TIMING AND CARRIER PHASE RECOVERY FOR A BURST ATM SATELLITE TDMA CHANNEL

by

WILSON W. Q. MAI

A thesis submitted to the
Department of Electrical and Computer Engineering
in conformity with the requirements for
the degree of Master of Science (Engineering)

Queen's University
Kingston, Ontario, Canada

October 1999

Copyright © Wilson W. Q. Mai, 1999
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.

0-612-45286-7
Abstract

The recent developments in satellite communications have opened the door to provide high speed B-ISDN to millions of home and small business users. With ATM becoming the standard for future multimedia transmissions, many new satellite networks have been proposed or built to carry ATM transmission. This thesis concentrates on the studies of synchronization techniques suitable for use in home-based satellite modems for transmission on the return link from a terminal to a hub through a satellite repeater. Two new burst timing algorithms based on Gardner's zero-crossing tracker are developed for a Nyquist sampling, 40% roll-off coherent QPSK burst modem. Using a bank of interpolation filters, these algorithms can acquire a timing offset in just 8 preamble symbols. By computer simulations, their BER performance is shown to be very close to the theoretical limit on an AWGN channel for $E_b/N_0$ as low as -3 dB. Two new phase acquisition algorithms are also discussed in this thesis that are suitable for burst transmission and capable of tracking a normalized frequency offset of $f_oT_s = 0.001$ at $E_b/N_0$ as low as 0 dB for uncoded burst transmission. Both phase acquisition schemes work entirely on phasors and are very efficient for DSP implementation. Rate-1/2 convolutional codes are used to improve decoding performance. In this case, synchronization is possible for $E_b/N_0$ as low as 1.5 dB. Techniques of using the convolutional decoder to assist phase acquisition are investigated as well.
I would like to thank my thesis supervisors, Dr. Peter McLane and Dr. N. Manjikian, for their invaluable guidance and encouragement throughout the course of this work. I would also like to thank Dr. M. Landolsi of Nortel Networks for his advice and suggestion of error correction coding in my studies. In addition, I like to thank Communications and Information Technology Ontario (CITO) and Nortel Networks for their financial support through their Research Internship Program. Sincere appreciation should also be given to staff in the CITO office, Mr. Harry Warder and Mrs. Pat Lewis, for their help and support.

I would like to express my gratitude to the faculty and staff of the department for their friendly assistance. In particular, I am indebted to Ken Gracie for answering my questions through emails and conversation. It has been a great pleasure to work with all my colleagues, particularly those in WLH 608, who make my years at Queen’s unforgettable.

Last but not least, I would take this opportunity to thank my family for their generous support and love over the years.
# Contents

1 Introduction ............................................. 1
   1.1 Motivation ........................................... 1
   1.2 System Description .................................. 4
      1.2.1 Transmitter ..................................... 4
      1.2.2 Receiver ....................................... 9
      1.2.3 Modes of Operations ........................... 12
   1.3 Literature Survey ................................... 13
   1.4 Contributions of the Thesis ....................... 15
   1.5 Presentation Outline .............................. 16

2 Burst Timing Acquisition .............................. 18
   2.1 Introduction ........................................ 18
   2.2 Basic Timing Structure ............................ 20
      2.2.1 Timing Error Detector ......................... 20
      2.2.2 Loop Filter ................................... 22
      2.2.3 Interpolation ................................ 24
   2.3 Burst Gardner Algorithm ......................... 34
      2.3.1 System Overview ............................... 34
List of Tables

2.1 Signal assignments for $e_k$ when previous data strobe is reliable. . . . . . . 38
2.2 Signal assignments for the first few sets of $e_k$s. . . . . . . . . . . . . . . . . . 38
4.1 Metrics Assignment for Soft Decision. . . . . . . . . . . . . . . . . . . . . . . . 107
List of Figures

1.1 Transmitter Block Diagram. ................................................. 4
1.2 Gray-Coded QPSK Constellation. ........................................ 5
1.3 Impulse response of the 40% rolloff RRC filter with zero timing offset. 7
1.4 Transmission with a Carrier Frequency. ............................... 8
1.5 Receiver Block Diagram. .................................................... 10

2.1 Effect of Timing Offset. ..................................................... 19
2.2 Block Diagram for Basic Timing Synchronization. .................... 20
2.3 Illustration of Terms in Equation (2.1). ............................... 21
2.4 First-order IIR Loop Filter. ............................................... 23
2.5 Illustration of the use of interpolation in timing correction. ......... 27
2.6 The Raised-cosine pulse in time domain. ............................... 29
2.7 Serial Version of Gardner's Algorithm. ................................ 31
2.8 Structure of the Error Detector Using Gardner's Algorithm. ....... 32
2.9 Interpolation Bank Structure. ............................................. 33
2.10 Timing Acquisition Block Diagram. ..................................... 35
2.11 Output from the interpolation filter bank. ............................ 36
2.12 Distribution of the timing error signal $e_k$. ........................ 39
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.13</td>
<td>Raised Cosine Samples for $h_{RC}(k - \frac{1}{2})$</td>
<td>41</td>
</tr>
<tr>
<td>2.14</td>
<td>Raised Cosine Samples for $h_{RC}(k)$</td>
<td>42</td>
</tr>
<tr>
<td>2.15</td>
<td>Terms in expressions $a$ and $b$</td>
<td>43</td>
</tr>
<tr>
<td>2.16</td>
<td>Structure of the energy accumulator for the filter pair $h_{rrc}(nT_s + \tau_l)$</td>
<td>45</td>
</tr>
<tr>
<td>2.17</td>
<td>The second set of outputs from the interpolation filter bank.</td>
<td>46</td>
</tr>
<tr>
<td>2.18</td>
<td>The flow diagram for the coarse timing acquisition.</td>
<td>47</td>
</tr>
<tr>
<td>2.19</td>
<td>The flow diagram for the fine timing acquisition.</td>
<td>49</td>
</tr>
<tr>
<td>2.20</td>
<td>Flow diagram for the energy check.</td>
<td>51</td>
</tr>
<tr>
<td>2.21</td>
<td>Mean (a) and Variance (b) of the residual timing offset.</td>
<td>53</td>
</tr>
<tr>
<td>2.22</td>
<td>BER plots of the NMS (a) and MS (b) timing algorithms for fixed timing offset.</td>
<td>55</td>
</tr>
<tr>
<td>2.23</td>
<td>Mean of the residual timing offset estimate for a time-varying timing offset at a rate of $0.001T_s$/symbol.</td>
<td>57</td>
</tr>
<tr>
<td>3.1</td>
<td>Demodulation of a pulse-shaped QPSK signal.</td>
<td>61</td>
</tr>
<tr>
<td>3.2</td>
<td>Effect of an phase error $\phi$ on QPSK constellation.</td>
<td>63</td>
</tr>
<tr>
<td>3.3</td>
<td>Block diagram of an analog phase-locked loop.</td>
<td>64</td>
</tr>
<tr>
<td>3.4</td>
<td>Partition of data into distinct data windows.</td>
<td>67</td>
</tr>
<tr>
<td>3.5</td>
<td>Structure of the early window phase acquisition.</td>
<td>68</td>
</tr>
<tr>
<td>3.6</td>
<td>First-order IIR Loop Filter.</td>
<td>71</td>
</tr>
<tr>
<td>3.7</td>
<td>Structure of the late window phase acquisition.</td>
<td>73</td>
</tr>
<tr>
<td>3.8</td>
<td>Phase estimates in data windows.</td>
<td>75</td>
</tr>
<tr>
<td>3.9</td>
<td>Performance comparison of the late window phase acquisition with and without low-pass filtering of the phase estimates.</td>
<td>78</td>
</tr>
</tbody>
</table>
3.10 Mean phase offset residue at $2\pi f_s T_s = 0.001$ for (a) message size = 550 bits and (b) message size = 5500 bits. ........................................ 81
3.11 Symbol error rates of the early data window phase acquisition for various frequency offsets. ........................................ 83
3.12 Mean phase offset residues for various frequency drifts. ....... 85
3.13 Symbol error probabilities of the late data window phase acquisition for various frequency offsets. ............................... 86
3.14 Mean phase offset error for various frequency drifts. ............ 88

4.1 Rate-1/2 convolutional encoder. ....................................... 91
4.2 General convolutional encoder of rate $k/n$. .......................... 91
4.3 State diagram for the rate-1/2 convolutional code. .................... 92
4.4 Trellis diagram for the rate-1/2 convolutional code. .................. 93
4.5 Hard decision quantization. ............................................. 95
4.6 Direct connection of hard decision Viterbi decoder. .................. 95
4.7 Performance of hard decision decoder in Fig. 4.6 under various frequency offsets. .......................................................... 96
4.8 Bit error curves of the rate-1/2, $K = 7$ hard convolutional decoder with perfect phase acquisition. ................................. 97
4.9 Block diagram for using the decoder output to assist phase acquisition. .......................................................... 98
4.10 Modified late window phase acquisition. ............................. 99
4.11 Average of phase estimates. ............................................. 100
4.12 Flow diagram of the phase comparison. ................................ 102
4.13 Bit error rate of the hard decision convolutional code with pilot symbol and quality control on phase estimate. .................... 103
4.14 Relative time relation between the two iterations. 104
4.15 Bit error rate for hard decision convolutional code with iterative decoding. 105
4.16 8-level Uniform Quantization. 106
4.17 Bit error rate of soft decision convolutional code for various quantization level spacing Q. 108
4.18 Bit error rate curve for soft convolutional decoder with perfect phase acquisition. 109
4.19 Bit error rate for soft decision convolutional code with techniques to improve phase acquisition. 110
# Abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPSK</td>
<td>Quadrature phase shift keying</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>B-ISDN</td>
<td>Broadband-Integrated Digital Services Network</td>
</tr>
<tr>
<td>GEO</td>
<td>Geosynchronous Earth Orbit</td>
</tr>
<tr>
<td>AWGN</td>
<td>Addictive White Gaussian Noise</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>RRC</td>
<td>Root-Raised Cosine</td>
</tr>
<tr>
<td>RC</td>
<td>Raised Cosine</td>
</tr>
<tr>
<td>UW</td>
<td>Unique Word</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>A/D</td>
<td>Analog to Digital</td>
</tr>
<tr>
<td>TED</td>
<td>Timing Error Detector</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>NMS</td>
<td>Non-Mixed-Signal</td>
</tr>
<tr>
<td>MS</td>
<td>Mixed-Signal</td>
</tr>
<tr>
<td>ISI</td>
<td>Intersymbol Interference</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-Locked Loop</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage Control Oscillator</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Kbps</td>
<td>Kilo Bit per Second</td>
</tr>
<tr>
<td>MspS</td>
<td>Mega Symbol per Second</td>
</tr>
<tr>
<td>I</td>
<td>In-phase Rail</td>
</tr>
<tr>
<td>Q</td>
<td>Quadrature-phase Rail</td>
</tr>
<tr>
<td>LEO</td>
<td>Lower Earth Orbit</td>
</tr>
<tr>
<td>Mod</td>
<td>Modulo</td>
</tr>
<tr>
<td>IF</td>
<td>Intermediate Frequency</td>
</tr>
<tr>
<td>OQPSK</td>
<td>Offset Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>TMUX</td>
<td>Transmultiplexer</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
</tbody>
</table>
List of Symbols

\( F_m \)  \( m \)th interpolation RRC filter in transmitter
\( \tau_m \)  timing delay of the \( m \)th interpolation filter in transmitter
\( \omega_c \)  carrier frequency
\( \phi_k \)  phase offset for the \( k \)th received symbol
\( x'_I \)  I-rail signal after multiplying with the local oscillator in the receiver
\( x'_Q \)  Q-rail signal after multiplying with the local oscillator in the receiver
\( R(\phi) \)  rotational matrix of \( \phi \) radians
\( x_I \)  I-rail after ideal low-pass filter in receiver
\( x_Q \)  Q-rail after ideal low-pass filter in receiver
\( x \)  ideal low-pass filter output in complex notation
\( r'_I[n] \)  I-rail output from the \( l \)th interpolation filter in receiver
\( r'_Q[n] \)  Q-rail output from the \( l \)th interpolation filter in receiver
\( r''_I[n] \)  timing offset corrected I-rail output selected from the filter bank in receiver
\( r''_Q[n] \)  timing offset corrected Q-rail output selected from the filter bank in receiver
\( E_b/N_o \)  received signal to noise ratio per bit
\( E_s/N_o \)  received signal to noise ratio per symbol
\( E_c/N_o \)  received signal to noise ratio per coded symbol
\( u_I \)  I-rail output of QPSK encoder
\(u_Q\)  Q-rail output of QPSK encoder

\(u_I'[n]\)  upsamplied I-rail output of QPSK encoder

\(u_Q'[n]\)  upsamplied Q-rail output of QPSK encoder

\(u_I''[n]\)  I-rail pulse shaping filter output in transmitter

\(u_Q''[n]\)  Q-rail pulse shaping filter output in transmitter

\(u_I'''[n]\)  I-rail phase rotator output in transmitter

\(u_Q'''[n]\)  Q-rail phase rotator output in transmitter

\(A_k\)  magnitude of the QPSK constellation vector for the \(k\)th symbol

\(Q_k\)  modulation phase of the \(k\)th symbol

\(h_{RC}\)  impulse response of the raised cosine filter

\(h_{RRC}\)  impulse response of the root-raised cosine filter

\(\alpha\)  roll-off factor of raised cosine filter

\(T\)  symbol period

\(N_s\)  number of samples per symbol

\(r_I[k]\)  downsampling timing offset corrected I-rail output

\(r_Q[k]\)  downsampling timing offset corrected Q-rail output

\(r_I^*[k]\)  phase offset corrected I-rail output

\(r_Q^*[k]\)  phase offset corrected Q-rail output

\(F_l\)  \(l\)th interpolation filter in the receiver

\(\tau_l\)  timing delay of the \(l\)th interpolation filter in the receiver

\(M\)  half of the total number of filters in the receiver filter bank

\(\tau_k\)  timing offset of the \(k\)th symbol

\(\tau_k'\)  timing offset estimate of the \(k\)th symbol by the receiver

\(e_k\)  timing error signal for the \(k\)th symbol generated by the receiver
$A$ loop filter coefficient
$B$ loop filter coefficient
$f_{3dB}$ normalized cutoff frequency of a loop filter
$f_c$ cutoff frequency of a loop filter
$f_s$ symbol rate
$x_s(t)$ sampled $x(t)$
$\delta(t)$ Dirac delta function
$X_s(f)$ Fourier transform of $x_s(t)$
$T_s$ sampling period
$h_{ipf}(t)$ impulse response of ideal low-pass filter
$\tau_o$ timing offset between transmitter and receiver
$x'(nT_s)$ pulse-shaping filter output in continuous timing
$\epsilon$ steady state timing error
$W$ normalized cutoff frequency of Nyquist pulse shaping filter
$\tau_{peak}$ peak of vector $r$
$E_l$ energy of the $l$th filter output in the receiver
$\bar{e}_k^l$ the $l$th low-pass filtered timing error signal of the $k$th symbol
$l^*_k$ filter index of timing-offset-corrected interpolation filter
$S_{CE}(t)$ complex envelope relative to $\omega_c$ of the transmitted message
$\omega_cl$ clock frequency of the local oscillator in the receiver
$\overline{D_f}$ normalized Doppler frequency relative to the symbol rate
$R$ symbol rate
$D_f$ Doppler frequency
$r'_k$ $k$th received symbol after removal of early phasor error estimate
$r_k''$  
$k$th received symbol after removal of both the early error phasor estimate and the data estimate

$r_k'''$  
loop filter output of $r_k''$

$B_n e^{j\hat{\phi}_n}$  
phasor error estimate calculated over the $n$th data window

$\hat{\phi}_k$  
phase error estimate for the $k$th symbol

$f_o$  
frequency offset

$N$  
window size in the phase acquisition algorithm

$K$  
constraint length of a convolutional code

$g$  
generator sequence of a convolutional code

$\Gamma$  
decision depth of a convolutional code

$w_k$  
weighting on phase estimate

$Th_{\text{phase}}$  
threshold for phase quality check in the phase acquisition process

xvi
Chapter 1

Introduction

1.1 Motivation

During the past decade, Asynchronous Transfer Mode (ATM) over satellite has been one of the fastest growing areas in the satellite communication field [1, 2]. Numerous new satellite networks, such as Spaceway, Astrolink and Teledesic, are all being built to specifically carry ATM traffic. While ATM is designed for Broadband-Integrated Services Digital Network (B-ISDN) and is widely accepted as the networking technology for the future, its uses are limited to terrestrial networks in which optical networks can be laid. However, it is highly desirable to provide remote users, such as ships at sea or army tanks in battle fields, with high-speed ATM services. This leads to the combination of ATM and satellite to exploit the advantages of satellite communications over terrestrial networks. Some of these advantages are:

- **Large coverage** - Only three GEO synchronous satellites are needed to provide almost world wide coverage (except the polar regions). Therefore, satellites are
an economical way to provide communication networks in remote areas that would otherwise be impossible for terrestrial networks.

- **Flexible inter-networking** - Satellites can be used to provide flexible network connections that include a large number of users from different areas on Earth. Also satellite networks can be easily reconfigured to meet changes in user requirements or network traffic conditions.

- **Broadcast capability** - Satellites readily provide broadcasting services from one service provider to many end users. The end users can also send data or report back to the service provider via satellites. Typical applications are TV broadcasting and video conferencing.

However, these great advantages of ATM over satellite can not be realized without some technical difficulty. The problem of ATM transmission is its small message size. Each ATM cell is only 53 bytes long, and is transmitted independently. Upon receiving an ATM cell, the receiver must be able to quickly recover all of the information related to timing, phase, and frequency of the received signal. Otherwise, it is impossible to demodulate that message. This presents a great challenge in modem design: signal synchronization. It is especially difficult for the short ATM cell due to its unpredictable arrival time.

Signal synchronization can be divided into two problems. One is timing recovery, and the other is carrier or phase recovery. In a fully digital receiver, samples of the received signal are often not taken at the optimal time instants. Timing recovery is used instead to correct this timing error by some digital signal processing operations. On the other hand, phase recovery is needed to combat a slow time-varying phase
difference between the transmitter and the receiver. The source of this phase difference can be either a frequency discrepancy between the oscillators in the transmitter and the receiver, or a Doppler frequency offset introduced by the channel.

In a typical system, these synchronization techniques are intended to be used in data transmissions from a home terminal set-top box to a satellite Hub station through a Ka-band satellite link. The transmission rates for these return-links are 144 kbps, 384 kbps and 2048 kbps. A typical system uses 40% excess bandwidth root-raised cosine pulse shaping, QPSK modulation and rate-1/2 convolutional coding. The ATM cell used for transmission must be modified by adding extra symbols for system synchronization. The satellite link will be represented as an AWGN channel and the impact of amplifier non-linearity in the satellite and the terminal will not be considered in this thesis. In addition, the synchronization techniques in this thesis are only considered for a GEO satellite.

With the above problems in mind, the goals of this thesis are set as the following:

- to develop a burst timing acquisition algorithm that uses the shortest timing preamble possible, but is still capable of working in a very low $E_b/N_o$ range for a coherent QPSK system with the Nyquist sampling rate,

- to develop burst phase acquisition schemes that are simple to implement and effective for low $E_b/N_o$,

- to optimize the synchronization algorithms for an AWGN channel,

- and to integrate and test the synchronization algorithms for use with convolutional codes for operation at low channel SNR.
1.2 System Description

This section provides a brief overview on the transmitter and receiver along with short descriptions of each functional block in the modem structure.

1.2.1 Transmitter

The overall block diagram of the transmitter is shown in Fig. 1.1. The very first step in the transmission is to map the input binary data stream into the appropriate modulation format. Coherent QPSK is used throughout this project because of its better bit error rate performance compared to the differential modulation scheme, and its widespread use in satellite systems. The modulation is performed by the block labeled QPSK Mapper. It takes in two data bits for each symbol interval, and produces the modulation phase according to a Gray coded mapping in Fig. 1.2. The \( k \)th symbol vector can be expressed as a complex baseband signal with the following components:

\[
\begin{align*}
   u_I &= A_k \cos[\theta_k] \quad (1.1) \\
   u_Q &= A_k \sin[\theta_k] \quad (1.2)
\end{align*}
\]
where $A_k$ is the magnitude of the signal vector and $\theta_k$ is the modulation phase for the $k$th symbol. By convention, the term $u_I$ is referred as the in-phase I component, while the term $u_Q$ is called the quadrature-phase Q component.

To obtain maximum immunity to noise and to eliminate intersymbol interference (ISI), each symbol vector should be filtered by a pair of pulse-shaping filters. In this project, the root-raised cosine (RRC) filter is chosen for its popularity in satellite systems and its good performance in suppressing ISI. Interested readers should refer to [3] for a thorough development of its underlying theory. The formula for the RRC filter in the time domain [4] is expressed as follows:

$$h_{RRC}(t) = \begin{cases} 
1 - \alpha + \frac{4\alpha}{\pi} & : t = 0 \\
\frac{\alpha}{\sqrt{2}}\left[(1 + \frac{2}{\pi}) \sin\left(\frac{\pi}{4\alpha}\right) + (1 - \frac{2}{\pi}) \cos\left(\frac{\pi}{4\alpha}\right)\right] & : t = \pm \frac{T}{2\alpha} \\
\frac{\sin\left[\pi(1 - \alpha)\frac{t}{T}\right] + 4\alpha \frac{t}{T} \cos\left[\pi(1 + \alpha)\frac{t}{T}\right]}{\pi \frac{t}{T}\left[1 - (4\alpha \frac{t}{T})^2\right]} & : otherwise,
\end{cases}$$

(1.3)

where $\alpha$ is the rolloff factor bounded by $0 \leq \alpha \leq 1$, and $T$ is the symbol time period. The rolloff factor $\alpha$ determines the extra bandwidth used by the filter output relative...
to that of the filter input. In this project, a 40% rolloff RRC filter ($\alpha = 0.4$) is used to meet the application requirements described in Section 1.1.

Unwanted spectral aliasing can result, however, if symbol vector $u_k$ is directly shifted into a root-raised cosine filter [3, 5]. A solution is to upsample the filter input by a factor of $N_s$, where $N_s$ is the number of samples per symbol. The upsampling can be achieved by attaching $N_s - 1$ zeros to each symbol vector to create a pulse sequence in time. In our system, the Nyquist sampling rate is used, implying that $N_s = 2$.

In an asynchronous communication system, the clocks in the transmitter and receiver are not necessarily synchronized. As a result, the sampling instants of the received message relative to the optimal ones vary from message to message. In simulations of this thesis, a bank of root-raised cosine transmitter filters each with a random timing offset is used to create this varying sampling offset. For a communication system with the Nyquist sampling rate, the timing offset between the transmitter and receiver can never exceed the range of $(-T_s, T_s)$, where $T_s$ is the sampling period. Therefore, the timing offset $\tau_m$ of the interpolation filter in the bank is uniformly distributed between $(-T_s, T_s)$ with a resolution of $\frac{T_s}{32}$ in order to approximate an arbitrary timing offset. This varying timing offset is modeled by randomly selecting one filter from the bank as the pulse-shaping filter in the transmitter. As a result, each message is sent with a different timing offset from the viewpoint of the receiver. Denoting the set of RRC filters as $F_m[n]$, we get:

$$F_m[n] = h_{rrc}(nT + \tau_m) \quad \text{for } m = -31, -30, ..., 0, ..., 32$$

where $\tau_m = \frac{m}{32}T_s$. Also, each RRC filter is chosen to span 5 symbol periods. Therefore, each filter has 10 taps for the case of the Nyquist sampling rate. A typical transmitter
filter impulse response with zero timing offset is shown in Fig. 1.3.

![Graph showing impulse response of RRC filter with zero timing offset.](image)

Figure 1.3: Impulse response of the 40% rolloff RRC filter with zero timing offset.

Denoting the signals after upsampling as $u'_I[n]$ and $u'_Q[n]$ respectively, the outputs from the pulse-shaping filters can be obtained by convolution:

$$u''_I[n] = u'_I[n] * h_{RRC}[n]$$  \hspace{1cm} (1.5)  $$
$$u''_Q[n] = u'_Q[n] * h_{RRC}[n].$$  \hspace{1cm} (1.6)  

In a practical communication system, the filter output vector $u''$ is usually raised to an intermediate frequency $\omega_c$ before entering the channel. At the receiver side, the signal is multiplied with a local carrier reference to convert it to a baseband signal again. The overall process is illustrated in Fig. 1.4.

There is always some mismatch in frequency between the two oscillators in the transmitter and receiver, and this mismatch can be made worse by a Doppler frequency shift introduced by the channel. To see how a received vector is affected by this phase offset, assume the two sources of phase offset are grouped as a single variable $\phi$ and is reflected in the local oscillator in the receiver. Then the received signal
for the I-rail is:

\[ x_I^f(t) = \left\{ u_I'' \cos(\omega_c t) + u_Q'' \sin(\omega_c t) \right\} \cdot \cos(\omega_c t + \phi) \]

\[ = u_I'' \cos(\omega_c t) \cdot \cos(\omega_c t + \phi) + u_Q'' \sin(\omega_c t) \cdot \cos(\omega_c t + \phi) \]

\[ = \frac{1}{2} u_I'' \{\cos(2\omega_c t + \phi) + \cos \phi\} + \frac{1}{2} u_Q'' \{\sin(2\omega_c t + \phi) + \sin \phi\}. \quad (1.7) \]

The low-pass filter in Fig. 1.4 has a cutoff frequency much smaller than \(2\omega_c\). Therefore, all terms with \(2\omega_c\) are filtered out to give:

\[ x_I(t) = \frac{1}{2} u_I'' \cos \phi + \frac{1}{2} u_Q'' \sin \phi. \quad (1.8) \]

Similarly, the filtered output for the Q-rail is:

\[ x_Q(t) = -\frac{1}{2} u_I'' \sin \phi + \frac{1}{2} u_Q'' \cos \phi, \quad (1.9) \]

\(x_I(t)\) and \(x_Q(t)\) can be combined in a complex notation as:

\[ x = \frac{1}{2} u'' e^{-j\phi}. \quad (1.10) \]

Clearly, Equation (1.10) shows that the received signal vector \(x\) is rotated by \(\phi\) from the original message \(u''\).
Baseband simulation is used throughout this thesis. In baseband transmission, there is no intermediate frequency involved. To model the time-varying phase offset discussed above, the block rotation in Fig. 1.1 is used. A rotation matrix $R(\phi)$ is defined as [6]:

$$R(\phi) = \frac{1}{2} \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix}.$$ 

(1.11)

Carrying out the matrix multiplication, we have:

$$x = \begin{bmatrix} x_I \\ x_Q \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} u'_I \\ u'_Q \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} u'_I \cos \phi + u'_Q \sin \phi \\ -u'_I \sin \phi + u'_Q \cos \phi \end{bmatrix}.$$ 

(1.12)

The result is identical to Equations (1.8, 1.9). Therefore, a rotation matrix is used to model a phase offset.

### 1.2.2 Receiver

The structure of the receiver is shown in Fig. 1.5. In this thesis, the channel is assumed to introduce additive white Gaussian noise (AWGN) to the transmitted signal. Therefore, the received signal at the input end of the receiver is:

$$x_i[n] = x'_i[n] + n_c[n]$$

(1.13)

$$x_Q[n] = x'_Q[n] + n_s[n],$$

(1.14)

where $E\{n_c(t)\} = E\{n_s(t)\} = 0$, and $\sigma_c^2 = \sigma_s^2 = \frac{N_0}{2}$.

These baseband signals are then processed by a bank of interpolation filters each with a different timing offset. As will be explained later in Chapter 2, these interpolation filters are actually root-raised cosine filters with different timing offsets.
Figure 1.5: Receiver Block Diagram.
The purpose of these filters is to fulfill the requirement for the overall raised-cosine filter response and to compensate for the possible timing offsets. Defining the set of root-raised cosine (RRC) interpolation filters as $F_l[n]$, we get:

$$\begin{align*}
F_l[n] &= h_{RRC}^l[n] = h_{RRC}(nT + \tau_l) \quad \text{for } l = -(M - 1) \text{ to } M \\
\end{align*}$$

where $\tau_l = \frac{l}{M} T_s$ and $2M$ is the total number of filters in a filter bank, then we can compute the filter output for the $l$th interpolation filter as:

$$\begin{align*}
r_f^l[n] &= x_f[n] * h_{RRC}^l[n] \\
r_q^l[n] &= x_q[n] * h_{RRC}^l[n]. \\
\end{align*}$$

The next step is to select the signal peak at the outputs of the filter bank. This is accomplished by the block timing acquisition in Fig. 1.5 which will be discussed in Chapter 2. Once the best filter output is chosen, decimation by a factor of $N_s$ is performed by the on-off switches. The time-corrected signals $r_f[k]$ and $r_Q[k]$ will then go into the block phase acquisition in Fig. 1.5 to remove the phase offset $\phi$. The details of phase acquisition will be discussed in Chapter 3. Please note that a rough estimate of the carrier frequency $\omega_c$ in Fig. 1.5 is assumed to be available to the receiver, and the process of obtaining it is beyond the scope of this thesis.

Once the phase offset is removed, the output $r^*$ from the phase acquisition process is decoded by a QPSK decoder as shown in Fig. 1.5. The decoding process is essentially the reverse of the QPSK mapper. Firstly the quadrant of the signal vector $r^*$ is determined. The decoded bits are then assigned based on the Gray coded mapping in Fig. 1.2.
1.2.3 Modes of Operations

Depending on the types of binary bits sent, two modes of operation can be identified in a transmission. At the beginning of a transmission, the transmitter sends a set of predetermined symbols to the receiver to assist the initial synchronization of the receiver. These symbols are called the training symbols for serial transmission or the preamble for burst transmission. In our application, this preamble is chosen to be an alternating "00" and "11" sequence. The reason is that these two symbols are separated by 180° in the QPSK constellation and the alternating pattern provides a regular zero-crossing that is best suited for the timing acquisition algorithms developed in this thesis. The predetermined symbols in the preamble also help the initial phase estimation because the phase acquisition algorithm used is decision-directed in nature. The first few symbols have to be correctly detected in order for the method to operate correctly.

It is also vital for the receiver to be able to determine the end of the preamble. The conventional method is to attach a unique-word (UW) that has special correlation properties to assist the detection of the end of the preamble by correlation. For the sake of simplicity, the simulations performed for this thesis do not use a UW. Instead, the receiver is assumed to be turned on as soon as the first valid bit is put onto the channel.

The second mode of operation is the data mode. At the end of the preamble, the receiver is assumed to achieve the initial synchronization. Data symbols can then be transmitted immediately after the end of the preamble. Since the timing and phase offset can be time-varying, some fine synchronization procedures might be needed during the data mode.
1.3 Literature Survey

The burst timing algorithms developed in this thesis draw upon several sources. The zero-crossing timing error signal formula employed by the algorithms was first introduced by Gardner in his report to the European Space Agency [7]. The use of an interpolating filter for timing correction was again proposed by Gardner in two of his papers [8, 9]. McLane, Choy and Tay employed the timing detection method in their 40% excess bandwidth modem system [10]. Belanger [11], Gracie [5] and Koblents [12] then implemented the zero-crossing timing error detector using interpolation filters in their real-time DSP modems. Their implementations were however only intended for serial transmission where a long time training sequence is feasible. Nevertheless, their work serves as a starting point for this thesis.

There is a lack of algorithms for burst timing acquisition in the literature. The idea of searching for the best timing offset among a bank of interpolation filters, each with a different timing offset, was studied by Majeed [13] for a satellite transmultiplexer (TMUX) using OQPSK. There are many references on digital timing acquisition for stream transmission. For instance, the papers by Park [14], Funderburk et al [15], and Ling [16] address the timing correction problem in a slightly different context, namely via sample-rate conversion. The books by Mengali and D’Andrea [17] and Meyr et al [18] are two of the latest on synchronization techniques, and are used for supplementary references in this thesis.

Books and papers covering the design and analysis of phase recovery algorithms are widely available in the literature [19, 17, 7, 20]. For example, the book by Mengali and D’Andrea [17] gives a treatment on general phase-locked loop theory of operation, while the book by Bingham [19] approaches the subject for practical considerations.
The report by Gardner [7] is also an invaluable source for issues related to implementation of phase synchronization as a functional block of a modem. The paper by Viterbi and Viterbi [20] proposes a classical phase estimation method for mPSK that uses a nonlinear transformation and a power law to remove the ambiguity from modulation. Conversions between the rectangular and polar constellation are needed for every symbol. In contrast, the phase acquisition algorithm used in this thesis works entirely on phasors. Hence, no constellation conversion is necessary. The algorithm is derived from the works by Gracie [5], Belanger [11], Belanger and McLane [21], and McLane [22]. Phase acquisition usually involves two stages. The first stage is the coarse frequency estimation which tries to match the carrier frequency in the transmitter and receiver. Summaries and comparisons of several coarse frequency acquisition techniques suitable for mPSK constellations can be found in the tutorial review by Morelli and Mengali [23]. The second stage in phase acquisition deals with the left-over frequency offset from the first stage. The phase acquisition algorithms developed in this thesis are all for the second stage which is often referred as fine carrier acquisition. The integration of the two stages of phase estimation is discussed in the paper by Mazzenga, Luglio, and Vatalaro [24].

References on convolutional encoder and Viterbi decoding are common. In particular, the books by Wicker [25] and Proakis [3] cover the theory of and the encoding/decoding procedures for convolutional codes. The computer programs for convolutional code used in this thesis were written by Karimi [26]. References of bit error rates used for performance comparison in this thesis are primarily taken from the paper by Heller and Jacobs [27], but their results are not available for $E_b/N_o$ below 3 dB. Therefore, the original computer program from Karimi [26] was executed
to obtain the required data points for both the hard and soft Viterbi decoders.

Many other references were used during the project. For example, the books by Proakis [3] and Oppenheim and Schafer [28] give the theoretical background on transmission and coherent detection of QPSK. On the other hand, the book by Ifeachor and Jervis [29] covers more practical implementation issues. Wu [30], and Miller et al. [31] provide the necessary understanding for satellite communication. The rest of the references are Ziemer and Tranter [32] on communications background, and Bishop et al [33] on feedback control system.

1.4 Contributions of the Thesis

- Two new, low-complexity, and effective burst timing acquisition algorithms were developed and tested by simulations. They are suitable for QPSK burst transmission with only 8 symbols of timing preamble and a sampling rate of two samples per symbol. By using a bank of interpolation filters, the timing algorithms are able to simultaneously monitor all the timing error signals associated with each filter in the filter bank, and to quickly choose the one with the optimum timing offset. The effectiveness of these timing algorithms is demonstrated by their ability to successfully estimate a timing offset at $E_b/N_0$ as low as $-2$ dB with only a small loss in performance.

- The timing acquisition algorithms can be made significantly more robust by checking the energy of the samples from the chosen interpolation filter at the end of the timing acquisition. This energy check is able to identify the possible "false" timing lock at the zero-crossing point of the received signal in which the
energy of the filter output is significantly lower than expected.

- Two simple burst phase acquisition algorithms were derived from a previously known method. The algorithms work entirely on phasors and are both very efficient for DSP implementation. Compared to the original method, additional low-pass filters are used in the new algorithms to minimize the adverse effects of erroneous symbol detection on the quality of the phase offset estimate. As a result, the new phase acquisition algorithms can combat a time-varying phase offset at a much lower range of $E_b/N_0$ than the original method.

- A collection of techniques for using the Viterbi decoder to assist phase acquisition were discussed. These techniques enable the more reliable symbol estimates from the Viterbi decoder to be used in the decision-directed phase acquisition, ensuring proper phase acquisition at a low SNR range.

1.5 Presentation Outline

Chapter 2 starts with a brief discussion on the importance of timing recovery to coherent symbol detection. A description of Gardner's zero-crossing timing error signal formula is given along with a simple time tracking algorithm for serial transmission. This is followed by the two new burst timing acquisition algorithms using an interpolation filter bank. The use of an interpolation filter bank to speed up the timing acquisition process is first discussed. The steps of the two timing algorithms are then explained and analyzed. Other design issues such as the length of the timing preamble and the detection of a "false" timing lock are also discussed. Finally, the performance of the two algorithms is accessed in both the timing acquisition mode and the data
Chapter 3 deals with the phase acquisition. It begins with a demonstration of how a phase error can degrade the coherent demodulation of a QPSK signal. Two phase acquisition algorithms suitable for burst transmission are presented. The failure of the algorithms due to poor symbol detection is analyzed. The idea of using a low-pass filter to combat these erroneous symbol estimates is then described. Finally, the performance of both phase acquisition algorithms under various frequency offsets is evaluated by simulations.

Chapter 4 explains the convolutional encoder and the Viterbi decoder algorithm. It then describes a scheme for directly combining the Viterbi decoder and the synchronization schemes, concluding that poor performance can result if they are not properly integrated. The poor performance results from the failure of the phase acquisition algorithm at low SNR. Several techniques for using the more reliable symbol estimates from the Viterbi decoder are then discussed in order to assist the phase acquisition.

Finally, Chapter 5 contains the conclusions of this thesis and suggestions for further studies.
Chapter 2

Burst Timing Acquisition

2.1 Introduction

Synchronization plays an important role in digital communication systems. To achieve synchronous demodulation, a receiver has to ensure the samples used for data detection are relatively free of both timing and phase offsets. This chapter focuses on the issues of timing offset detection and correction. When a message is transmitted over a channel, not only the message is corrupted with noise, but also the sampling time of the received signal by the receiver can vary unpredictably when compared to the optimal sampling instant. The effect of a symbol timing offset on data detection is illustrated in Fig. 2.1. A small timing offset $\tau$ shifts the sampling points (stars on plots of Fig. 2.1) away from the optimal detection points (circles on Fig. 2.1), reducing the effective signal strength. In a Nyquist system, a timing offset also causes intersymbol interference that can adversely affect the probability of error. Therefore, the objective of timing synchronization is to estimate $\tau_k$ based on the received message $x[k]$ from the A/D converter. In the remainder of this chapter, we will present
Figure 2.1: Effect of Timing Offset.
methods for timing synchronization for burst transmission.

2.2 Basic Timing Structure

For a fully digital receiver, timing synchronization can be divided into five different functional parts [17] as shown in Fig. 2.2. The matched filter is used to maximize the output signal-to-noise ratio (SNR) in an AWGN channel. The interpolation filter is used to correct a detected timing offset $\tau'$ from the received signal. TED stands for timing error detector, and is used to generate an error signal $e_k$ proportional to the difference between the true time offset $\tau$ and the current estimate $\tau'$. The loop filter is a low-pass filter used to suppress noise in the timing error estimate $e_k$.

![Figure 2.2: Block Diagram for Basic Timing Synchronization.](image_url)

2.2.1 Timing Error Detector

As stated above, a timing error detector has to be able to generate a signal proportional to the difference between the current estimate of timing offset $\tau'$ and the
true timing offset $\tau$. A timing recovery scheme, introduced by Gardner [8], has been examined and successfully implemented in DSP-based modems by Belanger[11], Koblents[12] and Gracie[5] for serial transmission. The QPSK version of Gardner’s algorithm in this thesis is found to be both robust and efficient, and the timing error measurement for this thesis can be stated mathematically as

$$e_k = r_I[k - \frac{1}{2}] * (r_I[k] - r_I[k - 1]) + r_Q[k - \frac{1}{2}] * (r_Q[k] - r_Q[k - 1]),$$  \hspace{1cm} (2.1)$$

where $r_I(k)$ and $r_Q(k)$ are points on the I and Q rails shown in Fig. 2.3.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{Figure2.3.png}
\caption{Illustration of Terms in Equation (2.1), where $R[k]$ denotes $r_I(k)$ or $r_Q(k)$.}
\end{figure}

Fig. 2.3 shows the outputs of a filter with 4 samples/symbol. The sampling rate can vary. The minimum number of samples per symbol that Gardner’s equation requires is two. Clearly from Fig. 2.3, when the time offset $\tau$ is zero, the mid-baud terms $r_I[k - \frac{1}{2}]$ and $r_Q[k - \frac{1}{2}]$ should be around the zero crossing point, while $r_I[k]$, $r_I[k - 1]$, $r_Q[k]$ and $r_Q[k - 1]$ represent the peaks on the I and Q curves. Thus,
Equation (2.1) should give a value close to zero for $\tau = 0$. On the other hand, a non-zero $\tau$ will push both $r_I[k - \frac{1}{2}]$ and $r_Q[k - \frac{1}{2}]$ away from the zero crossing point along the curve. Moreover $R[k - 1]$ and $R[k]$ are opposite in sign for an alternating source sequence, making $(R[k] - R[k - 1])$ large. With a proper timing preamble, both I and Q rails should have the same signal difference of $(R[k] - R[k - 1])$ at any time. Therefore, $e_k$ is dramatically increased for a non-zero timing error, and can be used as a good indicator for the timing offset.

One important benefit of using Gardner's method is its immunity to constant phase offsets. Belanger [11] has found that the effect of a class of time-varying phase offsets on the error detector is negligible. For example, a frequency offset of 120 Hz at a symbol rate of 2400 bps can introduce a significant $18^\circ$ phase difference between two consecutive symbols. However, Belanger [11] calculated that the magnitude of $e_k$ is only scaled by $\cos(9^\circ) = 0.98769$. In a system with higher transmission rate, the scaling factor approaches 1, resulting in negligible change in $e_k$. Therefore, it is possible to address the problem of timing synchronization first, and then estimate the phase and frequency offsets in the received signal.

### 2.2.2 Loop Filter

The loop filter shown in Fig. 2.2 is an important part in a timing recovery algorithm. Normally, a low-pass filter is used to suppress noise in the timing error signal $e_k$. Results in the experiments by Belanger [11] show that a first-order infinite impulse response (IIR) filter is sufficient for that purpose. The block diagram of the IIR is given in Fig.2.4. Mathematically, the filter output can be expressed as

$$y[n] = Bx[n] + Ay[n - 1].$$  \hspace{1cm} (2.2)
By taking the Z-transform of Equation (2.2), the transform function of the IIR filter is obtained as:

$$H(z) = \frac{B}{1 - Az^{-1}}.$$  (2.3)

The filter has a pole at $z = A$. Since $x[n]$ is a real number, the variables $A$ and $B$ are real numbers as well. Also, the variable $A$ must reside in the interval $(-1,1)$ for this filter to be stable. The cutoff frequency, $f_c$, is the frequency at which the magnitude of the filter output reduces to only one half of that of the filter input. A normalized cutoff frequency, $f_{3dB}$, can also be obtained using the symbol rate $f_s$ as $f_{3dB} = \frac{f_c}{f_s}$.

$f_{3dB}$ is controlled by the variable $A$ as derived in [34]:

$$A^2 + 2A(\cos(2\pi f_{3dB}) - 2) + 1 = 0.$$  (2.4)

The parameter $B$ controls the filter gain. For simplicity, the filter gain is made equal to unity at DC by choosing $B = 1 - A$.

Since the first-order IIR filter is found to be good for suppressing noise of the timing error estimate based on the previous results [11], we will not attempt to optimize the loop filter selection, and will use this first-order IIR filter throughout this thesis.
2.2.3 Interpolation

An all digital receiver does not have the capability to alter the sampling time of its A/D converter. As a result, symbol timing recovery must be performed asynchronously on the received samples, and is usually achieved by interpolation.

Basic Interpolation Formula

According to the Nyquist sampling theorem, a continuous time signal whose bandwidth is limited to $1/(2T_s)$ can be perfectly reconstructed if the signal is sampled at a rate greater than or equal to $1/T_s$. Let $x(t)$ be a signal that is band-limited to $1/(2T_s)$. Samples of $x(t)$ at period of $T_s$ can be expressed as [35]:

$$x_s(t) = \begin{cases} x(t) & \text{for } t = nT_s, n = 0, 1, \ldots, \infty \\ 0 & \text{otherwise.} \end{cases} \quad (2.5)$$

$x_s(t)$ can be approximately expressed in the form

$$x_s(t) = \sum_{n=0}^{\infty} x(t) \delta(t - nT_s) \quad (2.6)$$

where $\delta(t)$ is the Dirac delta function. The Fourier transform of this sampled signal $x_s(t)$ has an interesting relation with the Fourier transform of the original signal $x(t)$.

$$X_s(f) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} X(f + \frac{n}{T_s}), \quad (2.7)$$

where $X(f)$ is the Fourier transform of $x(t)$.

From Equation (2.7), we can see that the spectrum of $x_s(t)$ consists of periodic repetitions of the scaled spectrum $1/T_s X(f)$ located at $f = \pm 1/T_s, \pm 2/T_s, \ldots, \infty$. Therefore, if $x(t)$ is band-limited to $1/(2T_s)$, $x(t)$ can be reconstructed by passing
\( x_s(t) \) through an ideal low-pass filter.

\[
x(t) = x_s(t) * h_{lpf}(t) = x_s(t) * \text{sinc}\left(\frac{\pi t}{T_s}\right).
\] (2.8)

Substituting Equation (2.6) into Equation (2.8) gives,

\[
x(t) = \left\{ \sum_{n=0}^{\infty} x(t)\delta(t - nT_s) \right\} * h_{lpf}(t) = \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} x(\alpha)\delta(\alpha - nT_s) h_{lpf}(t - \alpha) d\alpha
\]

\[
= \sum_{n=0}^{\infty} x(nT_s) h_{lpf}(t - nT_s)
\]

\[
= \sum_{n=0}^{\infty} x(nT_s) \text{sinc}\left(\frac{\pi(t - nT_s)}{T_s}\right). \quad (2.9)
\]

If sampling at time instant \( nT_s + \tau_o \), where \( \tau_o \) is the timing offset between the transmitter and receiver, Equation (2.9) can be rewritten as:

\[
x(nT_s + \tau_o) = \sum_{k=-\infty}^{\infty} x(kT_s) \text{sinc}\left(\frac{\pi(nT_s + \tau_o - kT_s)}{T_s}\right)
\]

\[
= \sum_{k=-\infty}^{\infty} x(kT_s) \text{sinc}\left(\frac{\pi((n-k)T_s + \tau_o)}{T_s}\right)
\]

\[
= x(nT_s) * \text{sinc}\left(\frac{\pi(nT_s + \tau_o)}{T_s}\right). \quad (2.10)
\]

Using \( h_{lpf}(t) = \text{sinc}\left(\frac{\pi t}{T_s}\right) \), \( x(nT_s + \tau_o) \) can be rewritten as:

\[
x(nT_s + \tau_o) = x(nT_s) * h_{lpf}(nT_s + \tau_o). \quad (2.11)
\]

Equation (2.11) shows that a set of samples \( x(nT_s) \) can be shifted by \( \tau_o \) by using a low-pass filter sampled at \( t = nT_s + \tau_o \). Conversely, a timing offset \( \tau_o \) in the received samples can be removed by passing those samples to a low-pass filter sampled at
Mathematically this is represented as:

\[
x(nT_s) = x(nT_s) * h_{ipf}(nT_s)
= \sum_{k=-\infty}^{\infty} x(kT_s)h_{ipf}((n - k)T_s - \tau_o)
= x(nT_s + \tau_o) * h_{ipf}(nT_s - \tau_o).
\]  

(2.12)

The operation of Equation (2.12) is better illustrated by Fig. 2.5. A signal is sent onto the channel with a time offset of \(\tau\) between the sender and receiver. The samples received by the receiver are shown as dots on part (a) of Fig. 2.5. To correct this timing offset, the receiver uses an interpolation filter with a timing offset of \(-\tau\), whose impulse response is plotted on part (b). The resultant interpolation filter output is shown on part (c), and the timing offset is perfectly compensated.

However, a limitation of Equation (2.12) is the infinite impulse response of the ideal low-pass filter used in interpolation. In practice, a truncated version has to be used, introducing distortion in the filter output. A much better way to realize timing recovery through interpolation is by using a method called filter coefficient interpolation\[7\], which has been successfully implemented and tested by Koblents\[12\], Hung\[36\] and Belanger\[11\]. The method will be described in the coming sections.

Filter Coefficient Interpolation

In a band-limited channel, intersymbol interference becomes a serious problem, and is usually tackled by using root-raised cosine pulse-shaping filters in both the transmitter and receiver. Therefore, the interpolation filter output from Equation (2.12) needs to pass through a root-raised cosine filter response in order to achieve the overall
Figure 2.5: Illustration of the use of interpolation in timing correction.
Nyquist pulse-shaping requirement. Let the output of the pulse-shaping filter in the receiver be $x'(nT_s)$, then:

$$
x'(nT_s) = \{x(nT_s + \tau_o) * h_{ipf}(nT_s - \tau_o)\} * h_{rrc}(nT_s)
= x(nT_s + \tau_o) * \{h_{ipf}(nT_s - \tau_o) * h_{rrc}(nT_s)\}.
$$  \hspace{1cm} (2.13)

According to the ideal low-pass filter equation $h_{ipf}(nT_s - \tau_o) = \text{sinc}(nT_s - \tau_o)$, we have:

$$
h_{ipf}(nT_s - \tau_o) * h_{rrc}(nT_s) = \text{sinc}(nT_s - \tau_o) * h_{rrc}(nT_s)
= \left\{\frac{\sin(\pi(nT_s - \tau_o))}{\pi(nT_s - \tau_o)}\right\} * h_{rrc}(nT_s)
= h_{rrc}(nT_s - \tau_o).
$$  \hspace{1cm} (2.14)

Substituting Equation (2.14) back to equation (2.13), we get:

$$
x'(nT_s) = x(nT_s + \tau_o) * h_{rrc}(nT_s - \tau_o),
$$  \hspace{1cm} (2.15)

where $x(nT_s + \tau_o)$ are the A/D converted samples with a timing offset $\tau_o$, and $h_{rrc}(nT_s - \tau_o)$ is the root-raised cosine filter response with a timing offset of $-\tau_o$.

According to Equation (2.15), a timing offset of $\tau_o$ can be corrected by passing those A/D samples to a root-raised cosine filter with time offset of $-\tau_o$ in the opposite direction. Due to the smooth characteristics of the raised cosine spectrum, practical root-raised cosine filters can be readily designed for the transmitter and receiver to closely approximate the overall responses of the raised cosine function in Fig. 2.6.

In reality, the timing offset between the transmitter and receiver is rarely fixed. A small frequency discrepancy usually exists between the two, and will change the timing offset continuously. As a result, the receiver has to keep on updating its filter coefficients of $h_{rrc}(nT_s + \tau)$ in order to cope with this varying time offset $\tau$. The
computation required for this task is prohibitive in most of the cases. A solution is to store a set of coefficients of \( h_{rrc}(nT_s + \tau) \) for different timing offsets in memory. In fact, only timing offsets in the range \((-T_s, T_s)\) need to be considered for a system with a Nyquist sampling rate because any other offset \( \tau \) can be mapped back to the \((-T_s, T_s)\) interval. Defining the set of root-raised cosine (RRC) interpolation filters as \( F_l[n] \), we get:

\[
F_l[n] = h_{rrc}(nT_s + \tau_l) \quad \text{for } l = -(M-1) \ldots M
\] (2.16)

where \( \tau_l = \frac{l}{M}T_s \), and from [4], the formula of root-raised cosine function is:

\[
h_{rrc}(t) = \begin{cases} 
1 - \alpha + \frac{4\alpha}{\pi} & : t = 0 \\
\frac{\alpha}{\sqrt{2}}[(1 + \frac{2}{\pi})\sin(\frac{\pi}{4\alpha}) + (1 - \frac{2}{\pi})\cos(\frac{\pi}{4\alpha})] & : t = \pm\frac{T_s}{2\alpha} \\
\frac{\sin[\pi(1 - \alpha)\frac{t}{T_s}]}{\pi\frac{t}{T_s}} + 4\alpha\frac{\alpha}{\pi}\cos[\pi(1 + \alpha)\frac{t}{T_s}] & : \text{otherwise}
\end{cases}
\] (2.17)

with \( \alpha \) being the excess bandwidth, and \( M \) being the factor of interpolation. Each filter \( F_l[n] \) corresponds to a distinct timing offset \( lT_s/M \). Given a received signal
with a time offset of \( \tau_o \), the filter with the timing offset closest to \(-\tau_o\) is used in the receiver for both pulse shaping and timing correction. Since \( \tau \) is a discrete value, the incoming timing offset \( \tau_o \) might not be perfectly corrected. The difference between the chosen \( \tau \) and \( \tau_o \) can be made arbitrarily small by increasing \( M \). This increase in timing resolution however also increases the number of interpolation filters that need to be stored in memory. Balancing the two, the timing recovery scheme in this thesis uses a filter bank of 8 filters with \( M = 4 \), giving a timing resolution of \( \tau = \frac{1}{4} T_s \).

One limitation of the interpolation filter bank has to be addressed before it can be used in an implementation. The timing offsets represented in the filter bank are bounded by \((-T_s, T_s)\). Hence, the method is unable to correct continuously growing or decreasing timing offsets that exceed the bound. Ideally, the number of filters in the bank should be infinite. Clearly, this is impossible to implement. An elegant solution[12] is to add or drop a sample when the receiver needs to catch up with the varying timing offset. This method has been verified in implementations by Belanger[11], Koblents[12] and Gracie[5].

When adding or dropping a sample, the current interpolation filter has to be at one end of the filter bank. The dropping of a sample is done by shifting one less sample into the filter buffer before decimation, effectively decreasing the time decimation factor by one. The dropping of a sample is needed if the receiver clock has a faster frequency than the transmitter clock. The receiver has to periodically drop a sample to avoid getting ahead of the transmitter. On the other hand, the adding of a sample is accomplished by taking the filter output with one more A/D sample shifted into the buffer, effectively increasing the time decimation factor by one. The adding of a sample is used by the receiver to keep up with the transmitter if the clock in the
receiver is slower than the one in the transmitter. Thus we see that the adding and 
dropping of samples compensate for a difference in timing frequency between a sender 
and receiver.

![Diagram of Gardner's Algorithm]

Figure 2.7: Serial Version of Gardner's Algorithm.

The general timing synchronization scheme can be summarized by Fig. 2.7. The 
received samples are processed by the chosen interpolation filter. The filter outputs 
are then used to generate a timing error signal. If the filtered timing error signal 
exceeds the threshold, a timing correction will be taken by changing the active in-
terpolation filter in the filter bank. If necessary, the adding or dropping of a sample 
is taken by controlling the closing time of the switch. Fig. 2.8 shows the detailed 
structure of the timing error detector for QPSK according to Equation (2.1). Each 
delay element in the diagram causes a delay of one sample period in time. This is 
the timing recovery scheme used in the implementations by Koblents[12], Hung[36] 
and Gracie[5] for serial transmissions, and can also serve as a performance reference 
to the new timing algorithms developed in this chapter.

However, the timing offset estimate $\tau'_o$ using the scheme in Fig. 2.7 can only ap-
proach the true $\tau_o$ gradually by changing from one set of filter coefficients to another.
Figure 2.8: Structure of the Error Detector Using Gardner's Algorithm.

The method has been verified to work well for tracking a slowly varying time offset in serial transmissions on real-time DSP-based modems\[5, 12\]. However, when the receiver is first turned on, it has no knowledge of the true timing offset, and thus, it processes the first few received samples using the interpolation filter with zero time offset, and gradually moves toward the true timing offset. Therefore, the time required to achieve the initial timing lock varies depending on the true timing offset. In the worst case, the difference between $\tau_o$ and $\tau'_o$ can be as much as $T_s$, where $T_s$ is the Nyquist sampling period. The receiver then has to move from one end of the filter bank to the other to get a timing lock. The potential long seeking time for timing lock is not a serious problem in serial transmission where the payload is typically much longer compared to the timing preamble. But the method is ill suited for use in burst transmission where the preamble represents a significant percentage of the overall message. A faster method suitable for burst transmission has to be found.
Figure 2.9: Interpolation Bank Structure. Filters are indexed by \( l \) from \(-3, -2, \ldots, 4\), with respective time delays from \(-0.75T_s, -0.5T_s, \ldots, 0.75T_s, T_s\), where \( T_s \) is the sampling time.

The solution is to use a bank of interpolation filters as shown in Fig. 2.9. Each filter in the bank follows Equation (2.16). Thus, it is possible to compute the timing error signals for each different timing offset based on the corresponding filter outputs. By simulations, it is found that very good performance can be achieved by a filter bank of 8 filters with \( M = 4 \). The increase in the number of filters in the filter bank increases not only the memory and computational requirements, but also the average number of filter transitions needed to reach the best filter in the interpolation filter
bank. With $M = 4$, the steady-state timing error is bounded by,

$$|\epsilon| < \frac{T_s}{M} = \frac{T_s}{4} = \frac{1/2 \times T}{4} = \frac{T}{8}. \quad (2.18)$$

where the sampling interval $T_s$ equals to half of the symbol period $T$. This leads to a new approach in using the Gardner algorithm to tackle burst timing synchronization, which we call the burst Gardner algorithm.

### 2.3 Burst Gardner Algorithm

#### 2.3.1 System Overview

A burst timing acquisition algorithm using Gardner’s formula in Equation (2.1) is presented in the overall block diagram in Fig. 2.10. The scheme is specifically tailored for QPSK modulation, and is tested by simulations for baseband QPSK transmission.

Samples of the two QPSK rails are processed by the two interpolation filter banks, the I-bank and Q-bank. The outputs from the filter banks are then shifted into the two filter output buffers called the I-buffer and the Q-buffer. The timing error detector then uses those buffered outputs to compute the timing error estimates $e^i_k$ for each filter in the filter bank. Based on the filtered timing error estimates $e^i_k$, the timing decision unit chooses the best filter that has the lowest error signal. The outputs of the chosen filters are then selected by the two circular signal collectors as inputs to the phase acquisition algorithm in the receiver.
Figure 2.10: Timing Acquisition Block Diagram.
2.3.2 Timing Error Detector

In the method just described, all filters in the filter bank are active during the timing acquisition process. Their outputs are plotted in Fig. 2.11. The circles represent the A/D samples with zero timing offset $\tau_o = 0$, and the crosses are the interpolation filter outputs, delayed by their associated timing offsets. For convenience, Gardner's
The timing error detector formula is restated here again as

\[ e_k = r_l[k - \frac{1}{2}] * (r_l[k] - r_l[k - 1]) + r_Q[k - \frac{1}{2}] * (r_Q[k] - r_Q[k - 1]). \]  

(2.19)

There are two ways to collect those terms needed to compute \( e_k^l \) for the \( l \)th filter in Equation (2.19). The first and perhaps the most straightforward way is to follow the same method used in the serial version of the Gardner timing detector, in which all terms needed for \( e_k^l \) are from the same \( l \)th filter. For a system using the Nyquist sampling rate, the filter bank is required to produce one set of outputs for every A/D sample received, and to store those outputs in two three-element-buffers, the I-buffer and Q-buffer accordingly. The contents in those two buffers conveniently provide \( R[k], R[k - 1/2] \) and \( R[k - 1] \), where \( R \) can be either \( I \) or \( Q \). Due to the fact that outputs from different filters are not mixed in the calculation of \( e_k^l \), we denote this method as the non-mixed-signal burst Gardner's timing detector, or NMS algorithm, to differentiate it from the second method which will be described later.

However, the relative positions of the interpolation filter bank outputs shown in Fig. 2.11 suggest that it is possible to find a pair of filter outputs \( r_l^m[k] \) and \( r_Q^m[k] \) from the \( m \)th filter, \( m \neq l \) such that they are equivalent to the mid-baud samples \( r_l^l[k - 1/2] \) and \( r_Q^l[k - 1/2] \) of the \( l \)th filter. Therefore it is possible to compute only one set of filter outputs per symbol without violating Equation (2.19). For a system with the Nyquist sampling rate, the filter only needs to produce one output for every two samples received, reducing the computational burden by one half. This leads to our second burst timing algorithm called the mixed-signal Gardner timing detector, or MS algorithm. The MS method consists of two sets of signal assignments for the terms in Equation (2.19) depending on the state of the detector.

Assuming the receiver has been on for a period of time, the previous data strobe is
expected to be close to the optimum value at the maximum eye opening of the received signal if the system is operating normally. Only minor timing offset adjustment is needed. This case is referred to as the fine timing acquisition, and the error signals are generated according to the signal assignments shown in Table 2.1.

<table>
<thead>
<tr>
<th>Filter Index ( l )</th>
<th>Filter Delay (normalized to ( T_s ))</th>
<th>Filter Index for ( r_f(k - \frac{1}{2}) )</th>
<th>Filter Index for ( r_f(k) )</th>
<th>Filter Index for ( r_f(k - 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-0.75</td>
<td>1</td>
<td>-3</td>
<td>previous strobe</td>
</tr>
<tr>
<td>-2</td>
<td>-0.5</td>
<td>2</td>
<td>-2</td>
<td>previous strobe</td>
</tr>
<tr>
<td>-1</td>
<td>-0.25</td>
<td>3</td>
<td>-1</td>
<td>previous strobe</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>previous strobe</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>-3</td>
<td>1</td>
<td>previous strobe</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-2</td>
<td>2</td>
<td>previous strobe</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>-1</td>
<td>3</td>
<td>previous strobe</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>previous strobe</td>
</tr>
</tbody>
</table>

Table 2.1: Signal assignments for \( e_k \) when previous data strobe is reliable.

However when the receiver is initially turned on, it has no knowledge of the previous sampling point. Thus, the best previous data strobe can be from any one of the filters in the bank. Therefore, the first few sets of timing error estimates are taken with \( R_i[k - 1] \) being the output of the \( l \)th filter. These alternative signal assignments are summarized in Table 2.2, and are applied in the coarse timing acquisition.

<table>
<thead>
<tr>
<th>Filter Index ( l )</th>
<th>Filter Delay (normalized to ( T_s ))</th>
<th>Filter Index for ( r_f(k - \frac{1}{2}) )</th>
<th>Filter Index for ( r_f(k) )</th>
<th>Filter Index for ( r_f(k - 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-0.75</td>
<td>1</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>-2</td>
<td>-0.5</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>-1</td>
<td>-0.25</td>
<td>3</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>-3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>-2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>-1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2.2: Signal assignments for the first few sets of \( e_k \)s.
The NMS and MS timing detectors compute their timing offset error signal $e_k$ in significantly different ways. But once $e_k$ is calculated, they follow the same control algorithm to choose the best timing offset $\tau^*$, and to add or drop a sample if necessary.

Before leaving this section, it is important to study the distribution, or S-curve, of the timing error $e_k$ from Equation (2.19). A S-curve of $e_k$ is a curve of expected value of $e_k$ for various timing offset $\delta = \tau - \tau'$. The distribution is shown in Fig. 2.12 when $E_b/N_o = \infty$. Clearly, the curve touches $e_k = 0$ at $\delta = -T_s$, 0 and $T_s$. Therefore, there are at most three corresponding interpolation filters in the bank that can have very small timing error signals. Unfortunately, the Gardner zero crossing detector can only look at the magnitude of $e_k$. It is possible that the timing offset chosen by the detector is $\pm T_s$ away from the correct value. A method has to be found to avoid the potential "false locking".

Figure 2.12: Distribution of the timing error signal $e_k$. 

\begin{center}
\begin{tikzpicture}
\begin{axis}[
    title={Distribution of $e_k(\delta)$},
    xlabel={$\delta = \tau - \tau'$, normalized to $T_s$},
    ylabel={$e_k$},
    xmin=-1, xmax=1,
    ymin=-0.8, ymax=0.8,
    grid=both,
    legend style={at={(0.5,0.95)},anchor=north west}
]
\addplot[domain=-1:1,samples=100,smooth,thick] {0.5*exp(-0.5*(x)^2)};
\end{axis}
\end{tikzpicture}
\end{center}
False Timing Lock Detection

To better search for the true timing offset, we can exploit an interesting property of the expected value of the error signal $e_k^0$ for the filter with zero timing offset as described below.

For a system using a root-raised cosine pulse shaping function as in Equation (2.17), the overall response is a raised-cosine function whose impulse response is defined as

$$h_{RC}(t) = \left(\frac{\sin(Wt)}{Wt}\right) \left(\frac{\cos(\alpha Wt)}{1 - \left(\frac{2\alpha Wt}{\pi}\right)^2}\right).$$  \hspace{1cm} (2.20)

where $W$ is the normalized cutoff frequency of Nyquist pulsing shaping filter.

If there exists a timing offset $\tau$, normalized to the sampling period $T_s$, the zero-crossing signal points $r_l(k - \frac{1}{2})$ and $r_Q(k - \frac{1}{2})$ in Equation (2.19) can be calculated as,

$$r_l(k - \frac{1}{2} + \tau) = \sum_{i=-\infty}^{\infty} \cos(\phi_{k+i}) h_{RC}(-\frac{2i+1}{2} + \tau),$$  \hspace{1cm} (2.21)

$$r_Q(k - \frac{1}{2} + \tau) = \sum_{i=-\infty}^{\infty} \sin(\phi_{k+i}) h_{RC}(-\frac{2i+1}{2} + \tau),$$  \hspace{1cm} (2.22)

where $\phi_k$ is the modulation phase of the $k$th symbol. For illustration, the terms in the summation of Equation (2.21) are shown in Fig. 2.13. Similarly, the data strobes at time $t = kT_s$ are

$$r_l[k] = \sum_{i=-\infty}^{\infty} \cos(\phi_{k+i}) h_{RC}(-i + \tau)$$  \hspace{1cm} (2.23)

$$r_Q[k] = \sum_{i=-\infty}^{\infty} \sin(\phi_{k+i}) h_{RC}(-i + \tau).$$  \hspace{1cm} (2.24)

The terms in Equation (2.23) are illustrated in Fig. 2.14. Furthermore, $r_l[k - 1]$ and $r_Q[k - 1]$ can written as

$$r_l[k - 1] = \sum_{i=-\infty}^{\infty} \cos(\phi_{k+i-1}) h_{RC}(-i + \tau)$$  \hspace{1cm} (2.25)
The timing preamble is an alternating sequence whose modulation phase $\phi$ switches between the first and the third quadrants. Therefore, the difference between any two modulation phases in the timing preamble as given by,

$$\phi_{k+i} - \phi_{k+j} = (i - j)\pi.$$  \hspace{1cm} (2.27)

Substituting equations from (2.21) to (2.26) into Equation (2.19) and applying Equation (2.27), after some manipulations we have:

$$e_k = 2 \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (-1)^{i-j} h_{RC}(-\frac{2i + 1}{2} + \tau) h_{RC}(-j + \tau).$$ \hspace{1cm} (2.28)

From Figs. 2.13 and 2.14, we can see that $h_{RC}(-\frac{2i + 1}{2} + \tau)$ becomes negligible for $i$ except $i \neq -1, 0, 1$. $h_{RC}(-j + \tau)$ is also found to be very close to zero for
i \neq -1, 0 and 1. Therefore, Equation (2.28) can be approximated by

\[ e_k \approx 2 \sum_{i=-1}^{0} \sum_{j=-1}^{1} (-1)^{i-j} h_{RC}(-\frac{2i+1}{2} + \tau) h_{RC}(-j + \tau) \]
\[ = 2 [h_{RC}(\tau - \frac{1}{2}) - h_{RC}(\frac{1}{2} + \tau)] [h_{RC}(\tau) - h_{RC}(1 + \tau) - h_{RC}(\tau - 1)] \] \( 2.29 \)

The terms \( a \) and \( b \) of Equation (2.29) are plotted on Fig. 2.15. Then the signs of \( a \) and \( b \) can be determined as follows. If \( \tau > 0 \), then

\[
\begin{align*}
  a < 0 \\
  b > 0 \\
\end{align*}
\]
\[
\Rightarrow e_k < 0. \quad (2.30)
\]

Similarly, if \( \tau < 0 \), then
Therefore, the error signal $e_k^0$ of the 0th filter with $\tau = 0$ would be positive if there exists a negative timing offset. On the other hand, it would be negative if the timing offset is positive. With a low-pass loop filter to suppress noise, $e_k^0$ can reliably detect the sign of the timing offset between the transmitter and receiver after a few symbols. As a result, the range of search for the true timing offset $\tau$ can be reduced.
Energy Accumulator

From Fig. 2.12, there are at least two filters in the filter bank that can have the minimum error signal $e_k$. The control logic can only act on the magnitude of $e_k$. Due to this ambiguity, a false timing lock can occur if the mid-baud samples $r_{I}[k-1/2]$ and $r_{Q}[k-1/2]$ are at or near the maximum eye opening of the received signal, or its peak values. Consequently, the data strobe points $r_{I}[k]$, $r_{I}[k-1]$, $r_{Q}[k]$ and $r_{Q}[k-1]$ are at or close to the zero crossing points. In the absence of noise, we can describe those signals as:

$$
\begin{align*}
    r_{I}[k - \frac{1}{2}] &= r_{I(peak)} \\
    r_{I}[k] &= 0 \\
    r_{I}[k-1] &= 0
\end{align*}
$$

and

$$
\begin{align*}
    r_{Q}[k - \frac{1}{2}] &= r_{Q(peak)} \\
    r_{Q}[k] &= 0 \\
    r_{Q}[k-1] &= 0.
\end{align*}
$$

Substituting Equations (2.32) and (2.33) into Equation (2.19), we have $e_k = 0$. Therefore, the detector cannot correct a time offset of $T_s$ away from the true value $\tau$. In fact, this is a common problem in most timing synchronization schemes. The standard method to solve this is called “double-locking”. In this method, the timing estimate is intentionally moved away from its estimated value after the timing lock is first achieved. If the timing estimate falls back to the previous value, that timing offset can be confirmed to be close to the true timing offset. Otherwise, the initial
timing estimate is at a false stable point, and the true timing offset must be \( \pm T_s \) away. One drawback of this method is that the time required to finalize a timing lock is effectively doubled. This is not a serious problem in serial transmission where the amount of time used for synchronization is relatively small compared to the message payload. But timing acquisition for burst applications must be very short, hence the double-locking scheme may not be an appropriate solution.

A different approach is to check the signal energy of the chosen interpolation filter at the end of timing acquisition. If the final timing estimate is \( \pm T_s \) away from the true value of \( \tau \), the decimated samples from the particular filter should be close to the zero crossing points, whose accumulated energy is expected to be small compared to the energy associated with the best filter. Therefore, an energy check at the end of the timing acquisition will be able to resolve the ambiguity in the timing offset estimate. Fig. 2.16 shows the structure of the energy accumulator for the interpolation filter pair with timing offset \( \tau_l = l/MT_s \). The filter outputs are first decimated, then the energy is added to the total energy \( E_i \) for the \( l \)th filter.

![Figure 2.16: Structure of the energy accumulator for the filter pair \( h_{rrc}(nT_s + \tau_l) \).](image)

Fig. 2.16: Structure of the energy accumulator for the filter pair \( h_{rrc}(nT_s + \tau_l) \).
2.3.3 Control of Timing Acquisition

With the methods to detect the presence of a timing offset \( \tau \), we can now move on to discuss how to use the timing error \( e_k \) to correct a timing offset. The method can be divided into two different sections, namely coarse timing acquisition and fine timing acquisition. As mentioned above, the control steps are applied to both the mixed-signal and the non-mixed-signal Gardner timing offset detector.

Coarse Timing Acquisition

Sample Points Shown on the Input Curve

![Diagram showing sample points on the input curve.]

Figure 2.17: The second set of outputs from the interpolation filter bank.

The output of a \( N \)-tap filter becomes valid after \( N \) samples have been shifted into the filter buffer. With the first two sets of valid filter outputs shown in Fig. 2.17, the
timing control enters the *coarse timing acquisition* mode. The timing error signal $e_k^t$ is computed according to the proper signal assignments based on filter outputs shown in Fig. 2.17.

![Coarse timing acquisition flow control](image)

Figure 2.18: The flow diagram for the coarse timing acquisition.

Just as in the basic Gardner's timing recovery scheme in Fig. 2.7, a low-pass loop filter is vital to the proper operation of the timing detector. Belanger [11] found that a simple first-order low-pass IIR filter is adequate for the purpose of noise suppression of the timing error signals. Therefore after a few symbols, we can reliably determine the sign of the timing offset from the sign of $e_k^0$, which is the low-pass filtered output of the interpolation filter with zero timing offset ($l = 0$). A preliminary timing offset estimate is obtained using the flow diagram shown in Fig. 2.18.
However, this early estimate can fluctuate dramatically from one symbol period to another. To further increase the accuracy of the estimate, a majority rule is applied to select the final timing offset estimate at the end of each symbol interval. To do so, all the preliminary timing offset estimates are stored in memory, and the final timing offset estimate is the one that occurs most frequently. The number of these preliminary timing estimates stored in memory should be made equal to the length of the burst timing tracking period. By doing so, the chance of getting a correct timing offset at the end of the coarse tracking is maximized. Simulation experiments show that the coarse timing acquisition needs to run for only four symbols to have an acceptable rough estimate of the timing offset $\tau$. Longer burst tracking yields a diminishing return in performance. Therefore, the length of this coarse timing acquisition is fixed to four symbols for the rest of the chapter.

**Fine Timing Acquisition**

After the coarse timing acquisition, a rough estimate of the timing offset should be available. To further increase the accuracy of that estimate, fine timing acquisition is carried out after the completion of coarse timing acquisition.

Fine timing tracking differs from coarse timing tracking in the scope of the search. When the receiver is just turned on, the true timing offset $\tau$ can be anywhere within the range $(-T_s, T_s)$. Therefore, the coarse timing acquisition procedure has to search for the best timing offset estimate from all the filters in the interpolation bank. However, in the fine timing acquisition, a timing offset estimate $\tau'$ obtained at the end of the coarse timing acquisition process is assumed to be close to the true $\tau$ or $\tau \pm T_s$. Consequently, search scope of the fine timing acquisition procedure can be narrowed.
down to the nearby filters of the current timing offset estimate for better accuracy. The flow diagram for this fine timing tracking procedure is shown in Fig. 2.19. In the diagram, \( l^* \) denotes the index of the filter in the filter bank whose timing offset is the current estimate. We only look at the value of \( \hat{e}_k^{l^*} \) which is the filtered timing error signal with the current timing offset estimate. If \( \hat{e}_k^{l^*} \) is below the threshold, no correction is needed, and the timing offset estimate is still \( l^* \). Otherwise, the new timing estimate is either the increment or the decrement by one from the current timing offset estimate \( l^* \) depending on the sign of \( \hat{e}_k^{l^*} \).

**Fine timing acquisition flow control**

![Flow diagram for fine timing acquisition](image)

Figure 2.19: The flow diagram for the fine timing acquisition.
The value of the threshold $Th$ used in Fig. 2.19 is determined empirically relative to an input energy of unity. The symbol error probabilities at $E_b/N_0 = 7$ dB were obtained through simulation for different values of $Th$ in the range of $[0.2, 2]$ with a step increase of 0.05. For the NMS timing detector, it was found that the best performance occurred when $Th = 0.7$. For the MS detector, $Th = 0.75$ gave the lowest symbol error probability.

Another new feature in this fine timing acquisition mode is the adding or dropping of a sample by controlling the decimation interval. As explained earlier, the timing offset estimate can be driven to three equally likely stable points $\tau$ or $\tau \pm T_s$. It is necessary for the receiver to be able to identify and correct the cases when the timing offset estimate is at $\tau \pm T_s$. In the simulations of this thesis, the receiver is assumed to be activated within $\pm T_s$ after the message is put onto the channel. The cases that the receiver is activated with a delay of greater than $\pm T_s$ are not considered in this thesis.

As the final step in this fine timing tracking acquisition, an energy check is performed to ensure the timing estimate is not $\pm T_s$ away from the true value. In the case of a false timing estimate, the energy of the decimated filter outputs of that filter should be much less than the optimal filter. The flow chart of this final step is shown in Fig. 2.20. From the current timing estimate $l^*$, we can locate the index of the filter that is $T_s$ away from filter $l^*$. Denote that index as $l''$, then $l'' = l^* + 4$ if $l^* \leq 0$. Otherwise, $l'' = l^* - 4$ if $l^* > 0$. Therefore, if the energy of the filter $l''$ is larger than that of filter $l'$, then we declare the $l''$ filter as the optimal filter. Otherwise, we have to search for the filter with the highest energy on the side of the filter bank where filter $l''$ resides.
To determine the length of this fine timing acquisition, we should look at the statistical behaviour of the residual timing offset, \( \delta = \tau - \tau' \). In simulations, a message of 550 symbols is sent to the receiver with a random timing offset that is fixed for the duration of that message. The mean and variance plots of the residual timing offset are shown as a function of \( E_b/N_0 \) in Fig. 2.21, where each data point is gathered by transmitting 300,000 messages. Those results clearly show that the longer the timing tracking interval, the better the timing offset estimate becomes. If the timing tracking interval is too short, a large residual timing offset can be left at the end of the burst tracking interval. On the other hand, the residual timing offset
is minimized by taking a longer preamble. However, both speed and accuracy are considered to be equally important in timing acquisition for a burst communication application. The balance of the two can be achieved by choosing the mid-point of these two extreme cases. In this case, \( L = 8 \) is found to lie in the middle among the curves in Fig. 2.21. Therefore, a timing preamble of eight symbols is found to be acceptable, and is used for the rest of the thesis.

**Data Mode Tracking**

When the message payload is long or the timing offset \( \tau \) is changing rapidly, it becomes necessary to keep tracking the timing offset in the data mode. Gardner's timing algorithm works very well for the preamble with an alternating bit pattern. However, the random data in the message payload cannot provide the regular timing transition needed for the Gardner timing error detector. As a result, the timing error detector described above has to be modified before being applied in the data mode.

To overcome this random data transition, if we observe a long sequence of data before making an estimate, then the effects of good and bad sequences tend to cancel out, and the timing offset estimation becomes independent of the data pattern. A simple yet effective solution to this problem is to raise the threshold used in the timing control, which will result a longer time for the error signal to exceed the threshold before a timing correction is taken.

Another modification of the burst timing procedure to increase the tracking performance in data mode is to increase the number of interpolation filters used. This modification effectively increases the resolution of the timing algorithm. When the error signal exceeds the error detector threshold, a smaller correction step is taken,
Figure 2.21: Mean (a) and Variance (b) of the residual timing offset. The variable $L$ is the number of preamble symbols.
reducing the residual time offset after timing corrections.

2.4 Performance Evaluation

In communication systems, the most important and unambiguous performance index is the bit error rate. The two timing offset detector schemes described in the foregoing are then evaluated by modeling transmission of ATM cells and measuring the resulting bit error rate.

Due to the nature of an ATM cell, each cell is transmitted independently. For simplicity, transmission of each cell does not overlap in time, and a new cell is sent after a random time delay relative to the previous cell. Hence, the timing offset associated with each ATM cell is different. Also, due to the fact that an ATM cell consists of only 53 bytes or 424 bits, the timing offset remains approximately unchanged over the duration of the cell. Timing synchronization has to be re-established by the receiver for every ATM cell received. To do so, a timing preamble is attached to the start of each ATM cell. For QPSK in particular, it is a dotting sequence of \(10101010\) for both I and Q-rails. Simulation results of the bit error rate (BER) for both timing schemes are chosen to have a 99% confidence interval with \(\pm 10\%\) of error using the Chernoff bound[37]. Also, to ensure the timing detectors are thoroughly tested, at least 100,000 ATM cells were transmitted. Finally, the received messages are not stored or reprocessed for the simulations in this chapter.

Except for the forward error correction codes, storing and reprocessing of a message are not used in this chapter.

With a constant timing offset for each ATM cell, the BER plots of both timing
Figure 2.22: BER plots of the NMS (a) and MS (b) timing algorithms for fixed timing offset.
schemes are shown in Fig 2.22. The equation of the theoretical symbol error probability for coherent QPSK can be found in [3, p. 272]. Both timing detection schemes suffer a loss of only 0.15 dB in SNR and can follow the theoretical limit very closely at $E_b/N_o$ as low as -3 dB. Therefore, both the non-mixed-signal (NMS) and mixed-signal (MS) timing detectors work very well in tracking a constant timing offset of a short message.

It is also interesting to investigate how the two timing algorithms work under a constant timing drift. A timing drift can be modeled by switching transmit filter regularly. A bank of interpolation filters each with different timing offset is used by the transmitter as the pulse-shaping filters. The timing offset of the $l$th filter in the bank is $\frac{l}{32} T_s$, where $l = -31, -30, \cdots, 0, 1, \cdots, 32$. When transmitting a message, only one of the interpolation filters is used by the transmitter as the default pulse-shaping filter at any time, and the transmitter changes the pulse-shaping filter regularly to create a slowly varying timing offset. In our case, the sender changes the pulse-shaping filter every thirty symbols, and drops a sample for every 960 symbols sent. To cope with this continuously varying timing offset, the receiver performs timing tracking in the data mode as well. The resulting BER results for the two algorithms are shown in Fig. 2.23. Both schemes show significant improvement in BER for the high SNR range starting at about 3.5 dB in $E_b/N_o$. The loss in $E_b/N_o$ is lowered to about 0.08 dB from the previous 0.15 dB. This confirms the hypothesis that it is helpful to have some sort of timing tracking for the message payload. It not only keeps up with the varying timing offset, but also reduces the resultant residual timing offset by correcting the wrong estimate at the end of the preamble tracking. However, at lower $E_b/N_o$, the mixed-signal timing detector begins to show sign of failing, and eventually
Figure 2.23: Mean of the residual timing offset estimate for a time-varying timing offset at a rate of $0.001T_s$/symbol.
loses almost 3dB in SNR. On the other hand, the non-mixed-signal timing detector continues to work well with a constant loss of merely 0.08dB in SNR. Therefore, the NMS timing offset detection scheme is superior if $E_b/N_o$ is lower than 3dB. However, this better performance comes with a price in computational requirement. The NMS scheme requires two filter outputs for every symbol, while the MS scheme needs only one filter output per symbol. Therefore, using mixed signals from the filter bank to compute $e'_k$ reduces the computational requirements by one half. However, most practical applications guarantee the signal strength to be higher than 3.5 dB, and both methods work extremely well for that SNR range. Therefore, they should be regarded as acceptable in practice.
2.5 Summary

This chapter began by stating the basics of Gardner's timing error detector formula[9] and a general structure for timing offset detection. Coefficient interpolation and sample adding/dropping were then explained as methods for timing correction. Two new timing algorithms based on Gardner's formula were developed specifically for burst transmission. Both schemes required a timing preamble of only eight symbols and the resultant loss in SNR was shown to be very small.
Chapter 3

Phase Acquisition

3.1 Background

In the demodulation of a QPSK signal, it is a common practice to translate the received intermediate frequency signal down to baseband and then to operate on the resulting low-frequency waveform. Unfortunately, this process introduces a carrier phase error between the transmitted signal and the local oscillator, seriously degrading the coherent demodulation of the QPSK signal. Consider the demodulation of a pulsed-shaped QPSK signal as shown in Fig. 3.1, the mathematical model for the received passband signal $r_{IF}(t)$ is:

$$r_{IF}(t) = \Re\{S_{CE}(t)e^{j\omega c t}\},$$  \hspace{1cm} (3.1)

where $\omega_c$ represents the carrier frequency. $S_{CE}(t)$ is the transmitted message or the complex envelope relative to $\omega_c$ whose expression is:

$$S_{CE}(t) = \left\{\sum_i e^{j\theta_i}\delta(t - iT)\right\} * h_{rrc}(t)$$

$$= \sum_i e^{j\theta_i}h_{rrc}(t - iT),$$  \hspace{1cm} (3.2)
where \( \theta_i \) is the \( i \)th information symbol \( \theta_i \in \{ \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \} \) for QPSK. \( h_{rrc}(t) \) is the pulse-shaping root-raised-cosine function used in the transmitter to combat intersymbol interference (ISI).

As indicated in Fig. 3.1, the demodulation is accomplished by multiplying \( r_{IF}(t) \) with two local carrier references \( 2 \cos(\omega_{cl}t + \phi_l) \) and \(-2 \sin(\omega_{cl}t + \phi_l)\). The results are then filtered by root-raised-cosine filters to eliminate the high frequency terms around \( \omega_c + \omega_{cl} \), where \( \omega_c \) and \( \omega_{cl} \) are frequencies of the oscillator in the transmitter and receiver, respectively. Mathematically, the filter output \( r_i(t) \) is obtained as:

\[
\begin{align*}
    r_i(t) &= \{r_{IF}(t)2 \cos(\omega_{cl}t + \phi_l)\} * h_{rrc}(t) \\
    &= \text{Re} \{S_{CE}(t)e^{j\omega_c}2 \cos(\omega_{cl}t + \phi_l)\} * h_{rrc}(t) \\
    &= \text{Re} \{S_{CE}(t)e^{j\omega_c}2 \cos(\omega_{cl}t + \phi_l)\} * h_{rrc}(t) .
\end{align*}
\]

But since:

\[
\cos(\omega_{cl}t + \phi_l) = \frac{1}{2}(e^{j(\omega_{cl}t+\phi_l)} + e^{-j(\omega_{cl}t+\phi_l)}),
\]

\[61\]
Equation (3.3) becomes:

\[
\begin{align*}
    r_I(t) &= \text{Re} \left\{ S_{CE}(t) e^{j \omega_c (t + \phi_t) + e^{j (\omega_c \phi_t)}} \right\} * h_{rrc}(t) \\
    &= \text{Re} \left\{ S_{CE}(t) e^{j (\omega_c + \omega_{cl}) t + \phi_t} + S_{CE}(t) e^{j (\omega_c \phi_t)} \right\} * h_{rrc}(t) .
\end{align*}
\]  

(3.5)

The pulse-shaping root-raised-cosine filter is a low-pass filter whose cutoff frequency is much smaller than the carrier frequency \( \omega_c \). Therefore, the term with frequency \( \omega_c + \omega_{cl} \) is effectively removed by the filter. Furthermore, in a well designed system, \( \omega_c \) and \( \omega_{cl} \) should be quite close, and their difference is usually kept small by frequency recovery. Denoting that frequency difference as \( \Delta \omega = \omega_c - \omega_{cl} \), \( r_I(t) \) is reduced to

\[
    r_I(t) = \text{Re} \left\{ S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} * h_{rrc}(t) .
\]

(3.6)

Similarly, the filtered output \( r_Q(t) \) for the Q-rail can be derived as:

\[
    r_Q(t) = \text{Re} \left\{ -j S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} * h_{rrc}(t) .
\]

(3.7)

It is more convenient to combine \( r_I(t) \) and \( r_Q(t) \) to form a single complex waveform \( r(t) \) as:

\[
    r(t) = r_I(t) + j r_Q(t) = \text{Re} \left\{ S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} * h_{rrc}(t) + j \text{Re} \left\{ -j S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} * h_{rrc}(t)
\]

\[
    = \left\{ \text{Re} \left\{ S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} + j \text{Re} \left\{ -j S_{CE}(t) e^{j (\Delta \omega t - \phi_t)} \right\} \right\} * h_{rrc}(t)
\]

\[
    = e^{j (\Delta \omega t - \phi_t)} S_{CE}(t) * h_{rrc}(t) .
\]

(3.8)

Substituting Equation (3.2) into (3.8), we get:

\[
    r(t) = e^{j (\Delta \omega t - \phi_t)} \left\{ \sum \delta(t - iT) \right\} * h_{rrc}(t)
\]

\[
    = e^{j (\Delta \omega t - \phi_t)} \left\{ (\sum \delta(t - iT)) * h_{rrc}(t) \right\} * h_{rrc}(t)
\]

\[
    = e^{j (\Delta \omega t - \phi_t)} \left\{ \sum \delta(t - iT) \right\} * \{ h_{rrc}(t) * h_{rrc}(t) \} .
\]

(3.9)

62
However, as

\[ h_{rc}(t) = h_{rrc}(t) * h_{rrc}(t). \]

(3.10)

\( r(t) \) becomes

\[
\begin{align*}
r(t) &= e^{j(\Delta \omega t - \phi t)} \left\{ \sum_i e^{j\theta_i} \delta(t - iT) \right\} * h_{rc}(t) \\
&= e^{j(\Delta \omega t - \phi t)} \sum_i e^{j\theta_i} h_{rc}(t - iT).
\end{align*}
\]

(3.11)

Since the raised-cosine function satisfies the Nyquist criterion for zero intersymbol interference (ISI):

\[
h_{rc}(iT) = \begin{cases} 
1 & : \text{for } i = 0 \\
0 & : \text{for } i \neq 0.
\end{cases}
\]

(3.12)

At the sampling times \( t = iT \), we have

\[
r(i) \equiv r(iT) = e^{j(\Delta \omega t - \phi t)} e^{j\theta_i} = e^{j(\Delta \omega t - \phi t + \theta_i)}. \]

(3.13)

Clearly, even with perfecting timing synchronization, the information symbol \( \theta_i \) can still be corrupted by the phase error \( \Delta \omega t - \phi t \) between the sender and receiver as
shown in Fig. 3.2. The signal constellation is rotated by $\Delta \omega t - \phi_i$ radians away from its correct position, seriously reducing the noise margin for detection. In fact, if the constellation is rotated by $90^\circ$, then the transmitted information is completely lost. Therefore, the effective estimation and removal of this phase error has fundamental importance in coherent QPSK demodulation.

### 3.2 Phase Acquisition

$$u_i = \cos(\omega_i t + \phi_i(t))$$

![Phase Acquisition Diagram](image)

Figure 3.3: Block diagram of an analog phase-locked loop.

Conventional methods to recover carrier phase in a digitally modulated signal are based on phase-locked loops (PLL)\cite{17, 38} similar to the one shown in Fig. 3.3. To see how this analog PLL works, assume that $\Delta \omega = 0$ such that the phase offset does not vary over time. When the phase-locked loop is first turned on, the output of the VCO is zero. A nonzero phase offset will cause the phase comparator to develop a nonzero signal proportional to the phase difference $\phi_i$ and the VCO will be gradually driven to match the frequency and phase of the input signal. This PLL method has been thoroughly developed in literature and works very well to track a slowly varying phase offset. However, PLL suffer from a long tracking time just as in most feedback control...
system. The time required by the VCO to produce a reasonably accurate estimate is quite long and is usually around one hundred symbol periods[6]. Therefore, a PLL is not very suitable in fast phase acquisition for burst transmission. This leads us to search for an alternative.

The phase acquisition algorithm used in this thesis is derived from the open loop technique given by Gracie[5]. Two reasons lead to this selection. First, we need a method that is fast in tracking and efficient in computation. The algorithm used by Gracie is called Open Window Phasor Estimation. Its feedforward mechanism ensures fast acquisition of a phase offset. In addition, the algorithm operates entirely on complex baseband signals, in particular the I and Q-rails for QPSK. This eliminates the need for conversions between polar and rectangular coordinates normally required by PLL algorithms. Thus, it avoids those expensive fixed-point DSP operations needed to implement table look-up and division. Moreover, this phasor estimation scheme does not require normalization on the samples except for the proper treatments of underflow and overflow. Consequently, the complexity of a receiver is dramatically reduced because division is considered as one of the most expensive operations in fixed-point DSP implementations.

The second motivation for selecting Gracie's algorithm is the tracking capacity of the method. Although a frequency tracking circuit is normally used to track the carrier frequency of the received signal, there is almost always some frequency residue left at the end of the frequency recovery. A common source of this frequency difference is due to Doppler frequency shift. For instance, in a LEO satellite system with transmission symbol rate of 45M baud, the maximum Doppler frequency shift
can be as much as 40kHz\[39\], which translates to a normalized frequency of

\[
\overline{D_f} = \frac{D_f}{R} = \frac{40 \times 10^3}{45 \times 10^6} = 8.9 \times 10^{-4}. \tag{3.14}
\]

Gracie's open window phasor estimation has been shown to be able to track a normalized frequency drift of up to 0.001 cycles per symbol. Therefore, the algorithm can handle the Doppler effect in a LEO satellite system.

### 3.3 Open and Closed Loop Phasor Estimation

Our phase acquisition algorithms are based on the open loop phasor estimation techniques described by Gracie in \[5\]. The results are new phase acquisition schemes that can work in a much lower SNR range by adding two low-pass filters to suppress large fluctuations of the phase estimates.

The input to the phase estimation unit can be written on a per symbol basis as

\[
r[k] \equiv r_k = r_r[k] + j r_Q[k] = A_k e^{j \theta_k} e^{j (\Delta \omega t - \phi_k)} = A_k e^{j \theta_k} e^{j \phi_k}, \tag{3.15}
\]

where \( \theta_k \) is the \( k \)th symbol information, and \( \phi_k \) is the time-varying phase offset for symbol \( k \). The magnitude of the vector \( A_k \) is included in the expression to show that it does not affect the operation of the phasor estimation as long as it does not get too large or too small. In addition, timing synchronization is assumed to be perfect throughout this chapter and the decimated filter samples are taken at the maximum
eye opening of the received signal with zero ISI.

Two algorithms from [5] are examined in this thesis, namely the open loop phase acquisition with *early* data window and the open loop phase acquisition with *late* data window. Both algorithms operate on blocks of complex input vectors called *data windows*. The input vector streams are grouped into distinct blocks with non-overlapping time intervals. Each window of data is then used to produce an estimate of the phasor error. In other word, the phasor error estimate is updated on a per window basis, and is illustrated in Fig. 3.4.

![Figure 3.4: Partition of data into distinct data windows.](image)

### 3.3.1 Early Window Data Decision

The first technique to be described is the open loop phase acquisition with an *early* data window. As shown in Fig. 3.5, it takes in the complex decimated matched filter output $r_k$ at $t = kT_s$. The first step in the phase estimation process is to remove the phase error from the received vector as
Figure 3.5: Structure of the early window phase acquisition.
where $r_k$ is one of the data points in window $n$, and $e_{n-1} = B_{n-1}e^{-j\hat{\phi}_{n-1}}$ is the phase error estimate from the previous window $n-1$. The phase corrected vector $r'_k$ is then used to determine the information symbol $e^{j\theta_k}$. If the data decision is correct, the phase error between the phase error $e_{n-1}$ can be obtained by removing the information phase $\theta_k$ from the received input $r_k$. Mathematically, the operation is

$$
\begin{align*}
    r''_k &= r_k \cdot \left\{ e^{j\hat{\theta}_k} \right\}^* \\
         &= r_k \cdot e^{-j\hat{\theta}_k} \\
         &= A_k e^{j(\theta_k + \phi_k)} e^{-j\hat{\theta}_k} \\
         &= A_k e^{j\phi_k} e^{j(\theta_k - \hat{\theta}_k)}. \\
\end{align*}
$$

(3.16)

If $\theta_k = \hat{\theta}_k$, then we have

$$
    r''_k = A_k e^{j\phi_k},
$$

(3.17)

and the phase error estimate for the $k$th symbol is then

$$
    \hat{\phi}_k = \phi_k.
$$

(3.18)

This method works well if the data decision $\hat{\theta}_k$ is reliable, but the difficulty in making a correct data decision is that there are several combinations of $\hat{\theta}_k$ and $\hat{\phi}_k$ that can be mapped to the same observation $r_k$. This is easily seen by rewriting
Equation (3.15) as

\[ r_k = A_k e^{j\theta_k} e^{j\phi_k} \]
\[ = A_k e^{j\theta_k} e^{j(\pi/2-m\pi/2)} \]
\[ = A_k e^{j(\theta_k+m\pi/2)} e^{j(\phi_k-m\pi/2)} \]
\[ = A_k e^{j\theta_k} e^{j\phi'}, \]  \hspace{1cm} (3.19)

where \( \theta'_k = \theta_k + \frac{m\pi}{2}, \ m = 0, \ldots, 3 \) are all eligible symbol alphabets for QPSK, and there is no way to distinguish among those possible \( \theta'_k \) by the phase detector. Therefore, the data symbol estimate is only correct if the phase error estimate from the previous window is within \( \pm 45^\circ \) from the true \( \phi_k \) for QPSK. That is,

\[ |\varepsilon_k| = |\phi_k - \hat{\phi}_k| < \frac{1}{2} \cdot \frac{2\pi}{2} \]
\[ = \frac{\pi}{4}, \]  \hspace{1cm} (3.20)

where \( \varepsilon_k \) is the residue phase error.

Under normal channel conditions, the data decision is quite reliable. As a result, \( \varepsilon_k \) should remain within \( \pm \frac{\pi}{4} \) radians most of the time. However, this condition is violated every time the receiver makes an incorrect data decision and \( \varepsilon_k \) is dramatically changed relative to its previous value \( \varepsilon_{k-1} \). A low-pass loop filter can be used to suppress the erroneous jump in \( \varepsilon_k \), thus minimizing its adverse impact on the overall phase error estimate \( e_{n-1} \). Simulations showed that the simple first-order infinite-impulse response (IIR) filter used in Chapter 2 is adequate for this task. For completeness, its structure is included here again in Fig. 3.6.

Using a similar analysis as in Chapter 2, the two parameters \( A \) and \( B \) of the IIR
Figure 3.6: First-order IIR Loop Filter.

Filter can be determined according to the two following equations:

\[
\begin{align*}
A^2 + 2A(\cos(2\pi f_{3dB}) - 2) + 1 &= 0 \\
A + B &= 0.
\end{align*}
\]

(3.21)

The term \( f_{3dB} = 0.001 \) reflects the fact that the phase acquisition is intended to track 0.001 frequency drift. Solving Equation (3.21), \( A \) should be chosen as 0.99 to give a normalized cutoff frequency of 0.001. During implementation, however, it is found by trial and error that using \( A = 0.93 \) and \( B = 0.07 \) results in a better performance. This is because the tracking time required by the IIR filter to approach its asymmetric limit is proportional to the value of \( A \). If data decisions are mostly correct, then the true phase error \( \phi_k \) should be able to pass the filter almost undisturbed. Hence the output of the IIR low pass filter can be approximated as

\[
r''_k = A_k e^{i\phi_k}.
\]

(3.22)

Finally, this per symbol phase error estimate is accumulated to form the phase error estimate for window \( n \) according to

\[
\text{Re}\{B_n e^{i\phi_n}\} = \frac{1}{N} \sum_{k=(n-1)N}^{(n-1)N+(N-1)} \text{Re}\{r''_k\}
\]

71
where $N$ is the window size. Equation (3.23) can be combined to form a complex vector expression as

$$B_n e^{j\hat{\phi}_n} = \frac{1}{N} \sum_{k=(n-1)N}^{(n-1)N+(N-1)} \text{Im}\{ r_k'' \}$$

(3.24)

This phase error estimate is then used on the next window of data. It should be pointed out that the final symbol decision is based on $r_k'$. 

### 3.3.2 Late Window Data Decision

The second phase tracking algorithm investigated in this thesis is called *open loop phase acquisition with late window data decision*, and its block diagram is shown in Fig. 3.7. Similar to the early window method, it removes the phase error from the received input vector $r_k$ according to

$$r_k' = r_k e^{-j\hat{\phi}_{n-1}}$$

$$= r_k B_{n-1} e^{-j\hat{\phi}_{n-1}}$$

$$= A_k B_{n-1} e^{j\phi_k} e^{-j\hat{\phi}_{n-1}}.$$ 

(3.25)

The result is then used to form a preliminary data estimate $\hat{\phi}_k$. If that symbol decision is correct, the phase error at $t = kT_s$ can be estimated according to

$$r_k'' = r_k \cdot \left\{ e^{j\hat{\phi}_k} \right\}^*$$

$$= r_k e^{-j\hat{\phi}_k}$$

$$= A_k e^{j(\theta_k + \phi_k)} e^{-j\hat{\phi}_k}$$

$$= A_k e^{j\phi_k} e^{j(\theta_k - \hat{\theta}_k)}.$$ 

(3.26)
Figure 3.7: Structure of the late window phase acquisition.
If $\theta_k = \hat{\theta}_k$, then

$$r'_k = A_k e^{j\phi_k}$$

(3.27)

and the phase error estimate for this symbol becomes,

$$\hat{\phi}_k = \angle r''_k = \phi_k.$$  

(3.28)

Similar to the early window method, a first-order low-pass IIR filter is used to limit the effect of incorrect data decisions on the quality of phase error estimate. The filtered phase error can be approximated closely by

$$r'''_k = A_k e^{j\phi_k}$$

(3.29)

which accumulates over the length of a window ($N$ symbols) to form a new phase error estimate for window $n$ as

$$B_n e^{j\phi_n} = \frac{1}{N} \sum_{k=(n-1)N}^{(n-1)N+(N-1)} r'''_k.$$  

(3.30)

When compared to the early window method, this late data window phase acquisition scheme uses the same algorithm to obtain an estimate of phase error. It differs from its counterpart however on the use of those phase estimates to aid the symbol detection. In the early window method, the final data decision for the $k$th symbol in window $n$ is based on $r'_k$, which is the input vector $r_k$ rotated by the phase error estimate from the previous $(n - 1)$th window. That phase error estimate used to perform phase correction can be as many as $N$ symbols "older" than the input vector $r_k$.

Although this is not a problem if the phase offset is relatively constant, large phase error residue can be introduced when the phase offset is changing very fast. The problem is best illustrated in Fig. 3.8 where a positive frequency drift causes the phase offset to increase gradually. The phase estimates for two consecutive windows
are plotted as the mid-points within the corresponding window. It is clear that the phase error of the last symbol in window $n$ is significantly larger than the phase estimate $\hat{\phi}_{n-1}$ from the $(n - 1)$th window.

Mathematically, the phase difference can be analyzed if we let $\Delta \phi$ be the phase difference between the estimate $\hat{\phi}_{n-1}$ and the actual phase error $\phi_{nNT_s}$ at the start of the $n$th window, and $\beta$ be the symbol by symbol increase in phase error. For the early window method where $\hat{\phi}_{n-1}$ is used in window $n$, the amount of residual phase error left on symbol $k$ in the $n$th window is

$$\gamma = \Delta \phi + (k - (n - 1)N)\beta, \quad (3.31)$$

and the maximum residual phase error occurs at the last symbol of window $n$, and has the value of

$$\gamma_{\text{MAX}} = \Delta \phi + (nN - (n - 1)N)\beta = \Delta \phi + N\beta. \quad (3.32)$$
Therefore, to improve tracking performance against a moderate residual frequency offset, the late window phase acquisition method tries to use the most accurate phase error estimate on each received vector by storing those input vectors in a delay buffer of $N$ elements. By doing so, a data decision is deferred by one window (or $N$ symbols) in time and the phase error estimate from the $n$th window is applied to the input vectors in the same $n$th data window. The final symbol decision is then based on the signal

$$
d_{W,k-N} = r_{k-N} \cdot B_{n-1} e^{-j\hat{\phi}_{n-1}}
$$

$$
= A_{k-N} B_{n-1} e^{j\theta_{k-N}} e^{j(\phi(k-N) - \hat{\phi}(n-1))}
$$

(3.33)

where the subscript $k - N$ emphasizes the time relation between the received input $r_k$ and the actual output for the data decision $d_{W,k-N}$. It is interesting to note that the phase error estimate $e^{\hat{\phi}_n}$ serves both as the early phase error estimate for window $n$ when assisting the phase error extraction, and as the late phase error estimate when performing the final data detection for the $(n - 1)$th window. Finally for the late data window phase error acquisition algorithm, the residual phase error is approximately bounded by $\pm \Delta \phi$ which is clearly much better than the possible residual phase error in the early window method.

It is also worth noting the underlying mechanism employed in both methods to estimate a phase offset. The phase error estimation within a window follows a feed-forward algorithm. Those phase error estimates are then fed back to assist the phase error estimate in the next window. This forms a first-order feedback mechanism. As a result, the phase acquisition algorithm can use successive phase estimates to track a slowly varying phase offset. On the other hand, conventional PLL techniques would have to be second-order in order to deal with this frequency residue. The tracking
time required by a second-order feedback system is much longer than a first-order one [6], hence the phase algorithm used in this thesis is suitable for fast phase acquisition.

3.4 Performance Evaluations

Both phase acquisition algorithms discussed above are very efficient in computation. Excluding the handling of the buffered data, the early window method requires only two complex multiplications per symbol, which translates into 8 real multiplications and 4 real additions. The operations of filtering, scaling and averaging of the phase error estimate \( r_k \) take another 4 real multiplications and 4 real additions. Therefore, the total computational requirements for the early window phase acquisition are 12 real multiplications and 8 additions. Similarly, the late window method requires 16 multiplications and 10 additions. When compared to other functions in a practical system such as interleaving and error control coding, the computational burden of the two phase acquisition schemes can be considered to be very small. Also, the early window method is about 25% more efficient than the late window method in terms of computational requirements.

Similar to Chapter 2, symbol error probability is used as the main criterion to evaluate the performance of the two phase tracking algorithms under various normalized frequencies drifts defined as

\[
f_o T_s = \frac{f_o}{R}
\]  

(3.34)
in cycles per symbol, where \( f_o \) is the frequency offset in Hz, and \( f_m \) is the symbol rate. From the results in Gracie's thesis [5], the original early and late window phase recovery algorithms can track a frequency drift of up to 0.001 cycle per symbol when the signal-to-noise ratio is higher than 7 dB. However, the modified phase acquisition
schemes are intended to work in a much lower SNR range. Therefore, it is natural to first test and optimize the two algorithms under a slower frequency drift, and then to increase the frequency drift gradually to find out the maximum frequency drift they can handle.

![Symbol error rates with late data window, $2\pi f_s T_s = 0.001, N = 8$](image)

**Figure 3.9**: Performance comparison of the late window phase acquisition with and without low-pass filtering of the phase estimates.

Fig. 3.9 shows the error probability curves of the late window phase acquisition algorithms with and without the low-pass filtering of the phase estimate $r_k''$. The comparison is taken at a moderate frequency drift $2\pi f_o T_s = 0.001$ radians/symbol and $N = 8$. For the curve without filtering, it works quite well in the high SNR range with a loss of about 0.3 dB at $E_b/N_o = 9dB$. This agrees to the results in [5]. However, it performs poorly in the low SNR range due to the increased symbol
errors. On the other hand, the error probability curve with low-pass filtering can consistently track the theoretical limit closely at very low SNR values with a loss of only 0.1 dB in $E_b/N_0$. This verifies our claim that a low-pass filter can effectively suppress erroneous phase jumps due to incorrect symbol detection.

### 3.4.1 Window Size

From Equation (3.32), the phase offset residue $\gamma$ in Equation (3.31) is directly proportional to the window size $N$ used in the two phase acquisition schemes. A larger window size is better in cancelling channel noise. However, it also introduces larger phase error $\gamma$. On the other hand, a shorter window reduces the phase error $\gamma$ by sacrificing its ability to suppress noise.

Gracie [5] points out that the length of the message payload can greatly affect the overall system performance due to the decision-directed nature of the two algorithms. To better appreciate this statement, the first step in both phase acquisition schemes is to adjust the receive vector $r_k$ by the phase offset estimate from the previous window $\hat{\phi}_{n-1}$. The result is then used to determine the received symbol $\theta_k$. The correctness of the detected symbol $\hat{\theta}_k$ strongly depends on the quality of the previous phase estimate $\hat{\phi}_{n-1}$. If the difference between the phase offset estimate and the true phase offset exceeds $\pm \frac{\pi}{4}$, then the symbol decision and the new phase offset estimate for the $n$th window based on those erroneous $\hat{\theta}_k$ are likely to be wrong. Therefore, this chain reaction can lead to a catastrophic failure. Because the effect of a wrong symbol on the phase estimate can accumulate, the chance of failing is proportional to the length of the message when the system operates in a low SNR range. To better investigate this effect, two message sizes are used in simulations. The first size is a
550-bit message intends to model the transmission of a single ATM cell which has 53 bytes or 424 bits of data. The size of the second message is 5500 bits chosen to model the transmission of a super cell by putting 10 ATM cells together.

To search for the optimal window size $N$, we first look at the mean phase offset residue shown in Fig. 3.10. The results are obtained under a moderate frequency drift of 0.001 radians/symbol which is about one sixth of the maximum limit claimed by [5]. From Fig. 3.10, a common trend is obvious for all cases. When the SNR is low, significant mean phase error is left. Then it decreases gradually as the SNR increases.

The conclusion is that the phase acquisition is heavily affected by the quality of symbol detection. However, for the early window method, the residual phase error approaches an asymptotic non-zero limit at high SNR and the asymptotic limit appears to be proportional to the window size. The phase error difference between the curves for $N = 8$ and $N = 16$ is roughly constant. This is readily explained by Equation (3.32) which states that the phase error is proportional to the window size $N$. It is worth noting that performance of the two different message sizes are comparable in the high SNR range. But when the SNR is low, the phase error of the 5500-bit message is much larger than the 550-bit curve. This is because a longer message has a greater chance of accumulating enough symbol errors to cause the detector to make a wrong phase estimate.

For the late window method, significant improvement in the mean phase error over the early window method is observed. The mean phase error curves for different window sizes are much closer together than in the early window. In fact, all curves converge to zero as SNR increases. This confirms the earlier statement that the phase
Figure 3.10: Mean phase offset residue at $2\pi f_0 T_s = 0.001$ for (a) message size = 550 bits and (b) message size = 5500 bits.
error for the late window method is bounded by $\pm \Delta \phi$ and is expected to be close to zero on average. Therefore, the late window phase acquisition is insensitive to window size $N$. When the SNR is low, the mean phase error behaves differently depending on the message size. For a 550-bit message, the late window curve is consistently better than the best curve from the early window method by approximately 0.04 radians. For a 5500-bit message, similar performance gain is observed if the SNR is above 1 dB. However, if the SNR is below 1 dB, the mean phase error for the late window method deteriorates quickly, and merges with the curves from the early window method. This is not surprising because the late window method suffers the same penalty from poor symbol detection when the SNR gets too low.

The above discussions lead us to choose the optimal window size to be $N_{\text{opt}} = 8$ because the mean phase error is smaller for $N = 8$. An additional benefit from this choice of window size is that the formation of each new $e_n$ requires a division by $N$. Since $N = 8 = 2^3$, the division operation can be transformed into a series of right-shift operations which is much easier in DSP implementation.

Fig. 3.11 shows the symbol error probability of the early window phase acquisition scheme under various frequency drifts and $N = 8$. It works very well at a low frequency offset $2\pi f_o T_s = 0.001$ radians/symbol, following the theoretical limit closely to as low as -1 dB in $E_b/N_o$. The loss in signal-to-noise ratio is only 0.2 dB. As the frequency drift increases, however, the losses in SNR increase dramatically. When $E_b/N_o$ is above 4 dB, those losses remain approximately constant, varying from 0.5 dB for $2\pi f_o T_s = 0.00314$ to more than 1 dB for $2\pi f_o T_s = 0.00628$ for both sizes of messages. When $E_b/N_o$ gets below 4 dB, the early window method starts to move away from the theoretical curve, and breaks down at 2 dB and 3 dB for $2\pi f_o T_s =$
Figure 3.11: Symbol error rates of the early data window phase acquisition for various frequency offsets.

(a) Message size = 550 bits

(b) Message size = 5500 bits
$0.00314$ and $2\pi f_e T_s = 0.00628$, respectively.

Further insight can be gained by explaining the mean phase error of the early window method under the frequency drifts used above. The results are shown in Fig. 3.12. When the frequency drift is low, i.e. $2\pi f_e T_s = 0.001$, the mean phase error remains below $0.05$ radian until $1$ dB in $E_b/N_0$. This allows the performance to be close to the theoretical limit with a loss of only $0.2$ dB. Equation (3.32) states that the phase error for the early window method is $\Delta \phi + N\beta$, where $\beta$ is the rate of change in phase per symbol. Therefore, the increase in frequency drift translates into the step increases in the mean phase error floors, ranging from $0.025$ radians for $2\pi f_e T_s = 0.001$ to $0.17$ radians for $2\pi f_e T_s = 0.00628$. A phase error of $0.17$ radians in either direction will certainly have a serious impact on the symbol detection because the detection region for QPSK is only $\pm \frac{\pi}{4}$ radians.

By comparing Figs. 3.12 and 3.11, we can see that the mean phase error for different frequency drifts follows the same trend in the symbol error probability curves. It remains relatively constant until $4$ dB, then it deteriorates according to the waterfall pattern. A transition point can be identified to be approximately at $4$ dB. Above $4$ dB, the mean phase error changes very slowly, resulting the constant SNR losses between the curves in Fig. 3.11.

Fig. 3.13 shows the BER curves for the late window method under the same set of frequency drifts as in the early window case. The late window scheme works very well at $2\pi f_e T_s = 0.001$ with a loss in SNR of only $0.1$ dB compared to the $0.2$ dB loss for the early window method, also, the method can handle a higher frequency drift much better than the early window one does. The loss in SNR due to the increased frequency drift is significantly reduced to about half of that in the early window method. In
Figure 3.12: Mean phase offset residues for various frequency drifts.
Figure 3.13: Symbol error probabilities of the late data window phase acquisition for various frequency offsets.
particular, the loss at $E_b/N_0 = 9\, dB$ is about 0.25 dB for $2\pi f_o T_s = 0.00314$ and 0.5 dB for $2\pi f_o T_s = 0.00628$. Comparison between Fig. 3.13 and Fig. 3.11 reveals that they have approximately the same failure thresholds which are at 2 dB and 3 dB for $2\pi f_o T_s = 0.00314$ and $2\pi f_o T_s = 0.00628$, respectively. The reason for the two schemes to have the same failure threshold is that they use the same procedures to estimate a phase offset. Therefore, both should see a breakdown in the phase estimate at about the same SNR values.

Fig. 3.14 is the mean phase error for the late window phase acquisition at various frequency drifts. When compared to Fig. 3.12, the most obvious improvement is the convergence of the mean phase error to zero when the SNR is high. Also the phase error curves are much closer together than those in Fig. 3.12. This explains the reduction in SNR losses when the frequency drift is increased. Also the phase estimate breaks down at about the same SNR as in the early window one.
Figure 3.14: Mean phase offset error for various frequency drifts.
3.5 Summary

This chapter began by stating the importance of phase recovery in data detection. Then two new phase acquisition schemes based on phasor estimation suitable for use in the low SNR environment were discussed and evaluated under various frequency drifts. The simulation results indicated that both schemes could successfully track a slow frequency drift of \(2\pi f_o T_s = 0.001\) radians per symbol at as low as 0 dB in \(E_b/N_0\) with a loss of less than 0.1 dB in SNR. However, the late window phase acquisition scheme was better than the early window method in tracking a higher frequency offsets. The maximum normalized frequency offset they could handle was \(2\pi f_o T_s = 0.00628\). At that frequency offset, the loss in \(E_b/N_0\) for the late and early window schemes was approximately 0.5 dB and 1 dB, respectively.
Chapter 4

Convolutional Codes and Their Integration with Phase Acquisition

This chapter starts by introducing the structure of a typical convolutional encoder. Then it carries on to describe the Viterbi decoder as the optimal decoder for decoding convolutional codes. Finally, in Section 4.3, an algorithm that uses the convolutional decoder to assist the phase acquisition process in the demodulator is presented and analyzed.

4.1 Convolutional Encoder

Convolutional codes have been widely used in the satellite industry as an effective way to combat additive white Gaussian noise (AWGN). Fig. 4.1 shows a typical rate-1/2, $K = 3$ convolutional encoder. It consists of three shift registers and two modulo-2 adders. At each time instant, the source bit is fed into the left-most storage element and the contents in the rest of the register array are shifted one bit to the right. Then
Figure 4.1: Rate-1/2 convolutional encoder. The current input bit is shifted into the left-most storage element.

the sums of the two adders become the encoded bits at that time instant.

Often, an encoder takes in \( k \) source bits each time and generates \( n \) encoded bits. This kind of encoder is called rate \( k/n \) convolutional encoder, and is depicted [37] in

![Convolutional Encoder Diagram](image)

Figure 4.2: General convolutional encoder of rate \( k/n \).

Fig. 4.2. There are \( k \) sets of shift register arrays, each having \( m_i \) memory elements to store the most recent source input bits. The constraint length of the convolutional code is then defined as [25]

\[
K = 1 + \max_i m_i. \tag{4.1}
\]
The total memory of the convolutional encoder is the total number of memory elements in the encoder,

\[ M = \sum_i m_i. \]  (4.2)

At time \( l \), the encoder takes in a \( k \)-tuple input \( U_l = \{u_l^0, u_l^1, ..., u_l^{k-1}\} \) and generates a \( n \)-tuple encoded output \( X_l = \{x_l^0, x_l^1, ..., x_l^{n-1}\} \). The process can be described as [25]

\[ x_l^i = \sum_{j=0}^{m_i} \oplus u_{l-j} g_i^j, \]  (4.3)

where \( \sum \oplus \) denotes modulo-2 addition, and \( g_i \) is the generator sequence. The reason for the name "convolutional" is that Equation (4.3) is actually the discrete convolution between the input bit sequence \( x \) and the generator sequence \( g \).

### 4.2 Viterbi Algorithm

![State diagram for the rate-1/2 convolutional code.](image)

Figure 4.3: State diagram for the rate-1/2 convolutional code.

The state of an encoder can be described by the contents stored in the shift
register and the current input source bits at time $l$. Fig. 4.3 is the state diagram for the rate-$1/2$ convolutional coder in Fig. 4.1. The number inside each circle denotes the memory content of the encoder. Each branch is labeled with the source bit at time $l$ and the corresponding encoder output. Therefore, we can determine the output bits from a state diagram once the input bit is known.

![State Diagram](image)

Figure 4.4: Trellis diagram for the rate-$1/2$ convolutional code.

The Viterbi algorithm is based on a modified version of the state diagram, called the trellis diagram. It actually expands a state diagram along a time axis by tracing all possible input/output sequences and state transitions. The trellis diagram derived from the state diagram in Fig. 4.3 is shown in Fig. 4.4. When the first source bit is shifted into the register, the encoder can move either to state (10) if the input bit is 1, or to state (00) if the input bit is 0. When the next bit is received, the encoder state enters one of the four possible states (00), (01), (10) and (11). Similarly, the number of states is doubled for every additional bit received until it reaches the maximum $2^M$ states after $K - 1$ time instants (2 in this example). Then the trellis diagram
becomes repetitive until time $L - (K - 1)$ if $L$ source bits are shifted into the encoder [3].

The maximum likelihood decoding algorithm [25] attempts to calculate the likelihood values for all possible paths in a trellis diagram, and then chooses the one with the optimal likelihood value as the decoded codeword. This would involve formidable computation if the number of bits $L$ is large. However, the Viterbi algorithm makes use of the repetitive structure of the trellis diagram to simplify this maximum likelihood decoding process. If the channel errors are random, then the paths that are not optimal at the current stage can never become the optimal choice in the future. Therefore, in the Viterbi algorithm, only one path is chosen among the $2^k$ paths (1 for $k = 1$ in this example) entering each state. As a result, only $2^{k(K-1)}$ paths need to be stored at each time instant along with their corresponding likelihood values. This process is repeated until the end of the codeword is reached. The final decision is chosen as the path with the minimum or maximum likelihood value depending on the likelihood function [37].

### 4.3 Implementation of Viterbi Algorithm

With the above explanations on convolutional codes and the corresponding Viterbi decoding algorithm, we now move on to discuss the issues involved in the integration of this convolutional code to the timing and phase synchronization algorithms in Chapters 2 and 3. Since the rate-1/2, K=7 convolutional code is often regarded as the de facto standard for commercial satellite applications, we will implement that convolutional code using a set of programs originally developed by Joubin Karimi[26].
4.3.1 Hard Decision Decoder

In a hard decision decoder, a binary decision on the received signal is made based on its sign. If it is greater than zero, then a "1" is shifted into the decoder. Otherwise, a "0" is assumed to be the transmitted signal. The encoder then calculates the likelihood function based on those hard decision values. The hard decision process can be visually summarized by Fig. 4.5.

In practice, the signal that will undergo a "hard decision" must be reasonably free of both timing and phase error. As a first attempt, we implement the rate-1/2, $K = 7$ hard decision convolutional decoder by directly connecting it to the timing and phase acquisition modules as shown in Fig. 4.6. From the previous two chapters, a synchronization preamble is attached to the front of each message payload being sent. This preamble can be further divided into two parts. The first eight symbols are used for timing acquisition and the following sixteen symbols are used for phase acquisition. The late window phase acquisition algorithm with a data window size of eight is also used in this chapter for its superior performance over the one with
Figure 4.7: Performance of hard decision decoder in Fig. 4.6 under various frequency offsets.

Early window decision. Maximum likelihood decoding requires decisions on decoded bits be taken after looking at the entire codeword. However, this is neither possible nor necessary in practice. The decoded bits can be taken $\Gamma$ bits after their arrival at the receiver. $\Gamma$ is called the decoding depth for the decoder and is usually five to ten times of the constraint length $K$ [25]. Throughout this chapter, the decoding depth $\Gamma$ is chosen to be seven times the constraint length. The resulting bit error probability curves are shown in Fig. 4.7 for various frequency offsets. Each 550-bit message is sent with a different timing offset, and the timing offset stays unchanged over the transmission of that message. Clearly, the approach is unacceptable because the SNR loss is more than 1.3 dB for a frequency offset of $2\pi f_o T_s = 0.00628$ radians per symbol.

With perfect phase acquisition, the same set of simulations in Fig. 4.7 were
Figure 4.8: Bit error curves of the rate-1/2, $K = 7$ hard convolutional decoder with perfect phase acquisition.

taken, and the resulting bit error rate curves are plotted in Fig. 4.8. This clearly shows that the bit error rate curves are very close to the expected results when phase acquisition is perfect and the receiver only performs timing acquisition. Therefore, the poor performance in Fig. 4.7 is mainly due to the failure of the phase acquisition algorithm. With a convolutional code, both the timing and phase synchronization schemes operate in a very low SNR range, from 0 dB to about 4 dB in $E_c/N_o$ where $E_c$ is the coded symbol energy per bit. As a result, the uncoded symbol detection of the coded symbols is highly unreliable. However, the phase acquisition algorithm described in Chapter 3 is based on a decision-directed scheme. The poor uncoded symbol detection hinders its phase tracking ability. Our solution to this problem is to use the decoded symbols from the convolutional decoder to help the extraction of
phase offset in the phase acquisition process as shown in Fig. 4.9.

![Block diagram for using the decoder output to assist phase acquisition.](image)

Figure 4.9: Block diagram for using the decoder output to assist phase acquisition.

In order to accommodate the decoded symbols that are fed back, the phase acquisition algorithm has to be modified accordingly. Also, from Chapter 3, we know that the phase acquisition with late window decision is superior to that with an early window decision. Therefore, we will focus on the use of the late window phase acquisition algorithm. The updated block diagram of this phase algorithm is shown in Fig. 4.10. In the beginning of the phase acquisition process, it operates just as described in Chapter 3. The phase corrected signal from the phase acquisition module is fed to the decoder. After the decoder receives $m$ symbols, a preliminary decoded symbol decision can be made. This early symbol estimate $\hat{\theta}_{k-m}$ is then fed back to the phase recovery unit to remove the symbol information $\hat{\theta}_{k-m}$ from the received signal. Realizing that this early symbol is delayed by $m$ symbol periods, the phase acquisition unit has to store its received signal $r_k$ in a buffer. Once the preliminary symbol estimate is available ($\text{feedback-symbol-ready} = 1$ in Fig. 4.10), it switches to work on the delayed input $r_{k-m}$. However, this switch of $r_k$ increases the residual phase error between the phase estimate $B_{n-1}e^{j\theta_{n-1}}$ and the true phase offset of $r_{k-N}$ by $m\beta$, where $\beta$ is the per symbol increase in phase. It is noted that the preliminary symbol estimate is more reliable if the decoder can take in more symbols before an early decision is made.
Figure 4.10: Modified late window phase acquisition.
Therefore, there is a trade-off between the accuracy of the feedback symbols and the resulting phase residual error. By computer simulations, it was found that the best performance occurs for $m = 8$. Therefore, in the rest of this chapter, the early symbol estimate is fed back to the phase acquisition process after the decoder receives eight symbols.

For the modified phase acquisition scheme shown in Fig. 4.10, a new phase estimate is computed for every symbol received. At $t = kT_s$, the two summation buffers hold the phase estimates from $t = (k - N)T_s$ to $t = kT_s$. Hence their average is an estimate of the phase offset at $t = (k - N/2)T_s$ for a constant frequency offset. This process is illustrated in Fig. 4.11. The frequency offset should not change much over a short time duration of $N = 8$ symbols, and thus can be closely approximated to be constant. The length of the buffer used to compute $U_{w,kN/2}$ is then set to $N/2$ to match the delayed input vector $r_{rN/2}$ to the phase offset estimate $B_{k-N/2}e^{j\dot{\beta}k-N/2}$ in order to reduce the residual phase offset.

![Figure 4.11: Average of phase estimates.](image-url)

In Fig. 4.10, a modified Data Decision & Phase Mapping unit is used. Before the feedback symbol from the decoder is available ($feedback-symbol-ready = 0$), it is used to demodulate the signal $r'_k$ and to return the ideal signal constellation point $e^{j\dot{\beta}k}$ according to the symbol decision. Once the feedback symbol from the decoder is
available \( (feedback-symbol-ready = 1) \), it returns the signal constellation point \( e^{j\hat{\theta}_k-m} \) corresponding to the symbol from the decoder.

Pilot symbols will also be used to guide the phase acquisition process. As stated earlier, phase acquisition depends heavily on the accuracy of decoded symbol estimate. Burst errors can easily move the phase offset estimate into an incorrect quadrant and as a result, the subsequent detected symbols will be in error as well. We cannot avoid these burst errors, especially in the very low SNR range. However, we can bound the harmful effect of an incorrect phase estimate by inserting \( M \) periodic pilot symbols for every \( L \) transmitted symbols. Also, we can amplify the effect of these pilot symbols by increasing their weighting relative to the normal data symbols. Simulation results showed that we need to insert 2 symbols for every 100 symbols transmitted for good performance. Increasing pilot symbols more frequently not only wastes the channel capacity, but also diminishes the performance gain. The modified phase estimate input to the low-pass filters will be:

\[
\tau_k^{'''} = w_k \tau_k^{''},
\]

where

\[
w_k = \begin{cases} 
W_{\text{pilot}} : & k \in [iL, iL + M - 1], \quad i = 0, 1, 2, \ldots \\
W_{\text{data}} : & \text{otherwise}.
\end{cases}
\]  

As a quality control measure, we can also monitor the phase offset extracted from \( \tau_k^{'''} \). Immediately after the preamble of known symbols ends, the phase offset estimate should be quite close to the true phase offset. If the data symbol detection \( \hat{\theta}_k \) is correct, then the phase of \( \tau_k^{'''} \) in Equation (4.4) should not be significantly different from the previous phase estimate \( \hat{\phi}_{n-1} \). However, \( \Delta \tau_k^{'''} \) can be at least \( \frac{\pi}{2} \) radians away from the previous phase estimate \( \hat{\phi}_{n-1} \). Therefore, the comparison between the two
can easily identify the abrupt change of phase estimate. And actions can be taken to limit its impact on further phase estimates. The flow diagram for this comparison is shown in Fig. 4.12. If the phase difference is greater than the threshold $Th_{phase}$, then

![Flow diagram of the phase comparison](image)

**Figure 4.12: Flow diagram of the phase comparison.**

that $r_k'''$ will be discarded, and will not be shifted into the low-pass filter used for phase estimation in Fig. 4.10, otherwise, $r_k'''$ will be deemed acceptable and will be filtered by this low-pass filter to further reduce the phase change due to Gaussian noise. By trial and error, the best bit error rate is obtained when $Th_{phase}$ is set to 1 radians. Initially, one might think that the threshold should be kept very small. The reason for this large threshold is that the Gaussian noise can also introduce significant phase fluctuation. In a noisy channel, this variation in phase can be quite large from symbol
to symbol and it can be suppressed by the low-pass filters in Fig. 4.10. Therefore, a larger than expected threshold is required. $Th_{\text{phase}} = 1$ is a good compromise between allowing those acceptable phase estimates to pass and still blocking those unacceptable phase estimates due to erroneous symbol detection.

![Bit error rate for rate 1/2, K=7 hard convolutional code with pilot symbols and quality control on phase estimate](image)

Figure 4.13: Bit error rate of the hard decision convolutional code with pilot symbol and quality control on phase estimate.

After the implementation of the above modifications to the phase acquisition algorithm given in Fig. 4.10, the same set of simulations considered earlier in Fig. 4.7 were repeated under various frequency offset conditions. The resulting bit error rates are shown in Fig. 4.13. When compared to Fig. 4.7, we can see that these extra steps can reduce the SNR loss to just 0.7 dB compared to the loss of 1.3 dB in the earlier case when the frequency offset is 0.000628 radians per symbol. Similar
improvement is also observed for the other two frequency offsets. In particular, the SNR loss for \(2\pi f_c T_s = 0.001\) is only 0.3 dB.

![Figure 4.14: Relative time relation between the two iterations.](image_url)

However, the SNR loss under higher frequency offset is still quite large. To further reduce the SNR loss, we can process the message iteratively. The first iteration of this process follows the same steps described above. During this iteration, the input signal to the phase acquisition \(r_k\) is recorded in memory. Similarly, the final decoded symbols from the convolutional decoder are stored in memory as well. Note that these decoded symbol estimates are taken \(\Gamma\) bits after their arrival time at the decoder. Therefore, these symbol estimates are more reliable than the preliminary ones used in the first round of phase acquisition. When the message is processed again in the second iteration, we use the stored decoded symbols to assist the phase acquisition in the same way as in the first iteration. The only difference is that in the second iteration, the reliable symbol estimate is available immediately after the phase preamble ends because it is not necessary to wait for the end of the first iteration of decoding before starting the second iteration. From Fig. 4.14, we can see that the second iteration can be started after \(\Gamma\) bits are received by the decoder (\(\Gamma\) input bits are stored in a buffer for reprocessing). Therefore, the penalty of the extra time delay for the second iteration is small. This is important in a burst system in order to avoid collisions of burst messages.
Figure 4.15: Bit error rate for hard decision convolutional code with iterative decoding.
The bit error rate curves for processing the message twice are shown in Fig. 4.15 along with the bit error rate curves from Fig. 4.13. By two iterations, we can reduce loss in SNR from 0.65 dB at the end of the first iteration to just 0.5 dB at the end of the second iteration when the frequency offset is $2\pi f_o T_s = 0.00628$. For smaller frequency offsets, the performance is also improved. The loss in SNR is merely 0.15 dB for $2\pi f_o T_s = 0.001$ and 0.3 dB for $2\pi f_o T_s = 0.003$. However, there is no further performance gain for processing the message with three iterations. Close analysis shows that the second iteration can correct most of the phase-related errors from the first iteration. Processing the message one more time does not help. Therefore, we conclude that two iterations of processing are needed to decode a message with an acceptable bit error rate performance.

4.3.2 Soft Decision Decoder

In practice, convolutional decoders with soft decision are almost universally used due to the approximately 2 dB coding gain compared to the hard decision decoder

<table>
<thead>
<tr>
<th>Quantized levels</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thresholds</td>
<td>-∞</td>
<td>-3Q</td>
<td>-2Q</td>
<td>-Q</td>
<td>0</td>
<td>Q</td>
<td>2Q</td>
<td>3Q</td>
</tr>
</tbody>
</table>

Figure 4.16: 8-level Uniform Quantization.

with only a moderate increase in decoder complexity [3]. The idea of soft decision is to quantize the decoder input to more than two levels as in Fig. 4.16 depending on the signal strength. For an AWGN channel, Heller and Jacobs [27] showed that
uniform 8-level quantization is only 0.25 dB worse than that of infinite quantization. Labeling the spacing between quantization levels by $Q$, we can calculate the soft decision likelihood metrics according to Table 4.1. For example, if the input signal is between $-2Q$ and $-Q$, then the quantized input bit would be 5, and the metric would be 5 if the expected input bit is 1. With these metric assignments, the path with the smallest metric value will be declared as the decoded codeword.

<table>
<thead>
<tr>
<th>Metric Assignments</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantized bit</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Expected bit = 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Expected bit = 0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Metrics Assignment for Soft Decision.

With perfect timing and phase synchronization and a frequency offset of 0.00628 radians per symbol, the bit error rate curves for various $Q$ values are shown in Fig. 4.17. Since the published results for the rate 1/2, $K = 7$ soft convolutional code in [27] are not available for $E_b/N_o$ below 3 dB, the original source programs by Joubin Karimi [26] were run to collect the necessary sample points below 3 dB. We can see that the optimal quantization interval for soft decision is $Q = 0.315$. And the SNR loss in that case is negligible.

After the quantization interval is determined, we should first look at the bit error rate curve in Fig. 4.18 which is obtained by tracking only the timing offset under perfect phase synchronization. The loss of $E_b/N_o$ due to timing acquisition is about 0.2 dB at $BER = 5 \times 10^{-5}$. Employing the same techniques used for the hard decision decoder, the simulations in Fig. 4.15 were conducted again in which the receiver performed both timing and phase acquisition, and the results are shown in
Figure 4.17: Bit error rate of soft decision convolutional code for various quantization level spacing $Q$. 
Figure 4.18: Bit error rate curve for soft convolutional decoder with perfect phase acquisition.

Fig. 4.19 for various frequency offsets. In order to achieve a stable bit error rate, we have to increase the frequency of the pilot symbols from two symbols for every 100 encoded symbols to one symbol for every 50 encoded symbols. However, this implies that only 2% of the channel capacity is set aside for aiding the phase synchronization. Similar to the hard decision case, processing a message one more time can greatly reduce the SNR loss by one half for all the frequency offsets tested. The loss in $E_b/N_0$ at $BER = 10^{-4}$ for $2\pi f_o T_s = 0.00628, 0.003$ and $0.001$ is approximately $0.9$ dB, $0.64$ dB and $0.4$ dB respectively. These losses are higher than those of the hard decision decoder. However, one should note that the soft decision decoder operates at lower $E_b/N_0$ than the hard decoder. Thus the timing and phase estimation algorithms must operate at $3$ dB lower than the $E_b/N_0$ value given in Fig. 4.15, and this certainly makes the phase estimation process more difficult than that in the hard decision decoding case.
Bit error rates for rate 1/2, K=7 soft convolutional code with quality control on phase estimate, two iteration of decoding and 1 pilot symbols for every 50 encoded symbols.

Figure 4.19: Bit error rate for soft decision convolutional code with techniques to improve phase acquisition.
4.4 Summary

This chapter began with the descriptions of the convolutional codes and the corresponding Viterbi decoding algorithm. It then illustrated that poor performance would result if the convolutional decoder was not properly integrated with the synchronization procedures, especially for the phase acquisition algorithm. Methods of using the decoder output to assist the phase acquisition process were discussed. Those methods were iterative decoding, pilot symbols, quality control of phase estimates and matching of the phase estimate. They were shown to be effective by simulations.
Chapter 5

Conclusions

5.1 Conclusions

- Two new burst acquisition algorithms, namely the mixed signal (MS) timing detector and non-mixed signal (NMS) timing detector, were developed for burst QPSK transmission with the Nyquist sampling rate based on a zero-crossing timing error detection formula. Both methods require a timing preamble of only eight symbols with an alternating pattern for good performance. The fast timing acquisition is achieved by the use of an interpolation filter bank which allows the receiver to simultaneously monitor all the timing error signals associated with each filter in the filter bank and choose the one with the optimal timing offset.

- Performance tests of the two timing algorithms were conducted by simulations for an AWGN channel. The timing acquisition algorithms can be applied to only the timing preamble to estimate a fixed timing offset over a short message.
In that case, they were shown to be very effective, and performed equally well when tracking a fixed timing offset at $E_b/N_o$ as low as -3 dB with a loss of only 0.2 dB in SNR. However, the NMS timing detector was shown to be better than the MS detector for $E_b/N_o$ below 4 dB when they were used in the data mode to combat a continuously varying timing offset. Also, the loss of SNR for using the NMS detector in the data mode was reduced to merely 0.1 dB for $E_b/N_o$ from -3 dB to 8 dB.

- It was found that the timing acquisition algorithms could be made more robust by checking the energy of the samples from the chosen interpolation filter at the end of the timing acquisition procedure. This energy check can identify the possible "false" timing lock at the zero-crossing point of the received signal.

- Two simple burst phase acquisition algorithms were derived from a previously known method for burst transmission and for use in a much lower range of $E_b/N_o$. The two methods are called phase acquisition with early window decision and phase acquisition with late window decision, respectively. They work entirely on phasors and are very efficient for DSP implementation. It was found that the use of low-pass filter could effectively minimize the adverse effect of erroneous symbol detection on the phase offset estimate, enabling proper operation of the phase acquisition algorithms in a noisy environment. With a normalized frequency offset of $f_o T_s = 10^{-3}$, the loss of SNR at an error rate of $P_e = 10^{-4}$ was 0.5 dB for the phase acquisition with late window decision, or 1 dB for the one with early window decision.

- Several techniques of using the Viterbi decoder to assist the phase acquisition
process were investigated. Simulation results showed that these techniques ensured the quality of the phase estimate. The loss in $E_b/N_o$ for $f_oT_s = 0.001$ was 0.5 dB for the hard decision decoder and 0.9 dB for the soft decision decoder. The loss for the soft decoder was higher than that for the hard decoder because the soft decoder operated at a lower $E_b/N_o$ for the same BER, which stressed the synchronization process more than for a hard decoder.

5.2 Suggestions for Further Study

- Since the burst timing acquisition has been shown to work well with QPSK, it should be interesting to apply the method to other $m$PSK modulations such as 8PSK for better bandwidth efficiency.

- Further refining of the phase acquisition algorithm is possible. One suggestion would be to apply the open-loop phase acquisition algorithm along with a phase-locked loop (PLL). A PLL has better performance than an open-loop method in tracking a fast varying phase if it can start with a reasonably reliable phase offset estimate. This suggests that we can use the phase acquisition algorithms in this thesis to quickly get an early estimate of the phase. And in the data mode, this early phase estimate can help a PLL to start.

- The phase acquisition algorithms in this thesis work very well for small frequency offset. This suggests that a good frequency estimation algorithm can be added to a receiver to keep the residual frequency offset within an acceptable range.

- Turbo codes can provide excellent bit error probability performance that approaches the Shannon channel capacity at very low SNRs. However, such results
are obtained under the assumption of perfect synchronization. Since the timing algorithm developed in this thesis can operate at $E_b/N_0$ of -3 dB, it should be possible to use the timing algorithm to synchronize a Turbo coded message. However, doing phase acquisition at these low values of $E_b/N_0$ will prove to be challenging.
Bibliography


