REDUCED-COMPLEXITY CODED GMSK SYSTEMS USING ITERATIVE DECODING

by

Li-Wei Chen

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Li-Wei Chen 2000
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

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

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

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

0-612-53427-8
Reduced-Complexity Coded GMSK Systems Using Iterative Decoding

by

Li-Wei Chen

Master of Applied Science Thesis
Graduate Department of Electrical and Computer Engineering
University of Toronto
2000

Abstract

In this thesis, a number of coded Gaussian minimum-shift keying (GMSK) transceiver systems using the Laurent representation and iterative decoding are proposed and evaluated. First, a 4-state reduced-complexity linear demodulator is shown to perform close to the optimal 16-state demodulator for $BT=0.3$. This reduced-complexity receiver is used with a coded GMSK system, and a model for considering the code and modulation memory together as a serially concatenated code is proposed. Transceiver systems based on this model have no bandwidth expansion above the original coded GMSK, and are shown to have significantly better performance. The effect of SISO parameters on performance are examined. Results are presented for both additive white Gaussian noise (AWGN) and Rayleigh fading channels. The causes of performance loss in a Rayleigh fading environment are identified, and methods for making the system more robust to fading using an adaptive matched filter and coding are shown. Simulation results are presented demonstrating that the performance of the proposed adaptive receiver is superior to the non-adaptive receiver in a Rayleigh channel.
Acknowledgements

First, I would like to express my sincerest appreciation to my thesis advisor, Prof. S. Pasupathy, for the support, assistance, and encouragement he has shown me during my stay here at U of T. Working with him has been a wonderfully enjoyable and educational experience.

I would also like to thank Prof. F. Kschischang for his assistance with some of the turbo coding literature, and for providing me with the use of his computer cluster for simulation purposes.

I gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council in performing this research.

I'd also like to thank my friends in the Communications Group who have shared these last few years with me. Special thanks to Eddy, whose well-stocked room at home is probably responsible for adding an extra month or two to my stay here :)

Finally, and most importantly, I want to thank my father, for instilling in me a love of learning; my mother, for believing in me; and my brother, for reminding me that there is life outside of school. This work has grown out of your love and support.
# Table of Contents

Abstract .................................................................................................................. ii

Acknowledgements .................................................................................................... iii

1. Introduction ........................................................................................................... 1
   1.1 Organization of Thesis ...................................................................................... 3

2. On GMSK and Concatenated Codes ..................................................................... 4
   2.1 GMSK and the Laurent Representation .......................................................... 4
      2.1.1 GMSK as a CPM Scheme ........................................................................... 4
      2.1.2 The Laurent Representation of GMSK ...................................................... 7
   2.2 Serial Concatenated Codes .............................................................................. 10
      2.2.1 Some Background on Concatenated Coding ............................................ 10
      2.2.2 Decoding Serially Concatenated Codes .................................................. 10
      2.2.3 The Maximum \textit{A Posteriori} Probability (MAP) Algorithm ................... 15
      2.2.4 The max-log-MAP Algorithm ................................................................... 18
      2.2.5 The \textit{A Priori} Soft-Output Viterbi Algorithm (APRI-SOVA) ................... 20
3. Reduced-Complexity Receiver Design

3.1 The Laurent Linear Receiver

3.1.1 Construction of the Laurent Representation

3.1.2 The Laurent-Based Demodulator

3.2 Reduced-Complexity Receiver Performance

3.3 Precoded GMSK

3.3.1 ‘Gray Coding’ Property of the Laurent PAM Process

3.3.2 The Role of Precoding

3.3.3 Simulation Results for Precoded GMSK

4. Iterative Decoding of Coded GMSK

4.1 System Model

4.1.1 Transmitter Models

4.1.2 Demodulator Model

4.1.3 Choice of Outer Code

4.2 Simulation Results

4.2.1 Performance of Coded GMSK in AWGN

4.2.2 Performance Dependence on Block Size and Iterations

4.2.3 Performance Dependence on SISO Algorithm

4.2.4 Performance Dependence on Outer Code

5. Coded GMSK in a Wireless Environment

5.1 GMSK with Multipath Fading
Appendix C: MAP Algorithm Derivation................................................................. 87

C1. Description of Model .................................................................................. 87
   C1.1 The State Transition Probability ............................................................ 88
   C1.2 The Forward Recursion ......................................................................... 89
   C1.3 The Backward Recursion ...................................................................... 89
   C1.4 Calculating $\varphi_z$ ............................................................................. 90
   C1.5 Determining the Output Information Symbol Likelihood ...................... 90
          C1.5 Determining the Output Codeword Symbol Likelihood ................. 92

References ......................................................................................................... 94
List of Figures

Figure 2.1: GMSK frequency (left) and phase (right) pulses for BT = 0.3............... 7
Figure 2.2: Laurent model for a GMSK modulator.............................................. 9
Figure 2.3: Laurent-based demodulator............................................................... 9
Figure 2.4: An encoder for a parallel concatenated code................................. 11
Figure 2.5: An encoder for a serially concatenated code................................. 11
Figure 2.6: The SISO decoder block................................................................. 12
Figure 2.7: Flow of information in iterative SCC decoding................................. 13
Figure 2.8: The APRI-VA Metric.................................................................. 22
Figure 3.1: The Laurent $h_0(t)$ pulse............................................................. 31
Figure 3.2: Linear modulator structure (the $\Sigma$ denotes conventional addition). 32
Figure 3.3: The product differential encoder version of the Laurent modulator...... 33
Figure 3.4: Laurent receiver based on a one-pulse model................................... 36
Figure 3.5: Performance of the 4-state Laurent receiver vs. the optimum receiver... 38
Figure 3.6: The ISI region for Laurent GMSK, BT=0.3, delay 2......................... 42
Figure 3.7: Using precoding to remove the modulation memory........................ 43
Figure 3.8: A feed-forward encoder structure................................................... 43
Figure 3.9: Precoded GMSK, BT=0.3................................................................. 44
Figure 3.10: Precoded GMSK, various BT and L values................................. 45
Figure 4.1: Conventional (benchmark) system............................................... 47
Figure 4.2: SCC system using GMSK memory as inner code............................. 47
Figure 4.3: Proposed receiver for the SCC system........................................ 49
Figure 4.4: (23, 35) code structure. (23, 35) in binary = (10011, 11101) in binary.... 51
Figure 4.5: Performance in AWGN. 10k blocks, 10 iterations............................. 53
Figure 4.6: Performance dependence on number of iterations. 10k blocks............. 53
Figure 4.7: Performance dependence on block size. 10 iterations.......................... 54
Figure 4.8: Performance dependence on algorithm. 10k blocks, 10 iterations......... 55
Figure 4.9: Dependence on outer code. 10k blocks, 10 iterations.......................... 56
Figure 5.1: The Clarke model fade power spectrum........................................ 59
Figure 5.2: Coded GMSK in a fading vs. AWGN channel.................................. 61
Figure 5.3: Relative performance of cases #1-3 in a fading channel...................... 61
Figure 5.4: Uncoded GMSK in AWGN and fading channels................................. 67
Figure 5.5: Simulation of the adaptive receiver incorporating estimation information... 69
Figure 5.6: Alternate implementation of the adaptive receiver simulation.............. 69
Figure 5.7: Performance of adaptive receiver in a fading channel........................ 70
Figure 5.9: Adaptive receiver performance with suboptimal estimation.................. 73
Figure 5.10: Relationship between estimator MSE and decoder MSD..................... 75
Figure B1: Clarke fading channel module block diagram................................... 82
List of Tables

Table 2.1: SISO input and output quantities................................................................. 13
Table 2.2: Relative performance and complexity of SISO algorithms.......................... 15
Table 3.1: All possible pulse shapes for BT=0.3 .......................................................... 35
Table 4.1: Outer codes considered in this thesis.......................................................... 51
Table 4.2: Various cases studied under simulation....................................................... 55
Table 5.1: Clarke model parameters used in this thesis.............................................. 60
Chapter 1

Introduction

It is indisputable that communication systems have become an essential part of our everyday life. Digital networks now carry a wide variety of voice and data services for an ever-growing number of users. As a result, the spectrum available to support these services has become increasingly congested. This is especially true in wireless systems, where signals are carried over the air and the total amount of spectrum cannot be increased by laying more cabling as in wireline systems. Therefore the spectral efficiency of transceiver systems becomes a very important issue.

Due to its high spectral efficiency, Gaussian minimum-shift keying (GMSK) has become a popular modulation scheme for wireless applications. The GSM cellular standard [22] is the most well-known use of GMSK. Other applications include the Cellular Digital Packet Data (CDPD) [23] and Mobitex™ [24] standards. GMSK-modulated signals obtain spectral compactness by imposing certain smoothness constraints on the transmitted waveform. The costs of this spectral efficiency derive from the fact that these constraints add memory to the system, which not only increases the complexity of the receiver, but also degrades the bit error performance of the overall system. The goal of this thesis is to attempt to overcome some of the performance loss associated with the modulation memory by employing a coded modulation system with reduced complexity.

To reduce the complexity of demodulation, the Laurent representation first proposed by Pierre Laurent [2] will be employed. The Laurent representation allows for the design of
suboptimal reduced-complexity receivers for continuous-phase modulation (CPM) schemes by lowering the number of trellis states required. It also has the additional property of extracting the modulation memory as a recursive rate 1 code, which will prove useful later in the design of the iterative receiver. The challenges involved in designing a good receiver using the Laurent representation will be investigated.

To improve the performance of the overall system, principles from concatenated coding will be used. Our working assumption is that all practical systems will use an error-correcting code. In this thesis, we restrict our investigation to convolutional codes, one of the most popular types of error-correcting code. Typically, if a performance gain is required, the system must increase the strength of the code, which typically results in bandwidth expansion and/or an increase in decoder complexity. The recent success of turbo codes [5] and other concatenated coding schemes suggests that concepts developed for concatenated coding systems may be applicable here. As an alternative to going to a lower-rate error-correcting code, we will attempt to make use of the GMSK memory itself, which is already present, as the inner code of a serially-concatenated code and employ iterative decoding in the receiver. This would have the advantage of not imposing any additional bandwidth expansion above the existing coded system.

Finally, since we are interested in considering wireless applications, the performance of any proposed modifications must be investigated in a fading environment. It is expected that the harsher conditions of a fading environment will degrade performance relative to AWGN. This thesis will also investigate methods of mitigating the effects of fading for a coded GMSK system.
1.1 Organization of Thesis

Chapter 2 provides some background information and a brief tutorial on GMSK, the Laurent representation, concatenated coding, and iterative decoding. Chapter 3 looks at the design and optimization of a reduced-complexity receiver based on the Laurent representation. Chapter 4 investigates the role of concatenated codes and iterative decoding in improving the performance of the system. Chapter 5 examines the performance of the coded GMSK transceiver system proposed in Chapter 4 under fading channel conditions and proposes methods of compensating for the detrimental effects of fading. Chapter 6 summarizes the results of this thesis and suggests avenues for further research.
Chapter 2

On GMSK and Concatenated Codes

In this chapter, a review of GMSK and the Laurent representation will be given. The method of serial concatenation of codes to create stronger codes, as well as applicable iterative decoding techniques, will also be discussed.

2.1 GMSK and the Laurent Representation

2.1.1 GMSK as a CPM Scheme

Gaussian minimum-shift keying is a special case of a more generic class of modulation schemes known as continuous phase modulation (CPM). In CPM schemes, the signal envelope is kept constant and the phase varies in a continuous manner. This ensures that CPM signals do not have the high-frequency components associated with sharp changes in the signal envelope and allows for more compact spectra. Any CPM signal $s(t)$ can be written as

$$s(t) = \sqrt{\frac{2E_s}{T}} \cos[\omega_c t + \phi(t, \alpha)] ,$$

(2.1)
where $E_b$ is the energy per symbol interval, $T$ is the duration of the symbol interval, $\omega_c$ is the carrier frequency, and $\varphi(t, \alpha)$ is the "phase function" responsible for mapping the input sequence to a corresponding phase waveform. The term $\alpha = \{ \alpha_i \}$ is the input sequence taken from the M-ary alphabet $\pm 1, \pm 3, \ldots, \pm M - 1$. For convenience the focus here will be on the binary case, $\alpha_i \in \{ \pm 1 \}$. The generalization to M-ary signaling is straightforward and discussed in detail in [1].

The "continuous phase" constraint in CPM requires that the phase function maintain a continuous amplitude. In general the phase function is given by

$$\phi(t, \alpha) = 2\pi \sum_{n=0}^{N} \alpha_n h_n q(t - nT) ,$$

(2.1)

where $h_n$ is the (possibly time-varying) modulation index, and $q(t)$ is the phase pulse. Thus the phase function is seen to be essentially a pulse-amplitude modulation (PAM) process with impulse response $q(t)$. By convention the phase pulse $q(t)$ is typically specified in terms of a normalized, time-limited frequency pulse $f(t)$ of duration $LT$ such that:

$$q(t) = \begin{cases} 0 & \text{if } t < 0 , \\ \int_0^t f(\tau) d\tau & \text{if } 0 < t < LT , \\ \frac{\gamma}{2} & \text{if } t > LT . \end{cases}$$

By convention, the duration term ($LT$) is specified in terms of the bit duration $T$, and identifies the number of bit durations over which the frequency pulse $f(t)$ is non-zero. For CPM schemes where $f(t)$ has infinite duration, the frequency pulse will be
truncated to some finite duration which contains a large portion of the pulse energy, and the truncated duration becomes the duration $LT$.

For GMSK, $h = \frac{1}{2}$, and the frequency pulse is

$$f(t) = \frac{1}{2T} \left[ Q\left(2\pi B \frac{t - \frac{T}{2}}{\sqrt{\ln 2}}\right) - Q\left(2\pi B \frac{t + \frac{T}{2}}{\sqrt{\ln 2}}\right) \right].$$  \hspace{1cm} (2.2)

$Q(t)$ is the familiar Gaussian $Q$-function described by

$$Q(t) = \int_t^\infty \frac{1}{\sqrt{2\pi}} e^{-\tau^2/2} d\tau.$$

$B$ is a parameter in GMSK which controls the amount of bandwidth used as well as the severity of the intersymbol interference. Again, by convention, the $B$ parameter is normally expressed in terms of the inverse of the bit duration $T$; therefore designers usually refer to the bandwidth-symbol duration product $BT$ rather than to $B$ itself.

Qualitatively, a large value of $BT$ results in less inter-symbol interference (ISI) for the PAM process given by equation 2.1 at the cost of more bandwidth. Readers interested in the technical details of this tradeoff are referred to [1], which contains a more detailed discussion of this well-known result. In this thesis, a $BT$ of 0.3 (the value used by the GSM standard) will be adopted. The frequency and phase pulses are shown in Figure 2.1. Note that technically the frequency pulse has infinite duration; in practice, for $BT = 0.3$ it is truncated to a duration of $L = 3T$, as most of the pulse energy is concentrated within that duration.
2.1.2 The Laurent Representation of GMSK

In 1986 Pierre A. Laurent proposed the Laurent representation [2] as a way of looking at CPM signals as a sum of $2^{L-1}$ PAM signals, where $L$ is the frequency pulse duration described in the preceding section. The benefits of doing so are twofold: first, this new representation achieves energy compaction, and second, it paves the way for the design of reduced-state demodulators. Laurent showed that many of these PAM signals contained a negligible amount of signal energy, and that reduced-complexity receivers with a fewer number of trellis states could be designed by only demodulating those signals into which most of the signal energy had been compacted.

For binary GMSK, it can be shown [2] [3] that the Laurent representation at baseband is given by

$$s(t) = \sqrt{\frac{2E_b}{T}} \sum_{k=0}^{2^{L-1}-1} \sum_{n=0}^{N-1} a_{k,n} h_k(t-nT),$$

(2.3)
where:

\[ a_{0,n} = a_{0,n-1} j^{\alpha_n} \]

\[ a_{k,n} = a_{0,n-L} \prod_{i \in I_k} j^{\alpha_{n-i}} \]

The various \( h_k(t) \) terms are the impulse responses of the \( 2^{L-1} \) Laurent PAM processes, where \( k \) takes on values from the set of indices labeling the PAM processes. Note that in general these terms (and hence the PAM processes) are not orthogonal. There is no closed-form expression of the \( h_k(t) \) terms, and hence these terms must be analytically determined on a case-by-case basis. For our chosen modulation scheme (GMSK with \( L = 3T \)), the main pulse \( h_0(t) \) is derived in Appendix A.

Note that new input sequences \( \{a_{k,n}\} \) have been defined from the old sequence \( \{\alpha_n\} \). These new sequences take on complex values from the set \( \{\pm 1, \pm j\} \), and the \( k^\text{th} \) sequence modulates the \( k^\text{th} \) PAM signal. The form of the first new input sequence, \( \{a_{0,n}\} \), is the same for all forms of CPM. The equations governing the remaining input sequences all have a similar form, but the set \( I_k \) over which the product is taken does not have a general expression valid for all CPM schemes. Therefore, \( I_k \) must be determined on a case-by-case basis for the particular CPM scheme of interest. For some examples of how \( I_k \) may be determined for a few specific cases, readers are referred to [2] and the appendix of [3].

A pictorial representation of the Laurent model for a GMSK modulator is given in Figure 2.2, and Figure 2.3 shows the optimum receiver for this signal model. The percentage of signal power in each PAM signal must also be evaluated on a case-by-case basis for each value of BT. However, for typical values, [2], [3], and [4] show that most (~99%) of the signal energy lies in first component \( h_0(t) \), with most of the remainder in \( h_1(t) \) (~0.8%). Therefore, to keep the number of trellis states low in practical receivers, the
MLSE block of Figure 2.3 is often fed with the results of only the first one or two matched filters.

Figure 2.2: Laurent model for a GMSK modulator.

Figure 2.3: Laurent-based demodulator.
2.2 Serial Concatenated Codes

2.2.1 Some Background on Concatenated Coding

Adding an error-correcting code to a communication system typically involves striking a balance between the strength (error-correcting capability) of the code chosen, and the complexity involved in encoding and decoding it. For example, stronger convolutional codes tend to have longer constraint lengths, but the complexity of maximum-likelihood decoding increases exponentially with constraint length. In 1993, Berrou, Glavieux, and Thitimajshima proposed “turbo codes” [5] as a method for overcoming this problem through the iterative soft-decision decoding of concatenated codes.

There are two general classes of concatenated codes: parallel concatenated codes (PCCs), and serial concatenated codes (SCCs). Encoders for PCCs and SCCs are shown in Figures 2.4 and 2.5, respectively. Subsequent research in the area of concatenated coding [6] has shown that both types of concatenation can yield the characteristic “turbo”-like gains in performance. Hybrid SCC/PCC schemes have also proven fruitful [28]. In this thesis we will focus on serially concatenated coding schemes, as the SCC encoding structure lends itself nicely to the problem of combining the modulation memory with error correction. This appears to be a promising approach since it is known that even rate 1 recursive inner encoders can show “turbo”-like performance [24], and others have had good results when using the modulation scheme as the inner code [11] [14].

2.2.2 Decoding Serially Concatenated Codes

An SCC forms a strong overall code by concatenating two component codes in a serial fashion, separated by an interleaver, as shown in Figure 2.5. Maximum-likelihood
decoding of an SCC is prohibitively difficult since the trellis representation of the overall code is very complex. The “turbo” decoding approach is to instead decode the component codes cooperatively using separate devices known as soft-input soft-output (SISO) decoders [6] which exchange soft (reliability) information about the bits.

The term “SISO decoder” refers to an algorithm which accepts information and codeword symbol likelihood information, processes this information based on the code constraints, and outputs refined estimates of the likelihoods. Thus a SISO decoder is a four-port
device with two inputs and two outputs (Figure 2.6). The input and output quantities are summarized in Table 2.1.

The input likelihoods represent the probability distributions (called the soft information in the coding literature) of the information and coded symbols before processing. The SISO then generates updated output likelihoods for both information and coded symbols according to the chosen algorithm. Qualitatively, the output likelihoods represent an updating of the input likelihoods based on the structure of the code. The specific details of this updating are algorithm-dependent.

For an SCC, two SISO decoders are required – one to decode each of the inner and outer codes. Figure 2.7 shows the exchange of information between the decoders during the iterative process. The letters $I$ and $O$ are used to denote quantities at the inputs and outputs of the SISO block, respectively. Similarly, the superscripts $i$ and $o$ are used to denote the likelihood inputs and outputs associated with the inner and outer decoders, respectively. Thus $\lambda(c';I)$ is the input information symbol probability distribution for the SISO decoder of the inner code, etc.

![Diagram of the SISO decoder block](image)

**Figure 2.6:** The SISO decoder block.
<table>
<thead>
<tr>
<th>Likelihood term</th>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda { u; I } )</td>
<td>{ \Pr( u_k = d ) } at SISO input</td>
<td>The set of probability distributions for all information symbols ( u_k ) of a code, before processing by the SISO algorithm. Here ( d ) is an element from the set of information symbols.</td>
</tr>
<tr>
<td>( \lambda { c; I } )</td>
<td>{ \Pr( Y</td>
<td>c_k = x ) } at SISO input</td>
</tr>
<tr>
<td>( \lambda { u; O } )</td>
<td>{ \Pr( u_k = d</td>
<td>Y ) } at SISO output</td>
</tr>
<tr>
<td>( \lambda { c; O } )</td>
<td>{ \Pr( c_k = x</td>
<td>Y ) } at SISO output</td>
</tr>
</tbody>
</table>

Table 2.1: SISO input and output quantities.

![Flow of information in iterative SCC decoding](image)

Figure 2.7: Flow of information in iterative SCC decoding.
The iterative decoding process for a SCC can be described as follows:

1. Input distribution of codeword symbols $\lambda \{c^i ; I\}$ for inner SISO is fixed to be the demodulator soft outputs. Input distribution of information symbols for outer SISO $\lambda \{u^o ; I\}$ is fixed to be the uniform distribution.

2. Initialize input distribution of information symbols for inner SISO $\lambda \{u^i ; I\}$ with the uniform distribution.

3. Produce refined soft information for the information symbols using inner SISO. This soft information $\lambda \{u^i ; O\}$ is de-interleaved and becomes the input codeword symbol soft information $\lambda \{c^o ; I\}$ for the outer SISO.

4. Produce refined soft information for the codeword symbols $\lambda \{c^o ; O\}$ using outer SISO. This soft information is interleaved and becomes the input information symbol soft information for the inner SISO $\lambda \{u^i ; I\}$.

5. Repeat steps 3 and 4 until sufficient convergence has been achieved.

6. Decisions are taken by applying a slicer to the information symbol soft information $\lambda \{u^o ; O\}$ at the output of the outer SISO.

The performance of the decoder will depend on the algorithm used to update the likelihoods. The optimal algorithm for calculating a posteriori probabilities (APPs) is most commonly, if somewhat misleadingly, called the MAP algorithm. The term ‘MAP’ here is something of a misnomer, since the algorithm does not actually perform any maximization of probabilities. The MAP algorithm is also (perhaps more properly)
known as the BCJR algorithm, after the authors [7], or simply the APP algorithm. Here we will continue to use the term ‘MAP algorithm’ due to its popularity in the literature. Other suboptimal, but also common, APP algorithms include the max-log-MAP algorithm [8], and the SOVA algorithm [9], each of which will be described briefly.

The choice of algorithm is a tradeoff between performance and computational complexity. Algorithms with better performance tend to have higher computational complexity. The performance / complexity tradeoffs of the MAP, max-log-MAP, and SOVA algorithms are examined in [25]. The performance and computational complexity of the three algorithms when applied to iterative decoding are ranked in Table 2.2 (a lower number indicates better performance or lower complexity).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAP</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>max-log-MAP</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>SOVA</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.2: Relative performance and complexity of SISO algorithms.

### 2.2.3 The Maximum A Posteriori Probability (MAP) Algorithm

The maximum a posteriori probability (MAP) algorithm was proposed simultaneously by Bahl et. al. [7] and McAdam et. al. [10] in 1972. The original application was for decoding of convolutional codes to minimize the bit error probability (as opposed to the Viterbi algorithm, which minimizes sequence error probability). However, in the hard decision case, the performance gain over the Viterbi algorithm is not significant, and
MAP is therefore not often implemented in such cases because of the higher computational complexity.

Where the MAP algorithm shines is in the case where the output of one decoder feeds another. It is well known that by passing soft information from the first decoder to the second, performance is greatly improved by minimizing the loss of information between the decoders. Here the MAP algorithm is the better choice, as it naturally produces reliabilities for the information and codeword symbols, while the Viterbi algorithm is a hard-decision algorithm.

The SISO decoder version MAP algorithm is derived in Appendix C. Remember that the goal of the algorithm is to produce the refined likelihoods \( \lambda \{ u ; I \} \) and \( \lambda \{ c ; I \} \) from the input likelihoods \( \lambda \{ u ; O \} \) and \( \lambda \{ c ; O \} \), as described in Table 2.1. The update equations are:

\[
\begin{align*}
\lambda \{ u ; O \} : & \quad \Pr(u_k = d \mid Y) = K_1 \sum_{(s' : s) \in \mathcal{X}} \alpha_{k-1}(s') \cdot \Pr(Y \mid c_k = c(s', s)) \cdot \beta_k(s) \\
\lambda \{ c ; O \} : & \quad \Pr(c_k = x \mid Y) = K_2 \sum_{(s' : s) \in \mathcal{X}} \alpha_{k-1}(s') \cdot \Pr(u_k = u(s', s)) \cdot \beta_k(s)
\end{align*}
\]

where:

\[
\begin{align*}
\lambda \{ u ; O \}, \lambda \{ c ; O \} & = \text{refined likelihoods output by MAP algorithm} \\
d & = \text{an element from the set of information symbols} \\
x & = \text{an element from the set of codeword symbols} \\
Y & = \text{vector of observations } \{ y_i \} \\
K_1, K_2 & = \text{normalization constants} \\
(m', m) & = \text{set of states associated with a transition in the code trellis}
\end{align*}
\]
\[ m = \text{state at time } k \text{ (i.e. at a depth } k \text{ into the code trellis)} \]

\[ m' = \text{state at time } k - 1 \]

\[ u(m', m) = \text{the information symbol causing the state transition } (m', m) \]

\[ c(m', m) = \text{the codeword symbol produced by the state transition } (m', m) \]

\[ \alpha_k(m) = \Pr(s_k = m, y_{i:s_k}) \]

\[ \beta_k(m) = \Pr(y_{i:s_k} | s_k = m) \]

The terms \( \alpha_k(s) \) and \( \beta_k(s) \) are calculated via a forward and backward recursion, respectively:

\[
\begin{align*}
\alpha_k(m) &= \sum_s \varphi_k(m', m) \cdot \alpha_{k-1}(m') \\
\beta_{k-1}(m') &= \sum_s \varphi_k(m', m) \cdot \beta_k(m)
\end{align*}
\]

where a new term \( \varphi(m', m) \) has been defined as

\[
\varphi_k(m', m) = \Pr(s_k = m, y_k | s_{k-1} = m')
\]

and can be calculated according to the equation

\[
\varphi_k(m', m) = \Pr(u_k = u(m', m)) \cdot \Pr(Y | c_k = c(m', m))
\]

The operation of the MAP algorithm can be summarized as follows:

1. Initialize \( \alpha_0(m) \) and \( \beta_N(m) \). If the initial and final states are known, then
\[ \alpha_0 (m) = \begin{cases} 1 & \text{if } m = \text{initial state} \\ 0 & \text{otherwise} \end{cases} \]

\[ \beta_{\infty} (m) = \begin{cases} 1 & \text{if } m = \text{final state} \\ 0 & \text{otherwise} \end{cases} \]

If either is unknown, then the corresponding initialization is made to the uniform distribution.

2. Receiver receives another codeword \( y_k \) from the channel. The new codeword is stored in memory, and \( \alpha_k (m) \) is calculated for each state \( m \). The calculated values of \( \alpha_k (m) \) are also stored in memory.

3. Step 2 is repeated until the entire sequence is received.

4. The backward recursion is performed to calculate \( \beta_k (m) \) for all \( m, k \).

5. The probabilities \( \Pr(u_k = d \mid Y) \) and \( \Pr(c_k = x \mid Y) \) are calculated for each \( d \) and \( x \), at each step \( k \), to obtain the refined likelihoods \( \lambda \{ u; O \} \), \( \lambda \{ c; O \} \).

More information on the derivation of the MAP algorithm and its implementation can be found in [6], [7], [8], and [11].

2.2.4 The max-log-MAP Algorithm

Although the MAP algorithm produces the optimal \textit{a posteriori probability} values, its complexity is significantly higher than a conventional hard-decision Viterbi algorithm, as
it performs both a forward and a backward recursion, while the classical Viterbi requires only a forward recursion. The max-log-MAP algorithm was developed as a modification of the MAP algorithm which reduces the computational complexity while slightly degrading the performance.

The algorithm starts with the log form of the MAP algorithm. It can be shown [6, eqs. 37-42] that the likelihood ratios in the log domain are given by:

\[
\lambda_k (u_k; O) = \max_{s \in A} \{\alpha_{k-1} (s) + \lambda_k (c_k; I) + \beta_k (s)\}
\]

\[
\lambda_k (c_k; O) = \max_{s \in B} \{\alpha_{k-1} (s) + \lambda_k (u_k; I) + \beta_k (s)\}
\]

We adopt here the star notation first used in [29] for MAP detection of ISI channels. The max operator basically performs a log of sums of exponentials:

\[
\max_j (a_j) = \log \left[ \sum_j e^{a_j} \right]
\]

In practice this is a complex operation to implement. The max-log-MAP simplifies this step by using the approximation

\[
\log \left[ \sum_{j=1}^J e^{a_j} \right] \approx \max_j (a_j)
\]

In other words, the max operation is approximated by a conventional max.

More information on the derivation and implementation of the max-log-MAP algorithm can be found in [6] and [8].
2.2.5 The A Priori Soft-Output Viterbi Algorithm (APRI-SOVA)

The Viterbi algorithm (VA) is the standard method for performing sequence detection when the sequence can be described by a trellis. Decoding of these trellis codes is well known [12], [13, pp. 292-314]. The classic VA performs a maximum likelihood sequence detection with computational complexity that grows only linearly with sequence length by using the concept of survivors. The VA associates each transition from a state $s_{k-1}$ to $s_k$ with a metric value corresponding to the likelihood of that transition. At any given point in the trellis $k$, the VA only has to remember a single survivor for each state, where the survivor is defined as the path with the best metric value. It can be shown that none of the non-survivor paths can be the maximum likelihood path. The conventional VA metric is based either on Hamming distance (for hard inputs) or Euclidean distance (soft inputs).

The conventional VA has two drawbacks when considered as a SISO algorithm candidate: it cannot make use of any likelihood information about the information bits, and it produces only hard outputs. In [9] Hagenauer and Hoeher propose a modified VA (called the soft-output Viterbi algorithm, or SOVA) which outputs reliability information along with the hard decisions. Hagenauer later makes another modification to the VA metric which incorporates a priori knowledge of the source bits in [15] and calls this modified version the APRI-VA. By combining the APRI-VA and the SOVA into a single unit (the APRI-SOVA), the result can be used as a SISO decoding algorithm.

Due to its popularity, we will assume that the reader is familiar with the conventional Viterbi algorithm and proceed to describe the modifications which comprise the APRI-SOVA. Additional details on the conventional Viterbi algorithm may be found in Forney's tutorial paper [12].
The *a priori* Viterbi Algorithm

First proposed in [15], the APRI-VA attempts to incorporate *a priori* information into the conventional Viterbi algorithm metric. Since our main concern in this thesis is binary signaling, we will restrict our discussion to binary trellises. The APRI-VA searches for the state sequence $s^{(m_p)}$ which maximizes the *a posteriori* probability:

$$\max_{m_p} P\left(s^{(m_p)} \mid Y \right)$$  \hspace{1cm} (2.4)

Here $m_p$ is the index labeling a specific path through the code trellis, and $Y$ is the vector of observations.

Since we have assumed a binary trellis, at any stage in the trellis each state has two branches entering it. Hagenauer shows in [15] that equation (2.4) can be reduced to the maximization of an equivalent metric $M_k$ which is recursively defined to be

$$M_k^{(m)} = M_{k-1}^{(m)} + \sum_{n=1}^{N} \alpha_{k,n} L_{c_{k,n}} y_{k,n} + \mu_k^{(m)} L(\mu_k)$$  \hspace{1cm} (2.5)

where:

- $x_{k,n}^{(m)}$ = the $n^{th}$ bit associated with the codeword generated by a transition $m$ at the $k^{th}$ stage of the trellis
- $y_{k,n}$ = the $n^{th}$ bit of the received codeword at stage $k$
- $L_{c_{k,n}}$ = reliability term = $4a \left( E_b / N_0 \right)$
- $a$ = fade level ( $a=1$ for AWGN, time-varying for fading )
- $E_b / N_0$ = signal-to-noise ratio
\[ L(u_k) = \text{log-likelihood probability of the information bit } u_k \]

The log-likelihood probability is mathematically defined as

\[ L(u_k) = \log \left( \frac{Pr(u_k = +1)}{Pr(u_k = -1)} \right) \]

The APRI-VA thus operates exactly as the conventional Viterbi, with the noted modification of the metric. At each stage in the trellis, the metric which determines the survivor paths is replaced by equation (2.5).

The operation of the APRI-VA metric is depicted in Figure 2.8. The three inputs correspond to the three terms in equation (2.5) – the first term is the "old metric" (arising from the recursive nature of the metric calculation), the second term is the channel information, and the third term is the \textit{a priori} information. Intuitively the metric relies more on the received codewords when the bit SNR is high (corresponding to a good channel). When the channel is bad (such as in a deep fade), the metric relies on the \textit{a priori} information.

![Figure 2.8: The APRI-VA Metric](image-url)
The Soft-Output Viterbi Algorithm

The SOVA proposed by Hagenauer in [9] is essentially a conventional VA augmented with additional steps to produce the reliability information. The reliability information may be implemented by either the register exchange mode [9] or the traceback method [15]. The register exchange method is described below.

Each time the Viterbi algorithm takes another step through the trellis, it must decide for each state which path is the survivor path for that state. For simplicity, consider the binary case, where each state has two paths leading into it. The Viterbi algorithm selects the path with the smallest distance metric to be the survivor path. For convenience, label the survivor “path #1” with metric $M_1$ and the rejected path “path #2” with metric $M_2$. The probability of each path being correct is then:

$$\Pr(\text{path m}) \propto e^{-M_*}$$

By our choice of labeling, we have $M_1 < M_2$. The probability $p_{se}$ that the incorrect path was selected (i.e. that path #2 was actually the correct path) is

$$p_{se} = \frac{\text{prob( path 2 )}}{\text{prob( path 1 )} + \text{prob( path 2 )}} = \frac{e^{-M_2}}{e^{-M_1} + e^{-M_2}} = \frac{1}{1 + e^{M_2 - M_1}}$$

$$p_{se} = \frac{1}{1 + e^{\Delta_j}}, \quad (2.6)$$

where we have defined $\Delta_j = M_2 - M_1$. 

23
We now need to determine the two quantities $\lambda \{ u; O \}$ and $\lambda \{ c; O \}$. To do so, we must obtain $\Pr(u_k = d | Y)$ and $\Pr(c_k = x | Y)$. We will first consider $\Pr(u_k = d | Y)$.

Recall from the classical Viterbi algorithm that a stage in the code trellis is formed of a number of states connected with lines (called "edges"). Each edge is labeled with both an information symbol (which causes the state transition from the state at the left end of the edge to the state at the right end) and a codeword (generated by the transition). Again, we restrict ourselves to the binary case. Consider an information bit associated with any edge along path #1. If the corresponding bit in path #2 is the same, then regardless of which path is chosen that bit cannot be in error. However, if they are different, then we will have made an error if we choose path #1 when in fact path #2 was correct.

Recall that the truncation depth is the number of trellis stages (counting backwards from the current stage) into the trellis that the Viterbi algorithm will store information on. Decisions are made and output on the bits outside that depth. Typically the truncation depth depends on the probable length of an error event before the error event merges back into the correct sequence [12]. Define this truncation depth of the algorithm to be $\delta_m$. Suppose there are $e$ differing bits along the two paths, and $\delta_m - e$ identical bits. Assume that we have stored in memory the collection of probabilities $\{ p^{(u)}_{j,k} \}$ that the $j^{th}$ information bit of the $k^{th}$ stage in the trellis is incorrect. Then under SOVA these probabilities are updated according to the following update equation:

$$
\hat{p}^{(u)}_{j,k} \leftarrow p \left( \text{bit } j \text{ on path } #1 \text{ was previously incorrect} \right) \cdot p \left( \text{path } #1 \text{ was chosen} \right) + p \left( \text{bit } j \text{ on path } #1 \text{ was previously correct} \right) \cdot p \left( \text{path } #2 \text{ was chosen} \right)
$$

or, more concisely,

$$
\hat{p}^{(u)}_{j,k} \leftarrow p^{(u)}_{j,k} (1 - p_{se}) + (1 - p^{(u)}_{j,k}) p_{se} . \tag{2.7}
$$
Then the probabilities $\Pr(u_k \mid Y)$ are approximated by the appropriate $p_{j,k}^{(u)}$ via

$$\Pr(u_k \mid Y) \approx 1 - p_{j,k}^{(u)} \quad (2.8)$$

We can similarly approximate $\Pr(c_k \mid Y)$ using $p_{j,k}^{(c)}$, the probability that the $j^{th}$ codeword bit of the $k^{th}$ stage in the trellis is incorrect. By following the same line of reasoning as before, we obtain the update equation for $p_{j,k}^{(c)}$ as

$$\hat{p}_{j,k}^{(c)} \leftarrow p_{j,k}^{(c)} (1 - p_{se}) + (1 - p_{j,k}^{(c)}) p_{se} \quad (2.9)$$

and calculate $\Pr(c_k \mid Y)$ using

$$\Pr(c_k \mid Y) \approx 1 - p_{j,k}^{(c)} \quad (2.10)$$

Thus the implementation of the SOVA merely augments the operation of the conventional Viterbi algorithm. The SOVA proceeds exactly as the Viterbi, using the conventional or the APRI-SOVA metric (depending on whether a priori information is to be incorporated), with the additional chore of storing and updating the bit reliabilities $\{p_{j,k}^{(u)}\}$ and $\{p_{j,k}^{(c)}\}$.

The register exchange method for implementing the SOVA can be summarized in the following steps:

1. Proceed according to the standard VA for determining survivor paths, etc.
2. For each step through the trellis, add the following steps:
a) Initialize the probabilities of error for the information and codeword bits $p_{j,k}^{(u)}$ and $p_{j,k}^{(c)}$ associated with the most recent state transition $s_{k-1}$ to $s_k$ to 0.

b) For $j = k$ to $k - \delta_m$, compare the information bits associated with the transition at index $j$ along the two paths merging at $s_k$. If they differ, then update the probability of those bits via equation (2.7). Repeat for the codeword bits.

Figure 2.9 shows a pictorial representation of a step through a trellis of a code with 2 memory elements ($\nu = 2$) and truncation depth chosen to be 5. Only the information bits labeling the edges are shown. Notice that the last $\nu$ edges in each path are always labeled by the same information bits in order for the paths to end in the same state for non-recursive codes. This is not necessarily true of codeword bits. For a recursive code, the last $\nu$ edges may have differing information and codeword bits. More information on the APRI-SOVA can be found in [9] and [15].
Last $v$ bits must be the same for the two paths to end in the same state.

Figure 2.9: A step through the trellis of a non-recursive code with memory $v = 2$. 


Chapter 3

Reduced-Complexity Receiver Design

In this chapter, design issues for a GMSK receiver based on the Laurent representation will be examined. In this section, we have two main goals: complexity reduction and memory extraction.

In designing the demodulator, we would like to choose a representation which allows for complexity reduction. Here, we equate complexity with the computational requirements of demodulation. It is obviously intrinsically beneficial to design a reduced-complexity complexity demodulator because higher computational requirements typically translate into higher cost of implementation. We would also like to keep in mind that we will later be incorporating iterative decoding, a very computationally-intensive operation. Therefore it would be helpful if some corresponding complexity reduction could be achieved in the demodulation operation.

The representation of GMSK used to perform the demodulation should also provide the added feature of memory extraction. It is well known that GMSK, being a type of CPM, has modulation memory due to the imposed continuous phase constraints. We would later like to use this memory in conjunction with error-correction coding. However, it is not obvious how the modulation memory may be extracted for this use. Although there are a number of representations (such as Laurent, Rimoldi, etc.), each representation extracts the memory in a slightly different form, and the question of which form is most suitable for use in a serial concatenated code is not known.
Some work in the area of coded CPM was performed by Szeto [11] [27] using the Rimoldi representation which showed that the Rimoldi memory was not particularly well suited for use in a serial concatenated code scheme. Here we will focus on the use of the Laurent representation, which extracts the modulation memory in the form of a recursive rate 1 code, since results in [19] using recursive rate 1 inner codes were very promising.

In this thesis, we will choose to focus on the Global System for Mobiles (GSM) standard of GMSK which specifies a BT of 0.3. Following the convention in the literature, for BT=0.3 we will truncate the phase pulse to L = 3. In most cases, this choice of BT is done for convenience or because a numerical value must be chosen for simulation. In any cases where a result depends on the particular value of BT, the dependence will be explicitly stated.

Using the Laurent representation, a 4-state reduced complexity receiver will be derived for BT=0.3, and the performance compared with that of the optimal 16-state receiver in AWGN. The 4-state receiver will be seen to have only a slight performance degradation compared to the optimal receiver. Next, the modulation memory of GMSK will be isolated, and we will show that if this memory is removed, the remaining PAM modulator is actually performing an optimal bit sequence-to-waveform mapping, in Gray code fashion.

### 3.1 The Laurent Linear Receiver

This section will derive the form of the Laurent representation for GMSK with BT=0.3 and investigate a reduced-complexity demodulator constructed using this representation.
3.1.1 Construction of the Laurent Representation

Although [2] gives the method for deriving the Laurent representation, there is no closed-form general expression valid for all CPM schemes, and the impulse response of the PAM processes must be derived on a case-by-case basis. Therefore the Laurent representation and the main Laurent pulse for BT=0.3 is derived in Appendix A. We restate the final result of the derivation below:

\[
\begin{align*}
    s_{L1}(t) &= \sum_{n=0}^{N-1} \left( j^{n} \alpha_{n} \right) s_{0}(t-nT)s_{1}(t-nT)s_{2}(t-nT) \\
    &= \sum_{n=0}^{N-1} \left( j^{n} \alpha_{n} \right) s_{0}(t-nT)s_{1}(t-nT)s_{2}(t-nT) \\
    \text{modulating bit in PAM representation, at time n} & \quad \text{PAM pulse } h_{o}(t-nT)
\end{align*}
\]  

Here \( s_{0}(t) \), \( s_{1}(t) \), and \( s_{2}(t) \) are intermediate quantities in the final derivation of the Laurent representation. \( n \) is the index of the current bit of the input sequence, which is assumed to have total length \( N \). The index \( i \) is simply an index of summation, and not the imaginary number. The input bit sequence is composed of the collection of \( \{\alpha_{n}\} \), each of which takes on values in the set \( \{+1,-1\} \).

Comparing equation (3.1) with [3, eq. 4], we see that with a little rearranging the above equation becomes the conventional PAM form,

\[
    s_{L1}(t) = \sum_{n=0}^{N-1} \left( j^{n} \alpha_{n} \right) h_{o}(t-nT) 
\]  

(3.2)

where \( h_{o}(t) \) is the Laurent main pulse, given by

\[
    h_{o}(t) = s_{0}(t)s_{1}(t)s_{2}(t) 
\]  

(3.3)
The pulse shape for \( h_0(t) \) is shown in Figure 3.1. Note that the pulse duration is approximately 3T. If we define the complex encoded quantity

\[
a_n = j^{\sum_{n=0}^{\infty} a_n},
\]

then we can rewrite equation (3.1) in its PAM form:

\[
s_{t_1}(t) = \sum_{n=0}^{N-1} a_n h_0(t - nT).
\]

Note that the new, complex, "encoded" quantity \( a_n \) takes on values from the set \( \{\pm 1, \pm j\} \).

Figure 3.1: The Laurent \( h_0(t) \) pulse.
Recall that the exact trellis for GMSK is described by $4 \cdot 2^{L-1}$ states \([1]\). For $L=3$, this is 16 states. Equation (3.5) shows that the Laurent representation, which is essentially a PAM process with $L=3$, requires only $2^{L-1} = 4$ states in its trellis description. Conceptually, the Laurent linear approximation replaces the exact GMSK 16-state modulator with the structure shown in Figure 3.2.

![Figure 3.2: Linear modulator structure (the $\Sigma$ denotes conventional addition).](image)

In simulation, it will be more convenient to adopt the product form described in [4] for our expression of $a_n$ in equation (3.4). Essentially, this variation replaces the accumulator and I/Q mapper of Figure 3.2 with a mathematically equivalent product differential encoder and a modified I/Q mapper. If the initial state of the accumulator is assumed to be zero, then we can note that the accumulator expression can be rewritten as follows:

$$j^{\sum_{i=0}^{n} a_i} = \prod_{i=0}^{n} j^{a_i} = j^{n+1} \sum_{i=0}^{n} \alpha_i .$$

(3.6)
This is easily shown to be the case by first noting that \( \alpha_n \in \{ \pm 1 \} \), and verifying that for this set of possibilities \( j^{\alpha_n} = j \cdot \alpha_n \). A simple application of this identity to the product of exponentials in the second step of equation (3.6) leads to the final form.

Substituting this result into equation (3.4), we obtain:

\[
a_n = j^{n-1} \sum_{n=0}^{n} \alpha_i . \tag{3.7}
\]

Again using equation (3.5), but with the new formation of \( a_n \) described by equation (3.7), we obtain the product differential encoder version of the modulator shown in Figure 3.3.

![Diagram of product differential encoder version of the Laurent modulator.](image)

Figure 3.3: The product differential encoder version of the Laurent modulator.

The intermediate quantity \( \{ \beta_n \} \) consists of the differentially encoded bits prior to the I/Q mapping, also taking on values from the set \( \{ \pm 1 \} \), and is mathematically defined as
Equation (3.7) can be rewritten using this intermediate quantity as

\[ a_n = j^{n+1} \beta_n \]  

(3.9)

### 3.1.2 The Laurent-Based Demodulator

Numerical integration of the main Laurent pulse shown in Figure 3.1 reveals that it contains 0.997, or 99.7%, of the total signal energy. The strategy adopted by the Laurent-based demodulator is to process only the PAM process associated with the main pulse. The remaining pulses are essentially treated as noise. The MLSE receiver maximizes the criterion

\[ \lambda^{(a)} = -\int_{-\infty}^{\infty} |r(t) - s^{(a)}(t)|^2 \, dt \]

where \( r(t) \) is the received signal, \( a \) is an index assigned to a given path through the modulation trellis, and \( s^{(a)}(t) \) is the signal output obtained by traversing path \( a \) through the trellis.

Since the pulse duration of \( h_0(t) \) is 3T, each bit interferes with the two adjacent bits. In other words, for any given bit of interest there are 8 possible pulse shapes over the corresponding time interval of 2T. (Remember that bits are alternately assigned to the in-phase and quadrature channels, so that each channel is essentially transmitting half of the total bit sequence at half the rate. This accounts for why the time interval is 2T rather than T when the I and Q channels are being considered separately.) We denote these
pulse shapes \( g_k(t) \), where \( k \) is the index of the corresponding triplet of bits as indicated in Table 3.1, and takes on values in the set \( \{1, 2, \ldots, 8\} \). The pulse shapes \( g_k(t) \) are given by the equation

\[
g_k(t) = a_{n-1} h(t+2kT) + a_n h(t) + a_{n+1} h(t-2kT), \quad (n-1)T \leq t < (n+1)T
\]  \hspace{1cm} (3.10)

<table>
<thead>
<tr>
<th>Index ( k )</th>
<th>( a_{n-1} )</th>
<th>( a_n )</th>
<th>( a_{n+1} )</th>
<th>pulse shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>( g_1(t) )</td>
</tr>
<tr>
<td>2</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>( g_2(t) )</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>( g_3(t) )</td>
</tr>
<tr>
<td>4</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>( g_4(t) )</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>( -g_4(t) )</td>
</tr>
<tr>
<td>6</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>( -g_3(t) )</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
<td>( -g_2(t) )</td>
</tr>
<tr>
<td>8</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>( -g_1(t) )</td>
</tr>
</tbody>
</table>

Table 3.1: All possible pulse shapes for BT=0.3.

Therefore at most we need 8 matched filters. This can be reduced to 4 by noting that the four of the pulses are the negative versions of the other four. The outputs of the matched filter are then fed to a Viterbi algorithm which performs the MLSE. Figure 3.4 shows a block diagram representation of the demodulator.

In this demodulator, the complex baseband signal is first treated as two separate, half-rate signals being modulated on the I and Q channels, with the even bits on one and the odd
bits on the other. The output of the full-rate matched filters must be downsampled by a factor of two after the real and imaginary operations are performed for the I and Q streams. The signals are then demodulated by two 4-state, half-rate Viterbi algorithms and merged to give the differentially encoded bits \( \beta_n \in \{ \pm 1 \} \), which is then decoded using a feed-forward decoder to recover the input bits \( \alpha_n \in \{ \pm 1 \} \). Since the two Viterbi algorithms operate only at half rate and do not exchange information, the total computational requirements per bit are the same as for a full-rate 4-state Viterbi.

Figure 3.4: Laurent receiver based on a one-pulse model. The \( \downarrow \) denotes downsampling.
A note on simulation: an alternative method of implementing the Laurent receiver involves only as many matched filters as PAM pulses [3]. In this system, a receiver targeted to only the main pulse requires only one matched filter with duration 3T. A receiver targeted to recovering the first 2 Laurent pulses (which Kaleh shows in [3] to contain 0.9999 of the signal energy) would require 2 matched filters, and so forth. If we consider the number of pulses that the receiver recovers to be a parameter, then this alternative method requires a variable number of matched filters, which is inconvenient to implement in simulation. Our chosen method requires a fixed number of matched filters for both the single pulse and 2-pulse cases, which was more convenient to code, since initially both cases were investigated. Since both methods are mathematically equivalent, performance is unaffected by the choice of implementation method.

3.2 Reduced-Complexity Receiver Performance

Since the reduced-complexity linear receiver is only recovering the energy contained in the first signal component, we expect that the performance will be slightly poorer than that of the optimal 16-state receiver. This is validated through simulation, where Figure 3.5 shows that the performance loss in opting for the linear receiver is only ~0.2-0.3 dB.

The performance loss could be reduced by increasing the complexity of the Laurent receiver. The most straightforward way to accomplish this is to also take the second Laurent pulse into account, following the methodology described in [3, Section V]. This was investigated in simulation, but the more complex receiver failed to yield a significant gain over the linear receiver. Even in the best case (i.e. by using a full 16-state demodulator) the performance gain would only be a few tenths of a dB. As we will see
in Chapter 4, if additional computational resources are available, more significant performance gains can be achieved by using iterative decoding.

Figure 3.5: Performance of the 4-state Laurent receiver vs. the optimum receiver.

Note that for all simulations the conventional GMSK modulator described in Section 2.1.1 was used, even in the cases where a reduced-complexity demodulator is employed. This allows for a fair comparison of the different demodulation methods.

### 3.3 Precoded GMSK

It is already known that preceding a GMSK modulator with a feed-forward encoder improves the performance of coherent detection [4], as it corrects the double-error
characteristic of GMSK. We will show that GMSK can actually be seen as a "modulation memory" represented by a rate 1 code followed by a PAM process. The PAM process will be seen to have the 'Gray coding' property; that is, input sequences are mapped to waveforms in such a way that the Euclidean distance between two waveforms varies as the Hamming weight of the difference between the corresponding sequences.

### 3.3.1 'Gray Coding' Property of the Laurent PAM Process

Gray coding is a concept commonly employed in QPSK-type schemes. In a Gray-coded constellation, each constellation point is labeled with an appropriate block of bits such that adjacent points differ in only one bit [17, p. 328]. This minimizes the number of bits per symbol error.

We may apply a similar technique to modulation schemes described by trellises instead of constellations. In the trellis case, we would like input sequences to be mapped to waveforms such that waveforms that are adjacent in the Euclidean-distance sense (at the output of the matched filter) differ only by one bit. This is in fact achieved by the Laurent PAM process.

To see that this is the case, consider two possible channel bit sequences \( \{c_n\} \) and \( \{d_n\} \) which differ by those bits with indices \( i \in D \). The squared Euclidean distance is defined as

\[
\int \left| \sum_{n=0}^{N} (c_n - d_n) h(t - nT) \right|^2 dt .
\]

By definition, the difference \( c_n - d_n \) is non-zero only for those \( n \in D \). Therefore the summation reduces to:
\[
\int \left| \sum_{i \in D} (c_i - d_i) h(t - iT) \right|^2 dt.
\]

Applying the well-known identity

\[
\left| \sum_i x_i \right|^2 = \sum_i |x_i|^2 + 2 \sum_{i \neq j, i > j} \text{Re}\{x_i x_j^*\},
\]

where the asterisk is used to denote complex conjugation, we may expand the magnitude-squared operation and simplify the integrand as follows:

\[
\left| \sum_{i \in D} (c_i - d_i) h(t - iT) \right|^2 = \sum_{i \in D} \left| (c_i - d_i) h(t - iT) \right|^2 + 2 \sum_{i \neq j, i > j} \text{Re}\{(c_i - d_i)(c_j - d_j)^* h(t - iT)h(t - jT)\}
\]

Note that in the second summation, all terms where \(i\) and \(j\) are not both even or both odd are imaginary and therefore do not contribute to the summation. This condition is mathematically equivalent to requiring that \(i - j = 0 \mod 2\). If the energy of the PAM pulse is \(E_b\), then the integral of the first term is simply \(4kE_b\), where \(k\) is the number of elements in \(D\) (i.e. the number of differing channel bits). The integral of the second term is a perturbation caused by ISI and has the form

\[
8 \sum_{i \in D, j \in D \atop i \neq j, i > j \atop i - j = 0 \mod 2} \text{sgn}\left[(c_i - d_i)(c_j - d_j)\right] \int h(t - iT)h(t - jT) dt \quad (3.11)
\]
Notice that the integral \( \int h(t - iT)h(t - jT) dt \) in equation (3.11) decreases monotonically with increasing \(|i - j|\). Therefore, given the restrictions on the index of the summation that \( i \neq j \) and \( i - j = 0 \mod 2 \), the largest value that this integral obtains in the summation is achieved when \(|i - j| = 2\). In other words,

\[
\int h(t - iT)h(t - jT) \, dt \leq \int h(t)h(t - 2T) \, dt
\]

(3.12)

for all \( i, j \) in the summation.

The magnitude of this term is related to the amount of overlap between the two shifted versions of the pulses and is determined by the amount of ISI (see Figure 3.6). For \( BT=0.3 \), the right-hand term in equation (3.12) can be numerically integrated, and the result comes out to \(0.057E_b\). For waveforms which are "close" in Euclidean distance, the size of set \( D \) is small, the number of elements in the summation of equation (3.11) is also small, and the ISI term can be neglected. The squared Euclidean distance is then approximated by

\[
\text{Distance}^2 = 4kE_b
\]

(3.13)

Note that the distance is directly proportional to the number of bit differences in the sequences. Therefore, for any given sequence error probability, the corresponding bit error probability for the channel bits \( \{a_i\} \) has been minimized.

In general, for CPM schemes with different amounts of ISI, the size of the right-hand term in equation (3.12) must be investigated to determine if it is appropriate to neglect the effects of the ISI term. Obviously, for GMSK, the smaller the value of \( BT \), the more significant the effects of the ISI term will be.
3.3.2 The Role of Precoding

Any rate 1 code preceding the PAM process of the Laurent linear modulator will alter the input sequence-to-waveform mapping and degrade the performance of the system, since the modulator will lose its Gray coded property. This includes the differential encoder structure resulting from the GMSK modulation memory. To correct this, the input sequence may be precoded to remove the effects of the differential encoder.

Figure 3.7 shows the role of a precoder in removing the modulation memory. We would like to add a precoder which produces an output sequence \( \{a'_n\} \) from an input sequence \( \{a_n\} \) such that when \( \{a_n\} \) passes through the differential encoder, the output \( \{a'_n\} \) is
identical to the original input \( \{ \alpha_n \} \). To do this, we simply define the transfer function of the precoder to be the inverse of the modulation memory:

\[
f(D) = \frac{1}{1 \oplus D}
\]

which in the product domain is simply a feed-forward encoder, shown in Figure 3.8. The simple structure of the precoder makes precoding a very attractive method for low-complexity performance improvement, since a feed-forward precoder is very simple to implement.

Figure 3.7: Using precoding to remove the modulation memory.

Figure 3.8: A feed-forward encoder structure.
3.3.3 Simulation Results for Precoded GMSK

We confirm through simulation that a performance improvement can indeed be seen with GMSK using a precoder. Figure 3.9 shows a factor of 2 improvement in the BER of precoded GMSK with BT=0.3. In fact, other CPM schemes can also benefit from the same precoding, as long as the effect of the PAM ISI term in equation (3.11) is still negligible. This is true for most CPM schemes in practical use. Figure 3.10 shows the same improvement for GMSK schemes with varying BT values.

Figure 3.9: Precoded GMSK, BT=0.3.
Figure 3.10: Precoded GMSK, various BT and L values.
Chapter 4

Iterative Decoding of Coded GMSK

In the previous chapter, we described a reduced-complexity demodulator based on the Laurent representation. One of the results of the Laurent-based model was the extraction of the modulation memory in the form of a recursive rate 1 code. In this chapter, we will examine the use of this modulation memory as the inner code of a serial concatenated code scheme.

The starting assumption is that practical systems are typically protected by an error-correcting code such as a convolutional code. Conventional systems treat the coding and the modulation separately – at the receiver side, the demodulator and the decoder are typically discrete entities, where the decoder does not know the type of modulation scheme used, and the demodulator may not even be aware of the existence of the decoder. By adopting a joint view of modulation and coding, we will see that we can improve performance through the treatment of the error correcting code and the modulation memory as a serially concatenated code (SCC), and applying iterative decoding at the receiver. This allows for large gains in performance without bandwidth expansion, since no additional memory will be imposed on the system; the performance gain is strictly realized by a more innovative utilization of the pre-existing memory. The cost will mainly be from the increased computational requirements associated with iterative decoding, and longer delays resulting from the large block sizes necessary to achieve good performance with SCCs.
4.1 System Model

4.1.1 Transmitter Models

The benchmark system for comparison purposes will be a coded GMSK model shown in Figure 4.1. This is a conventional system where a good convolutional code is chosen to protect the input bits (many standard textbooks such as [18] have lists of ‘good’ codes), and the coded bits are passed onto the modulator.

![Figure 4.1: Conventional (benchmark) system.](image)

![Figure 4.2: SCC system using GMSK memory as inner code.](image)
It is well known [6] that iterative decoding of a serially concatenated code yields good results. Recent results have shown that this is the case even when the inner code is a rate 1 code rather than a more conventional, lower-rate code [19]. In light of this, we propose that a performance gain over the reference system can be achieved by inserting an interleaver between the error-correcting code of Figure 4.1 and the GMSK modulator, and treating the concatenation of the error-correcting code and the GMSK memory as an SCC.

Good results have been obtained by others using a similar structure for differential phase-shift keying (DPSK) by Narayanan and Stuber [14]. The use of DPSK allowed Narayanan and Stuber to easily identify the modulation memory, since DPSK is essentially defined to be a recursive encoder followed by conventional PSK. By contrast, for CPM schemes (such as GMSK), although modulation memory is present due to the continuous phase constraints, it is not obvious how to choose a representation which extracts the memory in a fashion most suitable for use in an SCC. Szeto examined the use of the Rimoldi representation to extract CPM memory for use in SCCs in [11], and concluded that SCCs formed with a single error-correcting code as the outer code and the Rimoldi memory as the inner code did not achieve the dramatic performance gains typically exhibited by SCCs, suggesting that the Rimoldi representation may be poorly suited to memory extraction from the perspective of SCC design.

Here we will examine the use of the Laurent memory as the inner code. The Laurent memory is essentially a recursive encoder and differs in structure from the Rimoldi memory. Recursive encoders rate 1 codes as inner codes were examined in [19], and the Laurent recursive encoder is identical to one of the high-performance rate 1 codes used in [19], which leads us to believe that the Laurent representation may allow us to achieve results better than [11] showed with the Rimoldi representation.

The proposed Laurent-based model is shown in Figure 4.2. At the modulator side, it differs only in the addition of the interleaver, which the coding literature in general has
shown to be a critical component of an SCC. Note that the interleaver does not cause bandwidth expansion (the overall code rate is unaffected), and most interleavers do not require any computational operations to implement.

4.1.2 Demodulator Model

The demodulator for the SCC system of Figure 4.2 is basically the same as the original linear receiver proposed in Chapter 3, with the differential decoder replaced by an iterative decoder which decodes the SCC.

![Diagram of the proposed receiver for the SCC system.](image)

Figure 4.3: Proposed receiver for the SCC system.

The iterative decoder is a block taking several parameters which can be varied to adjust the performance and computational costs of the system. These are:

- Block length
- Interleaver
- Number of iterations
- SISO decoding algorithm
A random interleaver was used in this thesis. All other parameters will be varied to gain a better understanding of their effects on overall system performance.

4.1.3 Choice of Outer Code

The choice of outer code is a parameter of the entire system (obviously both the transmitter and receiver must agree on the code to be used). Benedetto et al. describe some common design rules used for SCCs [6]. Some key points regarding the choice of outer code which apply to our case (SCC with fixed inner code) are:

- Outer code should have a large free distance.
- If possible, free distance of outer code should be odd.
- A convenient choice is a ‘good’ non-recursive outer code.

Benedetto’s rules are derived from the union bound approximation for bit error rates. Although the simulation cases examined in [6] confirm the design rules, their suitability is not assured in all cases since SCCs typically operate in the low SNR region, where the union bound is not tight. Additionally, the design rules do not take into account the suboptimal nature of the iterative decoding. Hence confirmation of the design rules for the case of coded GMSK through simulation is a necessity. Table 4.1 lists some ‘good codes’ that are considered in this thesis. (Codes here were taken from [18].) Where the outer code is not explicitly specified, the (23, 35) code was used.

All codes in this thesis are identified using the conventional octal notation. Each number within the parentheses represents an output, and when translated into binary, the 1’s indicate the positions of the taps. As an example, the structure of the (23, 35) code is shown in Figure 4.4.
Figure 4.4: (23, 35) code structure. (23, 35) in binary = (10011, 11101) in binary.

<table>
<thead>
<tr>
<th>Memory</th>
<th>Generators</th>
<th>$d_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(5, 7)</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>(15, 17)</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>(23, 35)</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>(53, 75)</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4.1: Outer codes considered in this thesis.

### 4.2 Simulation Results

Because it is very difficult to obtain any analytical performance results for iteratively decoded SCCs, we will investigate the effects of varying the system parameters in simulation.
4.2.1 Performance of Coded GMSK in AWGN

We verify through simulation that by treating GMSK as the inner code of a SCC system and applying iterative decoding, we can greatly improve overall performance of the system. The BER curve for the proposed SCC system, using 10 iterations, is plotted in Figure 4.5. In addition to uncoded GMSK, two other reference curves are presented for comparison. One uses the conventional transmitter described in Chapter 3, which feeds the GMSK modulator directly with the output of the encoder. The other uses the same transmitter as our SCC system, but does not perform iterative decoding. The iteratively decoded system shows a large gain. Also note that some gain may be obtained simply by interleaving, which helps make errors from the modulator more independent. Another way of looking at the non-iterative, interleaved system is to consider it a special case of the proposed SCC system, where the number of iterations is 1.

4.2.2 Performance Dependence on Block Size and Iterations

In general, performance of SCCs increases with block size and iterations. Unfortunately, practical systems favor low block sizes and iterations, as this decreases the delay and computational requirements. Hence the tradeoffs involved in varying the block size and number of iterations becomes a very important issue.

Figure 4.6 shows the performance dependence on number of iterations. We can see that there is a fairly strong dependence on number of iterations in the 1-10 iteration range. Using more than 10 iterations nets only a slight gain, and using more than 20 iterations shows no gain (convergence has been reached). For an application where computational resources are not overly scarce, 10 iterations appears to be a good choice. If necessary, this value can be reduced at the cost of performance.

Figure 4.7 shows the performance dependence on block size. A block size of 10,000 bits appears sufficient, as larger block sizes do not show significant gains in BER.
Figure 4.5: Performance in AWGN. 10k blocks, 10 iterations.

Figure 4.6: Performance dependence on number of iterations. 10k blocks.
4.2.3 Performance Dependence on SISO Algorithm

We consider the performance of the three algorithms described in Chapter 2: MAP, max-log-MAP, and APRI-SOVA. Additionally, the complexity of the demodulator can also be modified by varying the soft-output demodulation algorithm. Table 4.2 lists the different combinations which were examined through simulation. Cases 1, 2, and 6 are likely to be the ones of practical interest, since it makes more sense to reuse the same algorithm (for demodulation and decoding) rather than implement two different ones.

Figure 4.8 shows the simulation results. It appears that the choice of algorithm is more significant for the iterative decoder than the modulator, since cases 1 and 3 have virtually the same performance despite the suboptimal max-log-MAP used in the demodulator stage. Switching from MAP to max-log-MAP in the iterative decoder incurs a penalty of approximately 0.5 dB in the region of interest. The performance of APRI-SOVA is markedly poorer – there is a 2 dB difference between cases 1 and 5. In light of the poor
performance of APRI-SOVA, case 6 (which would perform even worse than case 5) was not considered as a candidate.

<table>
<thead>
<tr>
<th>Case #</th>
<th>Demodulator</th>
<th>Iterative Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MAP</td>
<td>MAP</td>
</tr>
<tr>
<td>2</td>
<td>max-log-MAP</td>
<td>max-log-MAP</td>
</tr>
<tr>
<td>3</td>
<td>max-log-MAP</td>
<td>MAP</td>
</tr>
<tr>
<td>4</td>
<td>MAP</td>
<td>max-log-MAP</td>
</tr>
<tr>
<td>5</td>
<td>APRI-SOVA</td>
<td>MAP</td>
</tr>
<tr>
<td>6</td>
<td>APRI-SOVA</td>
<td>APRI-SOVA</td>
</tr>
</tbody>
</table>

Table 4.2: Various cases studied under simulation.

Figure 4.8: Performance dependence on algorithm. 10k blocks, 10 iterations.
4.2.4 Performance Dependence on Outer Code

A number of different outer codes were tested out in simulation, and the results are shown in Figure 4.9. Recall from Table 4.1 that the (15, 17), (23, 35), and (53, 75) codes have minimum free distances of 6, 7, and 8, respectively. From the graph we can see that small changes in minimum free distance do not appear to have a significant effect on the BER in the region of interest. This suggests that choosing an odd distance code does not appear to be an important design rule for coded GMSK.

Figure 4.9: Dependence on outer code. 10k blocks, 10 iterations.
Chapter 5

Coded GMSK in a Wireless Environment

High-performance, spectrally efficient communication systems are of great interest in wireless applications. In general, wireless systems are characterized by scarcity of spectrum, since additional bandwidth cannot simply be added by installing more lines as in a wireline system. Additionally, wireless channels tend to be more hostile due to multipath fading.

This chapter will investigate performance issues related to communication in a fading environment. There has not been a great deal of investigation to date into the optimization of coded CPM for fading channel performance; in fact, often the same systems designed for AWGN are also used in fading channels without modification, although the channel characteristics are quite different [11]. Hence our first goal is to identify the causes of performance loss in a fading channel. The relationship between the effects of these challenges and the parameters of the channel will also be explored.

Once the causes are identified, we will examine the role of estimation from a detection perspective. Here our goal will not be to investigate the design of a fading channel estimator, which is outside the scope of this thesis, but rather to show how the estimation information should be used in the receiver to gain the most benefit. Additionally, we would like to obtain a characterization of the dependence between the estimation accuracy and the receiver performance, a highly non-linear relationship at best due to the complex algorithms used for demodulation and decoding. Such a characterization would
allow system designers to draw on the wealth of information in the estimation literature in choosing the estimator by giving them an understanding of how the estimator characteristics would affect overall system performance.

5.1 GMSK with Multipath Fading

5.1.1 Fading Channel Model

The fading channel model used in this thesis is the Clarke model [16]. The Clarke model assumes a fixed vertical transmitter which is sending a signal to a mobile receiver. The signal is scattered by buildings and other obstacles, and many copies of the transmitted signal are received, each at different arrival angles. The Clarke model assumes a uniform distribution of arrival angles for a very large number of equal-amplitude signal components. Under these assumptions, it can be shown [21, Ch. 4] that the amplitude of the envelope of the received signal \( r(t) \) is given by the Rayleigh distribution:

\[
p(r) = \begin{cases} \frac{1}{\sigma^2} \exp\left(\frac{-r}{\sigma^2}\right), & 0 \leq r \leq \infty \\ 0, & r < 0 \end{cases}
\]

where \( \sigma^2 = \frac{E_s^2}{2} \).

For an omnidirectional antenna, the in-phase and quadrature fading components have a U-shaped spectrum given by the equation
\[
S(f) = \frac{1}{\pi f_m \sqrt{1 - \left(\frac{f - f_c}{f_m}\right)^2}}
\]  

(5.1)

Here \( f_c \) is the center frequency, and \( f_m \) is the maximum frequency shift due to the Doppler effect caused by the receiver motion. Figure 5.1 shows the shape of the power spectrum.

Figure 5.1: The Clarke model fade power spectrum.

5.1.2 Simulating the Clarke Model

The Clarke model may be implemented in either the time or frequency domain. In this thesis, the time-domain approach was used. The simulation methodology is described in Appendix B.

The parameters for our Clarke model are the Doppler spread (expressed as a fraction of the sampling frequency) and the truncation length of the FIR Doppler filter. These are summarized for various coherence times \( T_c \) in Table 5.1.
5.1.3 Simulation Results

The transceiver system of Chapter 4 was tested with the addition of the fading channel module. As expected, multipath fading degraded the performance of the receiver. From Figure 5.2, we see that the coded GMSK system performed ~2.5 dB worse in the fading channel in the region of interest. The relative performance of the top-performing SISO decoder combinations from Chapter 4 remain unaffected, as seen in Figure 5.3.

In fact, it appears that fading has simply shifted each BER curve to the right. The causes of this performance drop, as well as methods by which some of the lost performance may be recovered, are described in the following sections.

5.2 Mitigating the Fading Effects

When the proposed transceiver is affected by a Clarke fading channel model, the performance loss may be attributed to two effects: distortion and SNR fluctuation. These two effects, along with methods of mitigating them, are described below.
Figure 5.2: Coded GMSK in a fading vs. AWGN channel.

Figure 5.3: Relative performance of cases #1-3 in a fading channel.
5.2.1 Multipath Distortion

When the channel has multipath, the receiver must deal with many superimposed multipath components arriving at the antenna, which results in distortion from self-interference. In the Clarke model, this effect is simulated by multiplying the waveform samples by a random process of non-uniform amplitude, which, in addition to changing the SNR, also distorts the shape of the waveform.

We neglect for now the change in SNR, whose effects we consider in the next section. (Equivalently, we can conceive of normalizing the average fade value in each bit interval to 1.) Performance suffers because the impulse response of the received signal is now time-varying in a random fashion. The ‘matched filter’ is no longer matched to the received signal and therefore does not provide a sufficient statistic. In order to combat this effect, the filter coefficients must be adjusted in a time-varying fashion to track the distortion. This requires information about how the channel is fading.

In the ideal case, we would have perfect knowledge of the fading, and could alter the filter coefficients adaptively to exactly match the faded waveform. If this is the case, then we will show that, given no change in $E_b$, distance properties for the PAM are unchanged by the distortion if there is no ISI. This is in spite of the fact that the set of transmitted signals are distorted by multipath interference. In the case where there is ISI, distance properties will be disturbed depending on the size of an ISI term. For reasonable GMSK memory lengths such as $L=3$, the ISI effect turns out to be insignificant and the results are similar to the non-ISI case.

Consider two bit sequences and the corresponding PAM signals of interest. Define these signals to be
\[ s_1(t) = \sum s_k^{(1)} h(t - kT) , \]
\[ s_2(t) = \sum s_k^{(2)} h(t - kT) . \]

The squared distance between them is:
\[ d_\xi^2 = \int |s_1(t) - s_2(t)|^2 dt . \]

Assume that both signals pass through the same fading channel and are faded by \( f(t) \).

The new signals at the receiver are then
\[ r_1(t) = f(t)s_1(t) , \]
\[ r_2(t) = f(t)s_2(t) . \]

The distance between the faded signals is given by:
\[
\begin{align*}
d_\xi^2 &= \int |r_1(t) - r_2(t)|^2 dt \\
&= \int |f(t)s_1(t) - f(t)s_2(t)|^2 dt \\
&= \int |f(t)|^2 \cdot |s_1(t) - s_2(t)|^2 dt \\
&= \int |f(t)|^2 \cdot \sum_{k=1}^{N} \left| s_k^{(1)} - s_k^{(2)} \right| h(t - kT)^2 dt \\
&= \int |f(t)|^2 \cdot \left\{ \sum_{k=1}^{N} \left| s_k^{(1)} - s_k^{(2)} \right| h(t - kT)^2 \\
&\quad + \sum_{i \neq j} (s_i^{(1)} - s_i^{(2)}) (s_j^{(1)} - s_j^{(2)}) h(t - iT) h^\ast(t - jT) \right\} \end{align*}
\]
If we define a ‘bit difference’ quantity \( d_k^{12} = s_k^{(1)} - s_k^{(2)} \), then the distance becomes:

\[
\begin{align*}
\Delta \xi^2 &= \int |f(t)|^2 \sum_{k=1}^N d_k^{12} |h(t-kT)|^2 \, dt \\
&+ \int |f(t)|^2 \sum_{i \neq j} d_i^{12} d_j^{12} \ b h(t-iT) h^* (t-jT) \, dt \\
= \{ \text{term 1} \} + \{ \text{term 2} \}.
\end{align*}
\]

The first term is a bit difference-dependent term, while the second term is an ISI-dependent term. We will first consider the difference-dependent term, which we shall denote by \( \{ \text{term 1} \} \). \( \{ \text{term 1} \} \) has the form

\[
\{ \text{term 1} \} = \sum_{k=1}^N d_k^{12} \ \int |f(t)|^2 \ |h(t-kT)|^2 \, dt.
\]

Recall that we are considering the constant \( E_b \) case; i.e., for all integer shifts \( k \) we have

\[
\int |f(t)|^2 \ |h(t-kT)|^2 \, dt = E_b = \int |h(t-kT)|^2 \, dt.
\]

Therefore equation (5.3) becomes:

\[
\{ \text{term 1} \} = \sum_{k=1}^N d_k^{12} \ E_b = E_b \sum_{k=1}^N |d_k^{12}|^2.
\]

Next consider the ISI-dependent term, denoted by \( \{ \text{term 2} \} \) in equation (5.2):

\[
\{ \text{term 2} \} = \sum_{i \neq j} d_i^{12} d_j^{12} \ \int |f(t)|^2 \ h(t-iT) h^* (t-jT) \, dt.
\]

64
For the case where there is no ISI, we have

\[ h(t - iT)h^\ast(t - jT) = 0 \quad \forall t \text{ if } i \neq j \]

so \{\text{term 2}\} is zero. For pulses with ISI (such as GMSK), \{\text{term 2}\} perturbs the distance slightly, depending on the magnitude of the integral. Since \( f(t) \) is a random process, this integral cannot be evaluated analytically, but our experience suggests that as a rule of thumb if \( \int h(t - iT)h^\ast(t - jT)\,dt << E_s \) then the effects are negligible.

Therefore, for pulses with small amounts of ISI, the distance of the faded signals is given by \( d_E \approx E_s \sum_{k=1}^{N} |d_k^{12}| \), with exactness in the case where there is no ISI. Following the same approach, we can also show that \( d_E \approx E_s \sum_{k=1}^{N} |d_k^{12}| = d' \). This suggests that, since the distance properties are unchanged, the fading distortion does not have to degrade BER performance if the matched filter coefficients can be adaptively adjusted to match the faded signal.

### 5.2.2 SNR Fluctuation

Because fading is a random process, although the average SNR may be unchanged, the actual SNR will be changing through time. For a fast-fading environment, each bit duration may have a widely different value of SNR. Since BER curves are typically highly non-linear, this has a detrimental effect on performance. To see this, we note that if the SNR is randomly fluctuating, then \( P_e \) in any given bit duration is a random quantity. The average \( P_e \) is given by:
For a nonlinear, `waterfall` BER curve, the poorly-performing terms at low SNR due to \( P(e \mid SNR = x) \) tend to dominate the integral and degrade the performance. In practical terms, this means that in a fast-fading channel, transmitted blocks will be susceptible to single errors (or at least short groups of errors) scattered throughout the block.

### 5.2.3 Compensating for SNR Fluctuation

Since error correcting codes tend to be good at correcting scattered bit errors, it appears likely that systems protected by such codes would be more robust to the effects of SNR fluctuation. We can demonstrate that this is the case by comparing the performance drop due to fading of an uncoded GMSK system (Figure 5.4) versus the proposed coded GMSK system (Figure 5.3). The uncoded system suffers a 2 dB penalty, while the coded system loses only 1 dB. The exact value of this performance loss will depend on the severity of the SNR fluctuation, which in turn depends strongly on the coherence time of the channel. A channel with a longer coherence time will experience long periods of low SNR, which will produce bursts of errors.

Note that both systems still suffer from the waveform distortion due to fading. This accounts for why the coded system still suffers a 1 dB loss. This loss can be overcome by gathering more information about the fading and using an adaptive matched filter which provides a true sufficient statistic.
5.2.4 Compensating for Distortion

As seen in the preceding sections, the effects of distortion can be largely removed if the coefficients of the matched filter can be correctly adapted. This requires knowledge of the fading – the better the knowledge, the more accurately the coefficients may be adapted. Some possibilities include adding side channel information, using adaptive channel estimation, or adaptive equalization.

In this section, we will show that a receiver can make use of side channel information to improve performance in a fading channel. Our adaptive filter should change its coefficients in a time-varying manner to match the faded signal, so that the sampled matched filter output provides the correlation:
matched filter output = \int r(t)r^*(t) dt = \int r(t)f^*(t)s^*(t) dt . \quad (5.5)

If we define a new quantity \( h_{MF}(t) = f(t)s(t) \) to be the matched filter response, then the output becomes:

matched filter output = \int r(t)h_{MF}^*(t) dt . \quad (5.6)

Because \( f(t) \) is randomly time-varying, the matched filter impulse response \( h_{MF}(t) \) must also be randomly time-varying. Therefore, the coefficients of any finite-length filter must be adaptively adjusted.

The implementation of the simulation follows the structure shown in Figure 5.5. The demodulator and iterative decoder remain unchanged. The key modifications are the inclusion of the estimator, and the time-varying adaptation of the matched filter coefficients. The estimator is modeled by a 'genie estimator' with AWGN added to the output. The genie estimator magically produces exact knowledge of the fading by reading the realization of the fading process from memory (i.e. it "cheats"). This exact fading information is then corrupted by AWGN with a given variance to produce a noisy estimator output at Point A in Figure 5.5. The adaptive matched filter takes this information, calculates the effect of the multiplicative fading on the impulse response of the Laurent PAM process, and modifies the filter coefficients accordingly to match the distorted impulse response.

Alternatively, if \( f(t) \) is known through an estimator of some sort, then we may use a static matched filter \( h_{MF}^*(t) = s(t) \) and simply premultiply \( r(t) \) by \( f^*(t) \) at the input of the matched filter. This is mathematically equivalent to the previous structure, as it simply implements equation (5.6) instead of equation (5.5), with \( h_{MF}^* = s(t) \). A block
diagram model of such a receiver is shown in Figure 5.6. This second form was used in computer simulation due to convenience.

The performance of this adaptive receiver is shown in Figure 5.7. The adaptive receiver shows almost a full dB gain over the non-adaptive one. The use of an adaptive matched filter, combined with error-control coding to combat SNR fluctuation, allows the adaptive receiver to perform almost as well in fading as the coded non-adaptive system does in AWGN.
Figure 5.7: Performance of adaptive receiver in a fading channel.

Figure 5.8: The effect of coherence time on performance.
The remaining 0.2 dB loss is attributed to the effects of SNR fluctuation (which is not fully countered by the error-correction code). The exact magnitude of this gap is a function of the coherence time – increasing coherence time will widen the gap, as it will lead to long stretches of low SNR resulting in large bursts of errors, which are more difficult to correct. This effect can be seen in Figure 5.8, which examines the effect of varying the coherence time of the channel. Any method which will reduce the coherence time (such as frequency hopping or time-domain interleaving) will improve the performance of the adaptive receiver in a fading environment.

5.3 The Role of Estimation

We have seen that an adaptive receiver with ideal knowledge of the fading amplitudes can dramatically reduce the gap between fading and AWGN performance. Practically, exact channel knowledge can never be achieved, and the receiver must deal with a distorted estimate of the fading. This section examines the role of practical, suboptimal estimators in the proposed adaptive receiver.

Because of the highly non-linear algorithms involved in demodulation and iterative decoding, we cannot analytically predict the nature of the relationship between the quality of the estimator output and the performance of the overall system. Therefore it will be necessary to characterize the dependence of performance on estimation through simulation. Doing so will also provide some additional insight into the design of the estimator from a detection-driven perspective.
5.3.1 Fade Amplitude vs. Signal Amplitude

After passing through the fading channel, the signal arrives at the receiver with its amplitude distorted by Rayleigh fading and is additionally corrupted by AWGN. The received signal has the form:

\[ r(t) = f(t)s(t) + n(t) \]

Ideally, we would like to incorporate an estimator which produces \( \hat{f}(t) \), an estimate of the fading amplitudes. This is a difficult task because of the multiplicative effect of the signal, which is a time-varying, random unknown nuisance parameter which depends on the data being transmitted. Much of the popular estimation theory has focused on estimating parameters with additive nuisance parameters rather than a combination of additive and multiplicative ones. A simpler alternative is to estimate the faded signal amplitude \( f(t)s(t) \) instead, which only contains additive noise and is typically a good enough approximation to \( \hat{f}(t) \). Empirical results for a coded GMSK system with BT=0.3 showed a mean-square error (MSE) per sample of only 0.0097 between \( f(t) \) and \( f(t)s(t) \), which (as we will see later) is not significant enough to degrade the overall performance of the adaptive receiver.

5.3.2 Suboptimal Estimation

In practice, no estimator is able to deliver perfect knowledge of the channel. Intuitively, we expect that better channel knowledge should translate into better system performance. Here we use the mean-square error (MSE) of the channel estimation as a quality measure, and investigate the performance gain due to the adaptive matched filter in the case where the fade information is imperfect.
Figure 5.9 shows the performances of the adaptive receiver when being fed suboptimal fading estimates with a given per-sample MSE. If this figure is compared with Figure 5.7, where perfect knowledge of the fading is assumed, it is clear that for MSE < 0.05 the performance loss due to the imperfect knowledge is negligible. It is also evident that the chosen estimator should not perform much worse than MSE = 0.2, since beyond this point the loss due to imperfect estimation actually degrades performance compared to the non-adaptive receiver.

![Figure 5.9: Adaptive receiver performance with suboptimal estimation.](image)

Therefore it is evident that the a practical system using the proposed adaptive receiver should employ an estimation scheme which is able to deliver fade knowledge with a per-sample MSE of between 0.05 to 0.2 MSE, with better estimates giving better performance. This rules out simple, non-adaptive schemes such as the Wiener estimator,
which for this channel was heuristically determined to have an MSE of 0.357. It is worth mentioning that for convenience estimators can use signal amplitude $f(t)s(t)$ as an approximation to the fading amplitude $f(t)$, since the 0.0097 penalty in MSE is not significant. Adaptive estimators with fast convergence such as a Kalman filter appear to be better candidates for the fast-fading channel case.

### 5.3.3 Feedback of Soft Information Mean-Square Difference

From the previous investigation, it is evident that the estimation error is very closely tied to the performance of the adaptive receiver. Therefore improving the quality of the estimator output is a very important problem. Many estimators are able to make use of decision feedback to improve their performance [17]. In such cases, the quantity being fed back to the equalizer gives some information about the equalization performance. We would like to identify a similar quantity at the output of the iterative receiver which provides some information about the equalizer MSE.

In a practical receiver with suboptimal estimation, the soft information at the output of the iterative decoder is corrupted by a combination of the additive noise and the estimation error. If the exact fade information is known, then a second set of soft information can be obtained which is affected solely by the additive noise. The mean square difference (MSD) of these quantities then reflects the influence of the estimation error. Here the MSD is defined as follows:

\[
MSD = \frac{1}{N} \sum_{i=1}^{N} \bigg| p(\alpha_i = 0 | r, \text{ optimal estimation}) - p(\alpha_i = 0 | r, \text{ suboptimal estimation}) \bigg|^2
\]

Note that the mean-square difference is calculated at Point B in Figure 5.5.
As the iterative decoding algorithms are highly non-linear, the nature of the dependence (linear or non-linear) is difficult to analytically determine. Figure 5.10 shows results obtained through Monte Carlo simulations of the MSD/MSE relationships at various SNRs. It is apparent from the graph that the relationship is highly linear. This suggests that the equalizer may benefit from a feedback path which provides it with the soft information MSD as a method for evaluating its own performance. In a practical receiver, the soft information resulting from optimal estimation is obviously not available, but in the high performance region (i.e. the low $P_e$ region) could be approximated by the hard decisions, under the standard decision-feedback equalizer (DFE) assumption that the decisions are correct. For the $(23, 35)$ code, this region corresponds to an SNR > 2dB.

![Graph](image)

**Figure 5.10:** Relationship between estimator MSE and decoder MSD.
Chapter 6

Conclusion

6.1 Summary of Results

In this thesis, we examined the issues surrounding the design of a reduced complexity, coded GMSK system. The research was divided into three parts: the design of a reduced-complexity demodulator, the addition of coding and the treatment of coded GMSK as a serially-concatenated code using the modulation memory as the inner code, and the performance of both an adaptive and non-adaptive receiver once multipath fading was introduced.

In the first part, it was concluded that a 4-state linear receiver based on the first pulse of the Laurent representation of GMSK was a good candidate for a reduced-complexity demodulator because it combined good performance with a 4-fold reduction in complexity. It was also discovered that GMSK could be considered as a 'coding' stage representing the modulation memory followed by a Gray-coded PAM process. The distance properties of the PAM were examined, and it was seen that the PAM minimized the number of bit differences between Euclidean-distance nearest-neighbour waveforms. Removing the modulation memory via precoding ensured Gray coding of the input bits and resulted in a twofold improvement in BER.
In the second part, it was shown that a coded GMSK system could be considered a serially concatenated code, with the modulation memory identified in the first part being treated as the inner code. The proposed system showed a significant performance gain over a conventionally-decoded system at no additional cost in bandwidth. The parameters involved (number of iterations, block length) were examined to see their effects on performance. Different SISO algorithms were also tested, with MAP and max-log-MAP both proving to be good candidates. APRI-SOVA performed significantly poorer and was not considered a good choice.

In the final part, the receiver of the second part was examined under fading channel conditions using Clarke’s model. As expected, the performance was poorer. The causes of the drop in performance were identified as waveform distortion and SNR fluctuation, and adaptive matched filtering and coding were proposed as ways of making the system more robust to these effects. The performance of a new, adaptive receiver implementing these changes was examined and shown to be almost as good in fading as the non-adaptive receiver in AWGN.

6.2 Suggestions for Future Work

There remain a number of avenues for further research in this area to pursue. In this thesis, we restricted ourselves to the use of a single error-correcting code. It is possible to add more codes in a serial fashion, such that a ‘true’ serially-concatenated scheme with two or more error-correcting codes is formed. It would be interesting to investigate the question of how the inner code might be optimally combined with the modulation memory to improve performance. Also, we did not look at optimizing the interleaver, although it is well known for SCCs that doing so would improve performance. This is another area that could be explored.
For the fading channel performance, the proposed adaptive receiver abstracted away from the issue of equalizer design. It would be interesting to investigate the question of estimator in a joint fashion, possibly using feedback of soft information MSD. Alternatively, employing some form of channel equalization instead of an adaptive matched filter may also prove fruitful.

Finally, in all cases a very specific form of CPM (i.e. GMSK with $BT=0.3$) was examined. Because most of the results in this thesis could easily be extended to other forms of CPM (since the Laurent models are very similar), it would be worthwhile to apply the same suggestions to other forms of CPM and examine the results.
Appendix A: Deriving The Laurent Pulse

Although [2] gives the method for deriving the Laurent representation, there is no closed-form general expression valid for all CPM schemes. Hence we will derive the Laurent representation for our specific case of GMSK with BT=0.3.

Note: The original Laurent paper uses slightly different notation than later publications. Recall that in this thesis we have used \( \{a_n\} \) to denote the information bits, and \( \{a_{i,n}\} \) to represent the outputs of the various complex encoders. Laurent uses \( \{a_n\} \) instead of our \( \{a_n\} \). To avoid confusion, we will stick to the Laurent notation, and at the end of the derivation we will restate the final result using our notation.

Our starting point is the re-written (exact) complex baseband representation of the CPM signal given in [2, eq. 9], restated here:

\[
s(t) = \sum_{i=N}^{N-1} \prod_{i=0}^{L-1} \left[ S_{i-N} (t) + j^{a_{i-N}} S_{i-N} (t) \right], \quad NT \leq t \leq (N + 1)T \quad (A.1)
\]

where \( N \) is the index of bit of interest, and \( a_N \) is the bit of interest.

The waveforms \( S_n(t) \) are defined to be

\[
S_n(t) = \frac{\sin \left[ \Psi(t + nT) \right]}{\sin \left( \frac{\pi}{1} \right)} = S_0 \left( t + nT \right),
\]
where:

\[ \Psi(t) = \begin{cases} 
q(t), & 0 \leq t < LT \\
\frac{\pi}{2} - q(t), & LT \leq t < 2LT \\
0, & \text{otherwise}
\end{cases} \]

It is conventional to choose \( L = 3 \) for \( BT = 0.3 \), as most of the pulse energy is contained within 3 bit durations. Equation (A.1) therefore becomes:

\[
s(t) = j^{\sum_{n=-\infty}^{\infty} a_n} \times \prod_{i=0}^{2} [S_{r+j-n} (t) + j^{a_g} S_{i+n} (t)] \\
= j^{\sum_{n=-\infty}^{\infty} a_n} \times [S_{r-n} (t) + j^{a_g} S_{r-n} (t)] \times [S_{r-n} (t) + j^{a_g} S_{r-n} (t)] \times [S_{r-n} (t) + j^{a_g} S_{r-n} (t)]
\]

Without loss of generality we can consider \( s(t) \) for \( N = 0 \). Other values of \( N \) are simply time-shifted versions of \( S(t) \) multiplied by different \( j^{a_g} \)'s. For notational simplicity we will also define a new quantity \( s_{dc}(t) = S_1 (t) S_1 (t) S_1 (t) \). The preceding equation then becomes:

\[
s(t) = j^{\sum_{n=-\infty}^{\infty} a_n} \times [S_{1} (t) + j^{a_g} S_{0} (t)] \times [S_{4} (t) + j^{a_g} S_{1} (t)] \times [S_{5} (t) + j^{a_g} S_{2} (t)] \\
= j^{\sum_{n=-\infty}^{\infty} a_n} \times [S_{4} (t) + j^{a_g} S_{1} (t)] \times [S_{5} (t) + j^{a_g} S_{1} (t)] \times [S_{5} (t) + j^{a_g} S_{1} (t)] \\
\]

\[
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) \\
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) \\
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) \\
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) + j^{(a_g, a_g, a_g)} S_{1} (t) \\
\]

\[
= j^{\sum_{n=-\infty}^{\infty} a_n} \times [S_{0} (t+3T) + j^{a_g} S_{0} (t+T) + j^{a_g} S_{0} (t+T) + j^{a_g} S_{0} (t+T) \\
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t+T) + j^{(a_g, a_g, a_g)} S_{1} (t+T) + j^{(a_g, a_g, a_g)} S_{1} (t+T) \\
\quad \quad + j^{(a_g, a_g, a_g)} S_{1} (t+T) + j^{(a_g, a_g, a_g)} S_{1} (t+T) + j^{(a_g, a_g, a_g)} S_{1} (t+T) \\
\]

80
Thus we can see that there are really only 4 unique waveforms: $s_{012}(t)$, $s_{024}(t)$, $s_{045}(t)$, and $s_{015}(t)$, with various delays. We can determine numerically that the main pulse (the one with most of the signal energy) is $s_{012}(t)$. If we define this to be the first PAM signal $s_{L1}(t)$ and collect all terms involving $s_{012}(t)$, then we see that

$$s_{L1}(t) = j^{\sum_{n=0}^{M} a_n} \times \left[ s_{012}(t + 3T) + j^{a_n} s_{012}(t + 2T) + j^{(a_n+a_{n-1})} s_{012}(t + T) + j^{(a_n+a_{n-1}+a_{n-2})} s_{012}(t) \right]$$

Comparing equation (A.2) with [3, eq. 4], we see that with a little rearrangement the above equation becomes the conventional PAM form and gives us the shape of the main pulse $h_0(t)$:

$$s_{L1}(t) = \sum_{N=0}^{M-1} j^{\sum_{n=0}^{N} a_n} s_0(t - NT)s_1(t - NT)s_2(t - NT)$$

where we have assumed that the input bit sequence has length $M$.

As promised, we will restate the final result (A.3) using our notation, which basically replaces $a_n$ with our $\alpha_n$, changes the index of the inner summation (in the exponent of $j$) from $n$ to $i$, changes the index of the outer summation from $N$ to $n$, and replaces the original $M$ (the length of the input bit sequence) with $N$.

$$s_{L1}(t) = \sum_{n=0}^{N-1} j^{\sum_{i=0}^{i=n} a_i} s_0(t - nT)s_1(t - nT)s_2(t - nT)$$

(A.4)
Appendix B: Fading Channel Simulation

In this thesis, the Clarke fading channel outlined in Rappaport [16] was used.

B1. Description of Model

Figure B1 shows a block-diagram representation of the fading channel module. The fading module operates at baseband and produces a sampled version of the fading amplitude using Gaussian noise generators passed through baseband Doppler filters. A step-by-step description of the fading channel module follows.

Figure B1: Clarke fading channel module block diagram.
To implement the fading channel model used in this thesis, the following steps were followed:

1. An N-point, sampled Doppler power spectral density was calculated using the following equation:

   \[
   S_k = \frac{1}{\pi f_m \sqrt{1 - \left(\frac{f_k - f_c}{f_m}\right)^2}}
   \]

2. Since at the edges \(S_k\) is infinite, a practical simulator must truncate to some finite value. For \(k = -N/2, \ldots, N/2\), the edge points are linearly extrapolated from the neighboring two points using the equations

   \[
   S_{\frac{\gamma}{2}} = S_{\frac{\gamma-1}{2}} + \left( S_{\frac{\gamma-1}{2}} - S_{\frac{\gamma-2}{2}} \right)
   \]

   \[
   S_{-\frac{\gamma}{2}} = S_{-(\frac{\gamma-1}{2})} + \left( S_{-(\frac{\gamma-1}{2})} - S_{-(\frac{\gamma-2}{2})} \right)
   \]

3. The filter impulse response was obtained by applying an IFFT on \(S_k\) using the equation

   \[
   \left\{ h_k \right\} = \text{IFFT}\left( \left\{ \sqrt{S_k} \right\} \right)
   \]

4. The filter impulse response is normalized so that the energy in the filter is 1:

   \[
   h_k = \frac{h_k}{\|h\|}
   \]
5. The simulator generates two independent, Gaussian sequences $i_k$ and $q_k$ of variance $\frac{1}{2}$. These sequences are then passed through the filters and the magnitude operation is taken to obtain the fading amplitude:

$$\{r_k\} = \sqrt{(i_k \ast h_k)^2 + (q_k \ast h_k)^2}$$

The factor of $\frac{1}{2}$ in step 5, above, ensures that the mean power of the generated output is 1.

**B2. Doppler Spread**

The Clarke model allows for the amount of Doppler spread as a parameter. The Doppler spread refers to the maximum shift in frequency $f_m$ caused by the motion of the receiver. This maximum Doppler shift is given by:

$$f_m = \frac{v \cdot f_c}{c} \quad \text{(B1)}$$

where:

- $v =$ speed of the receiver
- $f_c =$ carrier frequency
- $c =$ speed of EM propagation

If the carrier frequency is fixed, then for any given value of $f_m$ we can determine what the speed of the mobile must be by simply rearranging equation (B1):
The Doppler shift in the frequency domain is related (via a Fourier transform) to the coherence time in the time domain. The coherence time $T_c$ refers to the time duration over which two samples of the fading show a strong correlation. In general, fading samples which are further apart in time than $T_c$ are considered uncorrelated.

Unfortunately there are many different methods in the literature for quantifying the coherence time. This is because strictly speaking all fading samples are correlated to some degree, but practically after the correlation becomes sufficiently small the samples are considered uncorrelated. Typically these methods differ in how conservative a definition of "uncorrelated" they adopt. [16] and [21] list some commonly used definitions of $T_c$.

In this thesis, we will truncate the autocorrelation of the fading at the first null after which the correlation value has dropped to less than 0.1 of the maximum correlation. The coherence time is then the width of the truncated autocorrelation function. This is illustrated in Figure B2.

**B3. Delay Spread**

The delay spread determines whether the channel is frequency selective. In a frequency selective channel, signal components at different frequencies are faded differently. If the fading is the same over the spectral region occupied by the signal, then the channel is called frequency non-selective or flat. A frequency selective channel can be simulated using multiple Clarke models in conjunction with variable gains and time delays [16].
In this thesis, we restrict ourselves to the flat fading case.

![Graph showing first peak/ trough below 0.09](image)

Figure B2: Determining coherence time from Doppler autocorrelation.
Appendix C: MAP Algorithm Derivation

In this appendix, a form of the MAP algorithm usable for SISO decoding is derived. This derivation is based on details in [11] and [26].

C1. Description of Model

Consider a code which is described by a Markovian state-space model. Each edge in the trellis section is labeled with an information symbol and a codeword symbol. A sequence of N codewords (corresponding to an N-step walk through the trellis) is transmitted. We first summarize the mathematical notation that will be used in this Appendix:

\[
\begin{align*}
\lambda\{u;l\}, \lambda\{c;l\} & = \text{input likelihoods passed to MAP algorithm} \\
\lambda\{u;O\}, \lambda\{c;O\} & = \text{refined likelihoods output by MAP algorithm} \\
d & = \text{an element from the set of information symbols} \\
x & = \text{an element from the set of codeword symbols} \\
Y & = \text{vector of observations } \{y_i\} \\
K_1, K_2 & = \text{normalization constants} \\
(m', m) & = \text{set of states associated with a transition in the code trellis} \\
m & = \text{state at time } k \text{ (i.e. at a depth } k \text{ into the code trellis)} \\
m' & = \text{state at time } k - 1
\end{align*}
\]
\[ u(m',m) = \text{the information symbol causing the state transition } (m',m) \]
\[ c(m',m) = \text{the codeword symbol produced by the state transition } (m',m) \]
\[ \alpha_k(m) = \Pr(s_k = m, y_{isk}) \]
\[ \beta_k(m) = \Pr(y_{isk} | s_k = m) \]

C1.1 The State Transition Probability

The goal of the MAP algorithm is to determine the refined probabilities \( \lambda \{u;O\} \) and \( \lambda \{u;O\} \). These refined probabilities require the calculation of the state transition probability \( \Pr(s_{k-1} = m', s_k = m, Y) \). With some manipulation, we can re-express this quantity:

\[
\Pr(s_{k-1} = m', s_k = m, Y) = \Pr(s_{k-1} = m', s_k = m, Y) \\
= \Pr(s_{k-1} = m', s_k = m | y_{isk}, y_{isk}) \\
= \Pr(s_{k-1} = m', y_{isk}) \cdot \Pr(s_k = m, Y | s_{k-1} = m', y_{isk}) \\
\quad \cdot \Pr(y_{isk} | s_k = m) \\
= \Pr(s_{k-1} = m', y_{isk}) \cdot \Pr(s_k = m, Y | s_{k-1} = m') \\
\quad \cdot \Pr(y_{isk} | s_k = m)
\]

where the last step follows from the fact that given knowledge of the previous state \( s_{k-1} \), knowledge of the observations \( y_{isk} \) provides no additional information (Markov property).

By defining three new quantities
\[ \alpha_k (m) = \Pr(s_k = m, y_{i \leq k}) \]

\[ \beta_k (m) = \Pr(y_{i \leq k} | s_k = m) \]

\[ \phi(m', m) = \Pr(s_k = m, y_k | s_{k-1} = m) \]

we can re-express the state transition property as

\[ \Pr(s_{k-1} = m', s_k = m, Y) = \alpha_{k-1} (m') \cdot \phi_k (m', m) \cdot \beta_k (m) \]  \hspace{1cm} (C1)

C1.2 The Forward Recursion

The quantity \( \alpha_k (m) \) can be obtained via a forward recursion:

\[ \alpha_k (m) = \Pr(s_k = m, y_{i \leq k}) \]

\[ = \sum_{m'} \Pr(s_{k-1} = m', y_{i \leq k}) \cdot \Pr(s_k = m, y_k | s_{k-1} = m', y_{i \leq k}) \]

\[ = \sum_{m'} \Pr(s_{k-1} = m', y_{i \leq k}) \cdot \Pr(s_k = m, y_k | s_{k-1} = m') \]

\[ = \sum_{m'} \alpha_{k-1} (m') \cdot \phi_k (m', m) \]

C1.3 The Backward Recursion

The quantity \( \beta_k (m) \) can be obtained via a backward recursion:
\[ \beta_k (m) = \Pr(y_{i_k} | s_k = m) \]
\[ = \sum_{m'} \Pr(s_{k+1} = m', y_{i_{k+1}} | s_k = m) \]
\[ = \sum_{m'} \Pr(y_{i_{k+1}} | s_{k+1} = m') \cdot \Pr(s_{k+1} = m', y_{i_k} | s_k = m) \]
\[ = \sum_{m'} \beta_{k+1} (m') \phi_{k+1} (m, m') \]

### C1.4 Calculating \( \phi_k \)

The term \( \phi_k (m', m) \) involves both input probability distributions \( \lambda \{u; i\} \) and \( \lambda \{c; I\} \), as well as the code structure itself.

\[ \phi_k (m', m) = \Pr(s_k = m, y_k | s_{k-1} = m') \]
\[ = \sum_s \Pr(s_k = m, c_k = x, y_k | s_{k-1} = m') \]
\[ = \sum_s \Pr(s_k = m | s_{k-1} = m') \cdot \Pr(c_k = x, y_k | s_{k-1} = m', s_k = m) \]
\[ = \sum_s \Pr(s_k = m | s_{k-1} = m') \cdot \Pr(c_k = x | s_{k-1} = m', s_k = m) \cdot \Pr(y_k | c_i = x) \]
\[ = \sum_s \Pr(u_k = u(m', m)) \cdot \Pr(c_k = x | s_{k-1} = m', s_k = m) \cdot \Pr(y_k | c_k = x) \]
\[ = \Pr(u_k = u(m', m)) \cdot \Pr(c_k = c(m', m)) \]

### C1.5 Determining the Output Information Symbol Likelihood

Instead of directly calculating the output information symbol likelihood \( \Pr(u_k = d | Y) \), it will prove easier to first obtain the joint probability \( \Pr(u_k = d, Y) \). Then
\( \Pr\left( u_k = d \mid Y \right) \) can be obtained from the joint probability with a simple normalization operation.

We will define a set \( A \) as the set of state transitions \( (m', m) \) which are caused by information symbol \( d \). Mathematically,

\[
A = \{(m', m): u(m', m) = d\}.
\]

We can then express the joint probability as:

\[
\Pr(u_k = d, Y) = \sum_{(m, m') \in A} \Pr(s_{k-1} = m', s_k = m, Y)
\]

\[
= \sum_{(m, m') \in A} \alpha_{k-1}(m') \cdot \varphi_k(m', m) \cdot \beta_k(m)
\]

where the last step is a substitution using equation (C1). We can then obtain \( \Pr(u_k = d \mid Y) \) as follows:

\[
\Pr(u_k = d \mid Y) = \frac{\Pr(u_k = d, Y)}{\Pr(Y)}
\]

\[
= \frac{1}{\Pr(Y)} \cdot \sum_{(m, m') \in A} \alpha_{k-1}(m') \cdot \varphi_k(m', m) \cdot \beta_k(m)
\]

\[
= \frac{1}{\Pr(Y)} \cdot \sum_{(m, m') \in A} \alpha_{k-1}(m') \cdot \beta_k(m) \cdot \Pr(u_k = u(m', m)) \cdot \Pr(Y \mid c_k = c(m', m))
\]

Recall that definition of \( A \), we must have \( u(m', m) = d \), and therefore the term \( \Pr(u(m', m) = d) \) is independent of \( (m', m) \) and may be pulled out of the summation.
The term $K$ is a normalization constant which ensures that $Pr(u_k = d \mid Y) = 1.

\textbf{C1.5 Determining the Output Codeword Symbol Likelihood}

This derivation is similar to the previous: we will first calculate the joint probability $Pr(c_k = x, Y)$, then use it to obtain $Pr(c_k = x \mid Y)$ via a normalization.

We will define a set $B$ as the set of state transitions $(m', m)$ which cause the output of codeword symbol $x$. Mathematically,

$$B = \{(m', m) : u(m', m) = x\}.$$

We can then express the joint probability as

$$Pr(c_k = x, Y) = \sum_{(m', m) \in B} Pr(s_{k-1} = m', s_k = m, Y)$$

$$= \sum_{(m', m) \in B} \alpha_{k-1}(m') \cdot \varphi_k(m', m) \cdot \beta_k(m)$$
and obtain \( \Pr(c_k = x \mid Y) \) via

\[
\Pr(c_k = x \mid Y) = \frac{\Pr(c_k = x, Y)}{\Pr(Y)}
\]

\[
= \frac{1}{\Pr(Y)} \cdot \sum_{(m', m) \in \beta} \alpha_{k-1}(m') \cdot \varphi_k(m', m) \cdot \beta_k(m)
\]

\[
= \frac{1}{\Pr(Y)} \cdot \sum_{(m', m) \in \beta} \alpha_{k-1}(m') \cdot \beta_k(m) \cdot \Pr(u_k = u(m', m)) \cdot \Pr(Y \mid c_k = c(m', m))
\]

\[
= \frac{1}{\Pr(Y)} \cdot \sum_{(m', m) \in \beta} \alpha_{k-1}(m') \cdot \beta_k(m) \cdot \Pr(u_k = u(m', m)) \cdot \Pr(Y \mid c_k = x)
\]

\[
= \frac{1}{\Pr(Y)} \cdot \Pr(Y \mid c_k = x) \cdot \sum_{(m', m) \in \beta} \alpha_{k-1}(m') \cdot \beta_k(m) \cdot \Pr(u_k = u(m', m))
\]

\[
= K_z \cdot \sum_{(m', m) \in \beta} \alpha_{k-1}(m') \cdot \beta_k(m) \cdot \Pr(u_k = u(m', m))
\]
References


