+1)*cos(wm*t) + (1-K)*sin(wm*t);
  y1 /= (K*K + 1);
  y2 = (K-K*K+1)*cos(wm*t) + (1-K-K*K)*sin(wm*t)
      - (1+K-K*K)*exp(-wc*t/2.0)*cos(B*wc*t/2.0)
      + (2*K*K*K+K*K-K+1)/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0);
  y2 /= D;
  y -= (y1 - y2); }
return(y*K/sqrt(2.0));
}

double get_y2(t)
double t;
{ double y=0.0;
  if( (t>0.0) || (fabs(t) < 1.0e-8) )
  { y = 1.0 - exp(-wc*t) - 2/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0);
    if( t >= 1.0 )
    { t -= 1.0;
      y -= 1-exp(-wc*t) - 2/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0); } }
  return(y/sqrt(2.0));
}

double get_y3(t)
double t;
{ double y1,y2,y=0.0;
  if( (t>0.0) || (fabs(t) < 1.0e-8) )
  { y1 = -(K+1)*exp(-wc*t) + (K+1)*cos(wm*t) + (1-K)*sin(wm*t);
    y1 /= (K*K+1);
    y2 = (K-K*K+1)*cos(wm*t) + (1-K*K-K)*sin(wm*t)
        - (1-K*K+K)*exp(-wc*t/2.0)*cos(B*wc*t/2.0)
        + (2*K*K*K+K*K-K+1)/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0);
    y2 /= D;
    y = y1 - y2; }
  if( t >= 1.0 )
  { t -= 1.0;
    y1 = (1-K)*exp(-wc*t) + (K-1)*cos(wm*t) + (1+K)*sin(wm*t);
    y1 /= (K*K+1);
    y2 = (K*K+K-1)*cos(wm*t) + (1+K-K*K)*sin(wm*t)
        + (1-K-K*K)*exp(-wc*t/2.0)*cos(B*wc*t/2.0)
        + (2*K*K*K -K*K-K-1)/B*exp(-wc*t/2.0)*sin(B*wc*t/2.0);
    y2 /= D;
    y += (y1 - y2); }
  return(y*K/sqrt(2.0));
}

```



```
/* FUNCTION TO EVALUATE A DATA SIGNATURE */
```

```
double signal_val(r,t)
int r[];
double t;
{ int i;
double sum = 0.0;
for(i=0; i<f+d+1; ++i)
switch( mag(r[i]) )
{ case 1:
if( r[i] > 0 )
sum += get_y1(t-(i-d));
else
sum -= get_y1(t-(i-d));
break;
case 2:
if( r[i] > 0 )
sum += get_y2(t-(i-d));
else
sum -= get_y2(t-(i-d));
break;
case 3:
if( r[i] > 0 )
sum += get_y3(t-(i-d));
else
sum -= get_y3(t-(i-d));
break; }
return(sum);
}
```

```
/* FUNCTION TO CALCULATE THE COMPONENT INNER PRODUCTS */
```

```
double integrate(signal1,signal2,shift1,shift2,lim1,lim2)
int signal1,signal2,shift1,shift2;
double lim1,lim2;
{ int i,j,n,stop=NO;
double TR[TR_SIZE][2],inc,term1,term2,t,sum;
t = lim1;
TR[0][0] = 0.0;
for(i=0; i<=1; ++i)
{ switch (signal1)
{ case 1:
term1 = get_y1(t-shift1);
break;
case 2:
term1 = get_y2(t-shift1);
break;
case 3:
term1 = get_y3(t-shift1);
break; }
switch (signal2)
{ case 1:
term2 = get_y1(t-shift2);
break;
```

```

        case 2:
            term2 = get_y2(t-shift2);
            break;
        case 3:
            term2 = get_y3(t-shift2);
            break; }
    TR[0][0] += term1*term2;
    t = lim2; }
TR[0][0] *= (lim2-lim1)/2;
i = 0;
while( (!stop) && (i<TR_SIZE) )
{ for(j=0; j<i; ++j)
    TR[j][0] = TR[j][1];
    n = intpow(2,i+1);
    inc = (lim2-lim1)/n;
    TR[0][1] = TR[0][0]/2;
    sum = 0.0;
    for(j=1; j<n; j=j+2)
    { t = lim1 + j*inc;
        switch(signal1)
        { case 1:
            term1 = get_y1(t-shift1);
            break;
          case 2:
            term1 = get_y2(t-shift1);
            break;
          case 3:
            term1 = get_y3(t-shift1);
            break; }
        switch(signal2)
        { case 1:
            term2 = get_y1(t-shift2);
            break;
          case 2:
            term2 = get_y2(t-shift2);
            break;
          case 3:
            term2 = get_y3(t-shift2);
            break; }
        sum += term1*term2; }
    TR[0][1] += inc*sum;
    for(j=1; j<=i; ++j)
        TR[j][1] = (intpow(4,j)*TR[j-1][1]-TR[j-1][0])/(intpow(4,j)-1);
    TR[i+1][0] = (intpow(4,i+1)*TR[i][1]-TR[i][0])/(intpow(4,i+1)-1);
    stop = (fabs(TR[i+1][0] - TR[i][0]) < errorlimit) ? YES:NO;
    if( i<5 )
        stop =NO;
    ++i; }
return(TR[i][0]);
}

/* FUNCTION TO GENERATE 1 DATA SIGNATURE */

getiwave(x,a)

```

```

int x,a[];
{ int i,j,y;
  if( d!=0 )
  { y = (x < intpow(2,f+d)) ? x : x - intpow(2,f+d);
    for(i=0; i<d; ++i)
      a[i] = *(prewave + y/intpow(2,f)*d + i);
    a[i] = (a[i-1] > 0) ? 2 : -3;
    for(j=i+1; j<f+d+1; ++j)
      a[j] = *(postwave + y%intpow(2,f)*f + j-i-1);
    if( x >= intpow(2,f+d) )
    { for(i=0; i<d; ++i)
      { a[i] = -a[i];
        a[i] = (a[i] == 2) ? -3 : 2; } }
    else
    { if( x < intpow(2,f) )
      { a[0] = 2;
        y = x; }
      else
      { a[0] = -3;
        y = x - intpow(2,f); }
      for( i=0; i<f; ++i)
        a[i+1] = *(postwave + y*f + i); }
    return;
  }
}

```

/* FUNCTION TO PRINT PARAMETERS OF A DATA SIGNATURE */

```

printsig(index,a)
  int index,a[];
{ int i;
  fprintf(outfile,"r%4d =",index);
  for(i=0; i<f+d+1; ++i)
    fprintf(outfile," %2d",a[i]);
  fprintf(outfile,"\n");
}

```

```

get_waves(num_int,start,wave)
  int num_int,start,*wave;
{ int i,j,stepvalue,num;
  struct transition *step_ptr;
  num = intpow(2,num_int);
  for(i=0; i<num; ++i)
  { stepvalue = ((i & intpow(2,num_int-1)) == 0) ? 0:1;
    if( start == EVEN )
      step_ptr = & ET[stepvalue];
    else
      step_ptr = & OT[stepvalue];
    *(wave + i*num_int) = step_ptr->signal;
    for(j=1; j<num_int; ++j)
    { stepvalue = (( i & intpow(2,num_int-j-1)) == 0) ? 0:1;
      step_ptr = (stepvalue == 0) ? step_ptr->ptr0:step_ptr->ptr1;
      *(wave + i*num_int + j) = step_ptr->signal; } }
  return;
}

```

```
/* FUNCTION TO CALCULATE THE INNER PRODUCT BETWEEN TWO DATA SIGNATURES */
```

```
double get_ip(a,b)
  int a[],b[];
  { int i,j,m,n;
    double term,sum = 0.0;
    for(n=-d; n<=f; ++n)
      for(m=-d; m<=f; ++m)
        if( m<n )
          { term = *(CIPS[m+d] + 9*(n-m) + 3*(mag(b[m+d])-1)
                    + mag(a[n+d])-1);
            if( b[m+d]*a[n+d] > 0 )
              sum += term;
            else
              sum -= term; }
        else
          { term = *(CIPS[n+d] + 9*(m-n) + 3*(mag(a[n+d])-1)
                    + mag(b[m+d])-1);
            if( b[m+d]*a[n+d] > 0 )
              sum += term;
            else
              sum -= term; }
    return(sum);
  }
```

```
/* FUNCTION TO FIND AN IMPROVEMENT VECTOR FOR
   A NON FEASIBLE RECEIVER VECTOR -- SOLVE THEOREM 1 */
```

```
double invec(n,alpha)
  int n;
  double *alpha;
  { int i,j,k,a,b;
    double factor,*q[ACTIVELIN];

    /* Allocate memory for each row */
    for(i=0; i<n; ++i)
      if( (q[i] = calloc((n+1),sizeof(double))) == NULL )
        printf("NO MEMORY FOR GAUSSIAN ELIMINATION\n");

    /* LOAD THE MATRIX AI */
    for(i=0; i<n; ++i)
      { a=0;
        while( ISET[a] != i+1 )
          ++a;
        for(j=i; j<n; ++j)
          { b=0;
            while( ISET[b] != j+1 )
              ++b;
            *(q[i] + j) = inner_prod(a,b);
            if( i!=j )
              *(q[j] + i) = inner_prod(a,b); } }
    for(i=0; i<n; ++i)
```

```

        *(q[i] + n) = 1.0;

/* BEGIN GAUSSIAN ELIMINATION */
for(i=0; i<n-1; ++i)
    if( fabs(*(q[i] + i)) > 1.0e-9 )
        for(j=i+1; j<n; ++j)
            { factor = *(q[j] + i) / *(q[i] + i);
              for(k=i; k<=n; ++k)
                  *(q[j] + k) -= *(q[i] + k) * factor; }

/* SOLVE THE ALPHA VALUES */
for(i=n-1; i>=0; --i)
    { if( fabs(*(q[i] + i)) < 1.0e-9 )
      { if( fabs(*(q[i] + n)) > 1.0e-9 )
          fprintf(outfile,"zero row has nonzero constant\n");
        else
            *(alpha + i) = 0.0; }
      else
          { factor = *(q[i] + n);
            for(j=n-1; j>i; --j)
                factor -= *(alpha+j) * (*(q[i] + j));
            *(alpha + i) = factor / *(q[i] + i); } }
return(1.0);
}

/* FUNCTION TO CALCULATE THE ROTATION FACTOR LAMBDA */

double rotate(n,alpha)
int n;
double *alpha;
{ int i,new_av = -1;
  double lambda=0.0,delta,check;

/* Calculate the rotation factor lambda */
  for(i=0; i<NS; ++i)
      if( (ISET[i] == 0 ) && (fabs(rho[i]-dmin) > 1.0e-6) )
          if( rho_im[i] < dmin_im )
              { delta = (rho_im[i]-dmin_im)/(rho[i]-dmin);
                if( delta < lambda )
                    { lambda = delta;
                      new_av = i; } }
  delta = lambda;
  lambda = 1 / (1-lambda);

/* Perform the rotation --- calculate r = (1-lambda)*r + lambda*r' */
  if( !feasible )
      for(i=0; i<NS; ++i)
          { if( ISET[i] != 0 )
              rvect[i] = (1.0-lambda)*rvect[i] + *(alpha+ISET[i]-1)*lambda;
            else
              rvect[i] = (1.0-lambda) * rvect[i]; }
  else
      for(i=0; i<NS; ++i)

```

```

        { if( ISETF[i] != 0 )
          rvect[i] = (1.0-lambda)*rvect[i] + *(alpha+ISETF[i]-1)*lambda;
          else
            rvect[i] = (1.0-lambda) * rvect[i]; }
if( new_av == -1)
  return(1.0);
else /* ADD NEW AV AND LOOK FOR OTHERS */
{ ISET[new_av] = ++n;
  for(i=0; i<NS; ++i)
    if( (i != new_av) && (ISET[i] == 0) && (rho_im[i] < dmin_im) )
      { check = (rho_im[i]-dmin_im)/(rho[i]-dmin);
        if( fabs(check-delta) < 1.0e-6 )
          ISET[i] = ++n; }
  return(lambda); }
}

/* FUNCTION TO FIND AN IMPROVEMENT VECTOR FOR
   A FEASIBLE RECEIVER VECTOR -- SOLVE QUADRATIC PROGRAM */

double simplex(n,alpha,beta)
  int n;
  double *alpha,*beta;
{ int i,j,k,l,enter,leave,us,stop=NO;
  int bsc_pr=0,nonbsc_pr=0,stdform=YES;
  double z,ratio,factor,term,*q[ACTIVELIM+1],zfix=0.0,*top;
  struct
  { int status;
    int index;
  } base[ACTIVELIM];

/* ALLOCATE MEMORY FOR TABLEAUX */
  i = 2*intpow((n+1),2);
  if( (top = (double *) calloc(i,sizeof(double))) == NULL )
    printf(outfile,"NO MEMORY FOR SIMPLEX TABLEAUX\n");

/* Set up Pointers for each row in the Tableau */
  for(i=0; i<=n; ++i)
    q[i] = &top[i*(2*n+2)];

/* Adjust z to represent the true distance from x' */
  for(i=0; i<n; ++i)
  { j=0;
    while( ISET[j] != i+1 )
      ++j;
    zfix += *(alpha+i) * *(alpha+i) * inner_prod(j,j); }
  for(i=1; i<n; ++i)
  { k=0;
    while( ISET[k] != i )
      ++k;
    for(j=i+1; j<=n; ++j)
    { l=0;
      while( ISET[l] != j )
        ++l;
      zfix += 2.0 * *(alpha+i-1) * *(alpha+j-1)

```

```

        * inner_prod(k,1); } }

/* Set up the Tableau */

/* Initialize all Values to Zero */
for(i=0; i < 2*(n+1)*(n+1); ++i)
    *(top + i) = 0.0;

/* Load the -p Values and the -c Values */
for(i=1; i<=n; ++i)
{ *(q[i-1]) = -dmin;
  k=0;
  while( ISET[k] != i )
    ++k;
  for(j=1; j<=n; ++j)
  { l=0;
    while( ISET[l] !=j )
      ++l;
    *(q[i-1]+n+j) = - inner_prod(l,k); } }
for(i=n+1; i<= 2*n; ++i)
    *(q[n]+i) = - dmin;
*(q[n]+2*n+1) = 1.0;

/* Load the Identity Matrix */
for(i=0; i<n; ++i)
    *(q[i]+i+1) = 1.0;

/* Initialize the base array:
                                all u variables basic
                                all x variables nonbasic */
for(i=0; i<n; ++i)
{ base[i].status = U;
  base[i].index = i; }

/* Initiate the SIMPLEX Algorithm */

start:

/* Determine element to enter basis */
enter = -1;
if( stdform )
{ term = 0.0;
  for(j=0; j<n; ++j)
    if( (base[j].status == U) && ( *(q[j]) < 0)
        && (fabs(*(q[j])) > 1.0e-6) )
      if( *(q[j]) < term )
      { enter = base[j].index;
        term = *(q[j]); }
  if( enter == -1 )
    goto optimum_found; }

/* Determine column of element entering basis */
enter = stdform ? ( enter+n+1) : nonbsc_pr+1;

```

```

/* Determine the row of the element to leave the basis */
leave = -1;
/* Find the first basic x variable with a valid ratio */
i=0;
while( (!stop) && (i<n) )
{ if( (base[i].status == X) && (fabs(*(q[i]+enter)) > 1.0e-5) )
  { ratio = *(q[i]) / *(q[i]+enter);
    if( (ratio > 0.0) || (fabs(ratio) < 1.0e-5) )
      { term = ratio;
        leave = i;
        stop = YES; } }
  ++i; }
stop = NO;

/* Get ratios for other basic x variables and get min */
for(j=i; j<n; ++j)
  if( (base[j].status == X) && (fabs(*(q[j]+enter)) > 1.0e-5) )
    { ratio = *(q[j]) / *(q[j]+enter);
      if( (ratio > 0.0) && (ratio < term) )
        { term = ratio;
          leave = j; } }

/* Find the row corresponding to us */
if(stdform)
  us = enter - (n+1);
else
  us = bsc_pr;
i=0;
while( (base[i].status != U) || (base[i].index != us) )
  ++i;

/* Check for a tie with us */
if( fabs(*(q[i]+enter)) > 1.0e-5 )
  { ratio = *(q[i]) / *(q[i]+enter);
    if( leave != -1 )
      { if( (fabs(ratio-term) < 1.0e-8) || (ratio<term) && (ratio>0.0) )
        leave = i; }
    else
      if( (ratio > 0.0) || (fabs(ratio) < 1.0e-5) )
        leave = i; }
if( leave == -1 )
  goto optimum_found;

/* Perform the Pivoting */
factor = *(q[leave] + enter);
for(i=0; i<= 2*n+1; ++i)
  *(q[leave] + i) /= factor;
i=0;
while( i<=n )
  { if( i!=leave )
    { factor = *(q[i] + enter) / *(q[leave]+enter);
      for(j=0; j<=2*n+1; ++j)
        *(q[i]+j) -= *(q[leave] + j) * factor; }
    ++i; }

```



```

/* Determine:
    - basic and nonbasic pairs
    - if in standard form
    - adjustments to base */
if( stdform) /* an x enters basis */
{ if((base[leave].status == U) && (base[leave].index == enter-(n+1)))
    stdform = YES;
  else
  { stdform = NO;
    bsc_pr = enter - (n+1);
    nonbsc_pr = base[leave].index; }
    base[leave].status = X;
    base[leave].index = enter - (n+1); }
else /* a u enters basis */
{ if( base[leave].index == bsc_pr )
  { stdform = YES;
    base[leave].index = nonbsc_pr; }
  else
  { stdform = NO;
    i = nonbsc_pr;
    nonbsc_pr = base[leave].index;
    base[leave].index = i; }
    base[leave].status = U; }
goto start;

optimum_found:

if( stdform )
  z = *(q[n]);
else
{ i=0;
  while( base[i].index != us )
    ++i;
  factor = *(q[i++]);
  while( base[i].index != us )
    ++i;
  factor *= *(q[i]);
  z = *(q[n]) - factor; }

/* Determine the beta values */
for(i=0; i<n; ++i)
  *(beta + i) = 0.0;
for(i=0; i<n; ++i)
  if( base[i].status == X )
    *(beta+base[i].index) = *(q[i]);
free(top);
return(z - zfix);
}

main()
{ int post_num,pre_num;
  int a,b,i,j,k,l,stop,numset,shift,CSI=YES,rerun=NO,runcount=1;
  int r1[f+d+1],r2[f+d+1];

```

```

int *signals,numrx=0,*ptr,gotgamma=NO;
int numptrs=0,load_count=1,active_count=0;
double lambda,z,distance,min_d=100.0,max_d=0.0,EbNo,SNR,P,norm;
double *alpha,*load_ptr,*beta,*gamma;
double *loader,t,inc,lowlim,upplim,term,term1,term2,Ilo,Ihi;
struct aof
{ double coeff;
  int sig[f+d+1];
} *rx;

if( (outfile = fopen("result data","w")) == NULL )
  printf("NO DATAFILE");
fvsinit();
fprintf(outfile,"AOF RECEIVER CALCULATION FOR K= %.2f I= %.1fT\n",K,I);
fprintf(outfile," \nparameters : \n");
fprintf(outfile,"  f = %d  d = %d  Ebt = %.4f\n",f,d,Ebt);
fprintf(outfile,"  distance criterion = %f\n",epsilon);
fprintf(outfile,"  errorlimit = %.1e\n",errorlimit);
fprintf(outfile,"  delay = %.3f T\n",delay);
fprintf(outfile,"  starting vector = r%-3d\n",start_vect);

/* SET UP LINKED LIST DESCRIPTION OF STATE TRANSITIONS */
ET[0].signal = 1; ET[0].ptr0 = & OT[0]; ET[0].ptr1 = & OT[1];
ET[1].signal = 2; ET[1].ptr0 = & OT[0]; ET[1].ptr1 = & OT[1];
ET[2].signal = -2; ET[2].ptr0 = & OT[2]; ET[2].ptr1 = & OT[3];
ET[3].signal = -1; ET[3].ptr0 = & OT[2]; ET[3].ptr1 = & OT[3];
OT[0].signal = 2; OT[0].ptr0 = & ET[0]; OT[0].ptr1 = & ET[1];
OT[1].signal = 3; OT[1].ptr0 = & ET[2]; OT[1].ptr1 = & ET[3];
OT[2].signal = -2; OT[2].ptr0 = & ET[2]; OT[2].ptr1 = & ET[3];
OT[3].signal = -3; OT[3].ptr0 = & ET[0]; OT[3].ptr1 = & ET[1];
D = pow(K,4.0) - pow(K,2.0) + 1;
wm = pi/(2); wc = K*wm;

/* GENERATE THE COMPONENT INNER PRODUCTS */
Ilo = 1.5 + delay - I/2;
Ihi = 1.5 + delay + I/2;
for(i=0; i<f+d+1; ++i)
{ shift = i-d;
  numset = f+d+1-i;
  loader = CIPS[i] = (double*) calloc(9*numset,sizeof(double));
  for(j=0; j<numset; ++j)
  { if( (double) (shift+j) < Ihi )
    { upplim = Ihi;
      if( (double) (shift+j) < Ilo )
        lowlim = Ilo;
      else
        lowlim = (double) (shift + j);
      if( j!=0 )
        for(a=1; a<=3; ++a)
          for(b=1; b<=3; ++b)
            *loader++ = integrate(a,b,shift,shift+j,lowlim,upplim);
      else
        { for(b=1; b<=3; ++b)
          *loader++ = integrate(1,b,shift,shift,lowlim,upplim);
        }
    }
  }
}

```

```

        *loader++ = *(CIPS[i] + 1);
        for(b=2; b<=3; ++b)
            *loader++ = integrate(2,b,shift,shift,lowlim,upplim);
        *loader++ = *(CIPS[i] + 2);
        *loader++ = *(CIPS[i] + 5);
        *loader++ = integrate(3,3,shift,shift,lowlim,upplim); } }
    else
        for(k=0; k<9; ++k)
            *loader++ = 0.0;} }

get_signals:
/* GENERATE ALL POSSIBLE SIGNALS ON THE OBSERVATION INTERVAL */
signals = (int *) calloc(NSMAX,sizeof(int));
if( signals == NULL )
    { fprintf(outfile,"signals returned NULL\n");
      goto finish; }
pre_num = intpow(2,d);
post_num = intpow(2,f);

/* Generate Prewaves*/
prewave = (int *) calloc(d*pre_num,sizeof(int));
if( d!=0 )
    get_waves(d,even_or_odd(d+1),prewave);

/* Generate the Postwaves */
postwave = (int *) calloc(f*post_num,sizeof(int));
get_waves(f,EVEN,postwave);

/* DETERMINE THE INITIAL APPROXIMATION TO THE SIGNAL SET */
if( epsilon < 1.0e-7 )
    { NS = NUMWAVE;
      if( NSMAX != NUMWAVE )
          { printf("for epsilon = 0.0 NSMAX must equal NUMWAVE\n");
            goto finish; }
      for(i=0; i<NUMWAVE; ++i)
          *(signals+i) = i;
      goto innprod; }

NS = 1;
get1wave(0,r1);
*signals = 0;
i=1;
while( (i<NUMWAVE) && (NS<NSMAX) && (i<checklim) )
    { get1wave(i,r2);
      term = get_ip(r2,r2);
      if( i%100 == 0 )
          printf("checking signal %d NS = %d\n",i,NS);
      j = 0;
      stop = NO;
      while( (j<NS) && (!stop) )
          { get1wave(*(signals+j),r1);
            term1 = get_ip(r1,r2);
            term2 = get_ip(r1,r1);
            distance = term - 2*term1 + term2;

```

```

        min_d = (distance < min_d) ? distance : min_d;
        max_d = (distance > max_d) ? distance : max_d;
        if( fabs(distance) < epsilon )
        { stop = YES;
          if( term<term2 )
            *(signals+j) = i;    }
          ++j; }
        if( !stop )
          *(signals + NS++) = i;
          ++i; }
    if( NS > NSMAX )
    { printf("NSMAX IS TOO SMALL\n");
      goto finish; }
    signals = (int *) realloc(signals,NS*sizeof(int));
    fprintf(outfile,"minimum signal separation = %f\n",min_d);
    fprintf(outfile,"maximum signal separation = %f\n",max_d);

innprod:
    if( rerun )
    { fprintf(outfile,"\nrnrun # %d  NS = %d\n",runcount,NS);
      feasible = NO;
      active_count = 0;
      numrx = 0;
      CSI = YES;
      free(rx);
      free(ISET);free(rvect);
      rerun = NO; }

/* CALCULATE <ri,rj> for all {i,j} */
    for(i=0; i<NS; ++i)
    { getwave(*(signals+i),r1);
      data[i] = (double*)calloc((NS-i),sizeof(double));
      if( data[i] == NULL )
      { fprintf(outfile,"data pointer %d returned NULL\n",i);
        goto finish; }
      for(j=i; j<NS; ++j)
      { getwave(*(signals+j),r2);
        inner_prod(i,j) = get_ip(r1,r2); } }

    if( start_vect > NS )
    { fprintf(outfile,"INITIAL GUESS > NUMBER OF SIGNALS\n");
      goto finish; }

aolfp:
/* BEGIN THE AOLFP ALGORITHM */
    ISET = (int*) calloc(NS,sizeof(int));
    rho = (double*) calloc(NS,sizeof(double));
    rho_im = (double*) calloc(NS,sizeof(double));
    rvect = (double*) calloc(NS,sizeof(double));
    printf("Beginning AOLFP\n");

/* Set up initial Rx vector components */
    for(i=0; i<NS; ++i)
        rvect[i] = (i == start_vect) ? 1.0 : 0.0;

```

```

/* Set up initial inner product vector --- rho = <sj,r> for all j */
for(i=0; i<NS; ++i)
    rho[i] = inner_prod(start_vect,i);

/* Initialize the ISET array */
dmin = get_dmin(rho);
for(i=0; i<NS; ++i)
    if(fabs(rho[i] - dmin) < 1.0e-6 )
        ISET[i] = ++active_count;
    else
        ISET[i] = 0;

/* Check for Instant Solution */
if( fabs(dmin - rho[start_vect]) < 1.0e-6 )
{ fprintf(outfile,"BEHOLD !! AN INSTANT SOLUTION\n");
  fprintf(outfile,"THE AOF RECEIVER IS :\n\n");
  i = start_vect;
  fprintf(outfile,"%f  r%d  ",sqrt(1.0/inner_prod(i,i)),i);
  fprintf(outfile," r%d =",i);
  getiwave(*(signals+i),r1);
  for(j=0; j<f+d+1; ++j)
      fprintf(outfile," %d",r1[j]);
  fprintf(outfile,"\n");
  goto finish; }

/* FIND A FEASIBLE RECEIVER VECTOR */

get_feasible_soln:

    if( feasible )
        feasible = NO;

/* Get an Improvement Vector for the Nonfeasible receiver Vector */
alpha = (double*) calloc(active_count,sizeof(double));
dmin_im = imvec(active_count,alpha);

/* Calculate < sj,r' > for all j -- r' an improvement vector */
for(i=0; i<NS; ++i)
    rho_im[i] = 0.0;
for(k=0; k<NS; ++k)
{ if( ISET[k] == 0 )
    for(i=0; i<active_count; ++i)
    { j=0;
      while( ISET[j] != (i+1) )
          ++j;
      rho_im[k] += *(alpha+i) * inner_prod(j,k); }
    else
        rho_im[k] = dmin_im; }

do_rotate:

/* Rotate towards the improvement vector r' */
lambda = rotate(active_count,alpha);

```

```

        if( feasible )
            free(ISETF);

/* Get new value of active_count */
    for(i=0; i<NS; ++i)
        active_count = ISET[i] > active_count ? ISET[i] : active_count;
    if( active_count > ACTIVELIM )
        { printf("#active vectors exceeds ACTIVELIM\n");
          goto finish; }

/* Calculate <si,r> for all i -- r = receiver vector after rotation */
    for(i=0; i<NS; ++i)
        rho[i] = (1.0-lambda) * rho[i] + lambda * rho_im[i];

/* Get new dmin */
    dmin = get_dmin(rho);

/* Check Feasibility Criterion : lambda = 1 */
    if(fabs(1.0-lambda) > 1.0e-6 )
        { free(alpha);
          goto get_feasible_soln; }
    feasible = YES;

/* Check Optimality Criterion : r' has Positive Coefficients */
    i=0;
    while( ( i<NS ) && (( *(rvect+i) > 0.0 )
                    || (fabs( *(rvect+i)) < 1.0e-5)) )
        ++i;
    if( i == NS )
        goto solution_found;

/* Get an Improvement Vector for the Feasible Receiver Vector */
    beta = (double *) calloc(active_count,sizeof(double));
    z = simplex(active_count,alpha,beta);
    for(i=0; i<active_count; ++i)
        if( ( *(beta+i) < 0 ) && ( fabs(*(beta+i)) > 1.0e-6 ) )
            { printf("simplex has produced a negative beta\n");
              goto finish; }

/* Check Optimality Criterion : Minimum z = 0 */
    if( fabs(z) < 1.0e-10)
        goto solution_found;

/* Calculate <si,r'> for all i -- r' an improvement vector */
    for(j=0; j<NS; ++j)
        rho_im[j] = 0.0;
    for(j=0; j<NS; ++j)
        for(i=0; i<active_count; ++i)
            { k=0;
              while( ISET[k] != (i+1) )
                  ++k;
              rho_im[j] += *(beta+i) * inner_prod(j,k); }

/* Determine the new dmin' */

```

```

i=0;
while( ISET[i] == 0 )
    ++i;
dmin_im = rho_im[i];
for(j=i+1; j<NS; ++j)
    if( (ISET[j] != 0) && (rho_im[j] < dmin_im) )
        dmin_im = rho_im[j];

if( (ISETF=(int *) calloc(NS,sizeof(int))) == NULL )
    { fprintf(outfile,"ISETF returned NULL\n");
      goto finish; }
for(i=0; i<NS; ++i)
    *(ISETF+i) = *(ISET+i);
for(i=0; i<active_count; ++i)
    *(alpha+i) = *(beta+i);
free(beta);

/* Reset ISET to active vectors of both r' and rf */
j=0;
for(i=0; i<NS; ++i)
    if( ISET[i] != 0 )
        { if( fabs(rho_im[i]-dmin_im) < 1.0e-6 )
            ISET[i] = ++j;
          else
            ISET[i] = 0; }
active_count = j;
goto do_rotate;

solution_found:
free(rho);free(rho_im);
free(alpha);

/* Normalize the Receiver Vector */
z=0;
for(i=0; i<NS; ++i)
    z += rvect[i];
if( z<0.0 )
    { fprintf(outfile,"NORMALIZATION FACTOR IS NEGATIVE\n\n");
      fprintf(outfile,"SOMETHING IS SERIOUSLY WRONG\n\n"); }
else
    for(i=0; i<NS; ++i)
        rvect[i] /= sqrt(z);
fprintf(outfile,"\nTHE OPTIMUM RECEIVER VECTOR COMPONENTS ARE :\n\n");
for(i=0; i<NS; ++i)
    if( fabs(rvect[i]) > 1.0e-9 )
        { fprintf(outfile,"%10f  r%-3d  r%-3d = ",rvect[i],i,i);
          get1wave(*(signals+i),r1);
          for(j=0; j<i+d+1; ++j)
              fprintf(outfile," %2d ",r1[j]);
          fprintf(outfile,"\n"); }
fprintf(outfile,"\nACTIVE VECTOR SET :\n");
for(i=0; i<NS ; ++i)
    if( ISET[i] != 0 )
        { fprintf(outfile,"r%-3d =",i);

```

```

        getwave(*(signals+i),r1);
        for(j=0; j<f+d+1; ++j)
            fprintf(outfile," %2d",r1[j]);
        fprintf(outfile,"\n"); }
    norm = 1.0/sqrt(z);

/* STORE RX */
    rx = (struct aof *) calloc(active_count,sizeof(struct aof));
    if( rx==NULL )
        { fprintf("rx returned NULL\n");
          goto finish; }
    for(i=0; i<NS; ++i)
        if( fabs(rvect[i]) > 1.0e-6 )
            { (rx+numrx)->coeff = rvect[i];
              getwave(*(signals+i),r1);
              for(k=0; k<f+d+1; ++k)
                  (rx+numrx)->sig[k] = r1[k];
              ++numrx; }
    rx = (struct aof *) realloc(rx,numrx*sizeof(struct aof));

/* Calculate the AOF Impulse Response */
    fprintf(outfile,"\nAOF Response :\n");
    t = Ilo;
    inc = 0.10;
    while( t <= Ihi )
        { z = 0.0;
          for(i=0; i<numrx; ++i)
              { for(k=0; k<f+d+1; ++k)
                  r1[k] = (rx+i)->sig[k];
                z += (rx+i)->coeff * signal_val(r1,t); }
          fprintf(outfile,"rx(%6.3f T) = %7.4f\n",t,z);
          t +=inc; }

/* CALCULATE THE AOF PERFORMANCE */

/* CALCULATE THE GAMMA VALUES */
    if( !gotgamma )
        { gamma = (double *) calloc(NUMWAVE,sizeof(double));
          gotgamma = YES; }
    if( gamma == NULL )
        printf("gamma pointer returned NULL\n");
    for(i=0; i<NUMWAVE; ++i)
        { getwave(i,r1);
          *(gamma+i) = 0.0;
          for(j=0; j<numrx; ++j)
              { for(k=0; k<f+d+1; ++k)
                  r2[k] = (rx+j)->sig[k];
                *(gamma+i) += (rx+j)->coeff * get_ip(r1,r2); } }
    z = *gamma;
    for(i=1; i<NUMWAVE; ++i)
        if( *(gamma+i) < z )
            z = *(gamma + i);
    term = Ebt/z/z;
    fprintf(outfile,"FACTOR TO MATCH MSK = %f dB\n",10*log10(term));

```



```

fprintf(outfile, "\nERROR RATE PERFORMANCE : \n\n");
fprintf(outfile, "SNR (dB)      Pe\n");
SNR=0.0; P=0.0;
while( SNR < 19.0 )
{ EbNo = pow(10.0, (SNR/10.0));
  for(i=0; i<NUMWAVE; ++i)
  { term = *(gamma+i)*sqrt(EbNo/Ebt);
    if( term<13.0 )
      P += 0.5*fvscllr(DERFC,1,&term);
    else
      P +=0.0; }
  P /= NUMWAVE;
  fprintf(outfile, "%6.2f    %.4e\n", SNR, P);
  P = 0.0;
  SNR += 1.0; }

/* TEST IF OPTIMAL FOR ENTIRE SIGNAL SET */
if( z<norm && fabs(z-norm)>1.0e-6 && runcount<=runlimit )
{ rerun = YES;
  runcount++;
  for(i=0; i<NS; ++i)
    free(data[i]);
  signals = (int *) realloc(signals, NSMAX*sizeof(int));
  for(i=0; i<NS; ++i)
    if( ISET[i]!=0 )
      ISET[i] = *(signals+i);
  ptr = signals;
  for(i=0; i<NS; ++i)
    if( ISET[i]!=0 )
      *ptr++ = ISET[i];
  NS = active_count;
  for(i=0; i<NUMWAVE; ++i)
    if( *(gamma+i)<norm && fabs(*(gamma+i)-norm)>1.0e-6 && NS<NSMAX )
      { ++NS;
        *ptr++ = i; }
  if( NS!=NSMAX )
    signals = (int *) realloc(signals, NS*sizeof(int));
  goto innprod; }
finish:
;
}

```

REFERENCES

- [1] P. Galko and S. Pasupathy, "Linear receivers for correlatively coded MSK", *IEEE Trans. Commun.*, vol. COM-33, pp. 338-347, April 1985.
- [2] C.E. Sundberg, "Continuous phase modulation", *IEEE Communications Magazine*, pp. 25-38, April 1986.
- [3] E.A. Lee and D.G. Messerschmitt, *Digital Communication*. Boston: Kluwer Acad. Pub., 1988.
- [4] F. de Jager and C.B. Dekker, "Tamed frequency modulation, a novel method to achieve spectrum economy in digital transmission", *IEEE Trans. Commun.*, vol. COM-26, pp. 534-542, May 1978.
- [5] D.J. Vaisey and P.J. McLane, "Passive arm filters in I and Q receivers for MSK-type continuous phase modulations", *IEEE Trans. Commun.*, vol. COM-31, pp. 1235-1240, November 1983.
- [6] P. Galko and S. Pasupathy, "Optimization of linear receivers for data communication signals", *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 79-92, Jan. 1988.
- [7] C. van de Panne and A. Whinston, "The simplex and the dual method for quadratic programming", *Operations Research Quarterly*, vol. 15, pp. 355-388, Dec. 1964.
- [8] S. Pasupathy, "Minimum shift keying: a spectrally efficient modulation", *IEEE Communications Magazine*, pp. 14-22, July 1979.
- [9] F. Amoroso, "Pulse and spectrum manipulation in the minimum-shift keying (MSK) format", *IEEE Trans. Commun.*, vol. COM-24, pp. 381-384, Mar. 1976.
- [10] S.A. Rhodes, "Effect of noisy phase reference on coherent detection of offset-QPSK signals", *IEEE Trans. Commun.*, vol. COM-22, pp. 1046-1055, August 1974.
- [11] B.P. Lathi, *Modern Digital and Analog Communication Systems*. New York: Holt, Rinehart and Winston, 1983.
- [12] R. Rhodes, S. Wilson and A. Svensson, "MSK-type reception of continuous phase

- modulation: cochannel and adjacent channel interference", *IEEE Trans. Commun.*, vol. COM-35, pp. 185-193, Feb. 1987.
- [13] V. Varma and S. Gupta, "Performance of partial response CPM in the presence of adjacent channel interference and Gaussian noise", *IEEE Trans. Commun.*, vol. COM-34, pp. 1123-1131, Nov. 1986.
- [14] M. Gavish and I. Kalet, "Mean-square crosstalk in tamed frequency modulation", *IEEE Trans. Commun.*, vol. COM-35, pp. 455-458, April 1987.
- [15] P. Galko and S. Pasupathy, "Power spectra of correlatively coded CPFSK", in *Int. Commun. Conf. Rec.*, pp. 45.2-45.2.5, Boston, June 1979.