A Multi-functional Digital Receiver with Adaptive Filtering and Bit Synchronization

Stephen Powell

Department of Electronics

Carleton University

May 5, 2003

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisisitons et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-83477-8

Our file Notre référence

ISBN: 0-612-83477-8

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.

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.

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 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.

Canada

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

**A Multi-functional Digital Receiver with
Adaptive Filtering and Bit Synchronization**

Submitted by Stephen Powell, B.Eng. in partial fulfillment
of the requirements for the degree of

Master of Applied Science

Thesis Co-Supervisor

Thesis Co-Supervisor

Chair, Department of Electronics

Carleton University, May 2003

Abstract

A multi-functional digital receiver with adaptive filtering and bit synchronization was designed, implemented and tested. This receiver is capable of acquiring, tracking and demodulating signals in a noisy environment, confidently down to 0dB E_b/N_O . The types of modulation that can be processed are BPSK, QPSK and OQPSK. Adaptive filtering improves PLL performance under certain conditions and a novel way to achieve bit synchronization is implemented with excellent results.

A *MATLAB* model of the complete receiver was simulated in order to verify its functionality. A BPSK version of the receiver was coded into *C* and included in the Mobile Knowledge CDGPS product.

Acknowledgements

This thesis would not have been possible without two kind individuals and one kind entity:

First and foremost, Gyles Panther, my supervisor at Mobile Knowledge. Many thanks for your guidance and your endless supply of investigatory questions. It was great working with you.

Dr. Jim Wight, my supervisor at Carleton. Thanks for presenting me with the opportunity to work with Gyles, and for polishing my thesis.

Mobile Knowledge (formerly Sigem). Thanks to David Levy and company for financially supporting me in my endeavor to become a Masters graduate.

Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	vii
List of Tables	ix
List of Acronyms	x
1 Introduction	1
1.1 Objectives	3
1.2 Thesis Organization	4
2 Background	6
2.1 Modulation Schemes	7
2.1.1 BPSK	7
2.1.2 QPSK	8
2.1.3 OQPSK	8
2.2 Receiver Requirements	10
2.2.1 Hardware	10
2.2.2 The Digital Receiver	11

2.2.3	Data Channel Specifications	13
2.2.4	Receiver Flow	13
3	The Phase Locked Loop	15
3.1	Obtaining Carrier Phase	16
3.2	Phase Detector	17
3.3	VCO, NCO	18
3.3.1	Analog (VCO) Phase Increment	19
3.3.2	Digital (NCO) Phase Increment	19
3.4	Loop Filter	20
3.4.1	Design of the DPLL Loop Filter	22
3.5	Z-Domain Model	26
3.5.1	Stability	28
3.5.2	Noise Bandwidth	29
3.5.3	Stimulus Response	32
3.6	Implemented Algorithm	33
4	Remodulator	34
4.1	BPSK Remodulator	37
4.1.1	BPSK Components	37
4.2	QPSK, OQPSK Remodulator	40
4.3	Remodulator Algorithms	42
4.3.1	Search Algorithm	42
4.3.2	Locking Algorithm	45
4.3.3	Adaptive Filtering Algorithm	49
5	Bit Synchronization	52
6	Simulations	57
6.1	Signal Generation	58

6.1.1	Noise Generation	58
6.1.2	Data Generation	62
6.2	Simulation Results	65
6.2.1	Receiver Performance	66
6.2.2	Effects of Quantization	70
6.2.3	Pattern Sensitivity	72
6.2.4	Real life Data	75
7	Conclusions	77
7.1	Suggestions for Further Work	79
	 Appendices	 80
A	Computer Implementation	80
B	Further Noise Explanation	83
C	Derivations	86
C.1	Variance of a Filtered Signal	87
C.2	Variance due to Drift Rate	88
D	Additional Simulation Results	89

List of Figures

2.1	BPSK Constellation	7
2.2	QPSK Constellation	8
2.3	OQPSK Constellation	9
2.4	Receiver Down-conversion	11
2.5	Receiver Control Flow	14
3.1	Phase Locked Loop	16
3.2	DTL S-Curve with Various SNR	18
3.3	PLL Basic Model	22
3.4	70Hz Filter Magnitude and Phase Response	23
3.5	Pole and Zero Plot of 70Hz Filter	24
3.6	IIR Filter Block Diagram	27
3.7	Block Diagram of Implemented Digital PLL	27
3.8	PDF of PLL Frequency Error	31
4.1	Block Diagram of BPSK Remodulator	35
4.2	Block Diagram of QPSK Remodulator	40
4.3	Histogram Frequency Bucket Setup	42
4.4	Histogram Frequency Sweep	43
4.5	Histogram Results	43
4.6	Circular Buffer	46
4.7	Variance Threshholds	48

4.8	Locking and Adaptive Filtering Algorithm Sample Run	51
5.1	Bit Sync Table	54
5.2	Bit Synchronization Logic Flow	56
6.1	Generation of a BPSK Signal	59
6.2	Filtered Noise	60
6.3	Up-Conversion of Noise	61
6.4	RRC Filtered Data Signal	62
6.5	Locking % vs. Modulation Schemes for a 20Hz Carrier Offset	66
6.6	Locking % vs. Frequency for a QPSK Signal	67
6.7	Time to Lock vs. Modulation for a 20Hz Carrier Offset	68
6.8	False Locking vs. Modulation for a 20Hz Carrier Offset	68
6.9	BER for Different Modulated Signals at a 0Hz Carrier Offset	69
6.10	Locking % vs. Quantization for a QPSK Signal with a 10Hz Carrier Offset	70
6.11	BER for Different Quantizations of a QPSK Signal with a 0Hz Carrier Offset	71
6.12	Locking % vs. Patterned Data on a QPSK Signal with a 20Hz Carrier offset	73
6.13	BER for Patterned Data on a QPSK signal with a 0Hz Carrier Offset	74
6.14	BER for BPSK Modulated Satellite Data	75

List of Tables

3.1	Filter Coefficients, Zeros and Poles	25
3.2	Noise Bandwidths	29
3.3	PLL Steady State Response to an Input	32
4.1	Locking Variances due to Filters	50
D.1	BER for Different Modulations at a 0Hz Carrier Offset	90
D.2	QPSK BER for Different Quantizations at a 0Hz Carrier Offset . . .	91
D.3	OQPSK BER for Different Quantizations at a 0Hz Carrier Offset . .	91
D.4	QPSK BER for Different Sampling Rates at a 0Hz Carrier Offset . .	92
D.5	BPSK BER for Different Patterns at a 0Hz Carrier Offset	92
D.6	QPSK BER for Different Patterns at a 0Hz Carrier Offset	93
D.7	OQPSK BER for Different Patterns at a 0Hz Carrier Offset	93

List of Acronyms

A/D	Analog to Digital Converter
BPSK	Binary Phase-Shift Keying
CDGPS	Canadian Differential Global Positioning Service
D/A	Digital to Analog Converter
DPLL	Digital Phase Locked Loop
DSP	Digital Signal Processor/Processing
DTL	Digital Tanlock Loop
FIR	Finite Impulse Response
GPS	Global Positioning Service
IF	Intermediate Frequency
IIR	Infinite Impulse Response
ISI	Inter-Symbol Interference
LO	Local Oscillator
LPF	Low Pass Filter
MSK	Minimum-Shift Keying
NCO	Numerically Controlled Oscillator
OQPSK	Offset Quadrature Phase-Shift Keying
PD	Phase Detector
PLL	Phase Locked Loop
PSK	Phase-Shift Keying
QPSK	Quadrature Phase-Shift Keying
RRC	Root Raised Cosine
SNR	Signal-to-Noise Ratio
VCO	Voltage Controlled Oscillator

Chapter 1

Introduction

Telecommunications is a rapidly evolving engineering science. With the increase of processing power and chip complexity, more and more products are leaning towards the all-in-one multi-functional IC: complex receiver products are fully integrated on a single receiver/transmitter and processing chip, such as Bluetooth. These tiny chips allow for very small products, such as GPS watches and cell phones. That being said, all wireless units share some basic necessities:

- **an antenna:** to receive/transmit a signal
- **a transceiver:** to convert a signal into means for transmission or reception, half or full duplex
- **a digital signal processor:** to analyze/produce the data signal

There exists novel new technologies in each of these items, however for the expanse of this thesis, we will assume a generic hardware receiver layout, and investigate the DSP possibilities.

For complete control over an incoming signal, the DSP must:

- acquire the signal
- track the signal
- extract the data from the signal

The advantages of delegating more responsibilities to the DSP are guaranteed system stability, predicted responses, software flexibility and good noise immunity [1].

Different carrier synchronizers can be implemented in a digital receiver, such as two different implementations of the Costas loop, the normal and the crossover [2], MAP estimation loop, remodulator loop and squaring loop [3]. While Costas loops

are more well known, remodulator loops offer good performance and are generally implemented at lower than IF frequencies, allowing for an easy digital baseband realization [4].

1.1 Objectives

Mobile Knowledge (formally Sigem) was developing a new product, called the Canadian Differential Global Positioning System (CDGPS), and was looking for a method to track the low level signals emanating from the orbiting satellites.

The architecture proposed included the implementation of a digital demodulator at the back end of an analog down-conversion. Due to the type of very low level signals the receiver would have to track, the main requirement was that it would have to operate successfully for very low SNR. Although BPSK was the modulation format of the satellite data signals, it was decided to expand the receiver to accept QPSK and OQPSK modulation in order for it to be more generic. Providing excellent low level signal tracking makes it a good choice in applications such as indoor GPS or any other situation where SNR is at a premium.

The main criteria of the receiver was to implement a PLL carrier-tracking demodulator which could establish bit synchronization.

The amount of jitter/phase noise present on a recovered carrier is related to the bandwidth of the PLL. However, the acquisition time is inversely related, so a method to optimize the performance of the PLL needed to be established. A second method was also needed to help reduce the probability of a false lock, a characteristic of all PLLs.

Bit synchronization is another essential criteria of a receiver. For the extent of this receiver, clock synchronization is assumed, but bit synchronization is not. A technique to establish the bit timing had to be developed in order for efficient data recovery.

A fully working BPSK version was to be coded in *C* and implemented for use in the CDGPS, while the QPSK and OQPSK versions were to be implemented in *MATLAB* and simulated.

1.2 Thesis Organization

First, Chapter 2 provides an introduction to the modulation formats this receiver is being designed for, and introduces the basic hardware layout for a typical receiver front end. The specifications of the digital receiver to be implemented are also listed.

Chapter 3 presents the digital PLL designed for this receiver. Analysis of each component, as well as an overall analysis of the entire PLL is performed, including the derivation of the *Z*-Domain models. The PLL *Z*-Domain model is used to ensure PLL stability, as well as to calculate noise bandwidth and view the responses of different input stimuli.

Chapter 4 presents the BPSK, QPSK and OQPSK remodulators and the components they require to function. It also presents three algorithms: a frequency searching algorithm for speeding up the PLL acquisition time, a locking algorithm to determine the lock status of the PLL, and an adaptive filtering algorithm to determine which loop filter should be used, and if the loop bandwidth should be increased/decreased.

Chapter 5 presents the novel bit synchronization technique developed.

Chapter 6 describes the generation of the signal required to test the digital receiver. It also presents the simulation results, including BER and the effectiveness of the lock algorithm for signals at both different SNR and quantization levels, and also signals with patterned data.

Chapter 7 provides a summary of this thesis' findings, and provides suggestions for additional work.

Chapter 2

Background

2.1 Modulation Schemes

Several modulation techniques exist, but in satellite communications, the most common forms of modulation are different types of PSK (Phase-Shift Keying), namely BPSK (Binary Phase-Shift Keying), QPSK (Quadrature Phase-Shift Keying) and OQPSK (Offset-QPSK). MSK (Minimum-Shift Keying) and $\pi/4$ QPSK are also used, but these modulation formats are not implemented in this receiver. Along with these modulation schemes, there also exists coding techniques, such as differential, convolutional and block (among others), which serve to improve channel performance.

2.1.1 BPSK

Binary Phase-Shift Keying is the simplest form of PSK. It comprises of a single carrier multiplied by a baseband data stream, i.e. $d_1 \cdot \cos(\omega_C t)$, where d_1 is ± 1 . It can also be represented as $A \cdot \cos(\omega_C t + \Theta(t))$, where $\Theta(t)$ is either 0° or 180° . The BPSK symbol constellation is shown in Figure 2.1.

When mixing down a BPSK signal in quadrature (multiplying with sine and cosine LO (Local Oscillator) signals), the new signals become:

$$I_S = d_1 \cdot \cos(\omega_{RR}t + \theta) \quad (2.1a)$$

$$Q_S = d_1 \cdot \sin(\omega_{RR}t + \theta) \quad (2.1b)$$

where ω_{RR} is the residual carrier after down-conversion, and θ is a constant phase.

Downconversion in quadrature is used in this receiver as a means to generate the carrier's phase. This will become apparent later, in the PLL phase detector section.

Figure 2.1: BPSK Constellation

2.1.2 QPSK

Quadrature Phase-Shift Keying is an extension to BPSK. It can be represented by either two BPSK signals in quadrature, $d_1 \cdot \sin(\omega_C t) + d_2 \cdot \cos(\omega_C t)$, where d_1, d_2 represent two independent data streams with symbols ± 1 , or by $A \cdot \cos(\omega_C t + \Theta(t))$, where $\Theta(t)$ is $0^\circ, \pm 90^\circ, 180^\circ$. The QPSK symbol constellation is shown in Figure 2.2.

When mixed down in quadrature, the output signals become:

$$I_S = d_1 \cdot \cos(\omega_{RR} t) - d_2 \cdot \sin(\omega_{RR} t) \quad (2.2a)$$

$$Q_S = d_1 \cdot \sin(\omega_{RR} t) + d_2 \cdot \cos(\omega_{RR} t) \quad (2.2b)$$

The problem with these two modulation schemes (BPSK, QPSK) is that they both allow an ‘instantaneous’ carrier phase change of 180° . With filtering, these phase shifts will cause the modulation envelope to go to zero. Although a constant envelope can be reconstructed through non-linear amplification [1], spectral regrowth occurs as a result of the non-linearities producing phase/amplitude distortion. This distortion can only be avoided through the use of a highly linear amplifier.

Figure 2.2: QPSK Constellation

2.1.3 OQPSK

Offset QPSK is very much like QPSK, with only one difference. The signals are the same as equations (2.2), however the data streams are offset from each other. In QPSK, the symbol changes for d_1, d_2 occur at the same time. In OQPSK, the

symbol changes are offset from each other by $\frac{1}{2}$ a period; i.e. d_1 can change at a given time t , but d_2 can change only at $t + \frac{T_s}{2}$. This means that the maximum instantaneous phase change that can take place is $\pm 90^\circ$, as seen in Figure 2.3.

This form of modulation provides a more constant envelope, whose amplitude will drop a maximum of $\frac{1}{\sqrt{2}}$. This helps to avoid out-of-band interference with the reconstruction of the constant envelope.

Figure 2.3: OQPSK Constellation

2.2 Receiver Requirements

In every receiving system, an incoming signal is picked up by some kind of antenna at a high frequency, usually too high for the DSP (Digital Signal Processor). As digital signal processing is usually reserved for operating on lower frequencies, a down-conversion from the original high frequency must take place. There are two methods for down-conversion: super-heterodyne, which involves several down-conversion stages, and direct down-conversion, which involves one. Each down-conversion stage is characterized by ‘mixing’ the signal with an LO signal, whose frequency is chosen sometimes due to emission requirements, sometimes due to commonly used frequencies (i.e. cheap, generic components at a common IF). Super-heterodyne receivers have the advantage of providing gain in stages at several frequencies, as opposed to a high gain at a single frequency, whereas direct down-conversion receivers have the advantage of using just one LO.

2.2.1 Hardware

Shown in Figure 2.4 is a typical receiver front end. A signal is picked up through an antenna, or cable, and then is multiplied by a LO signal, $\cos(\omega_{LO}t)$, and band-pass filtered to an IF. After this first downconversion, the signal is then multiplied by another LO signal and its quadrature counterpart, $\sin(\omega_{IF}t)$ and $\cos(\omega_{IF}t)$. After low-pass filtering, two virtual baseband signals are present (see equations (2.1) and (2.2)), and are ready for analysis by the DSP.

These signals are introduced into the software through an A/D (Analog to Digital Converter). This A/D samples and quantizes the signal into values useful for signal processing. As seen later (Chapter 7: Simulations), determining the minimum number of quantization levels is very important. Using fewer levels can save complexity in the A/D, however it has an effect on the dynamic range and

Figure 2.4: Receiver Down-conversion

rounding/truncation of the signal.

The sampling rate (at which the A/D operates), F_s , must also be carefully chosen. Typically, the sampling rate is an integer multiple of the data rate, F_b , however as Meyr [5] has shown, this is not a necessity. Without this integer multiple, interpolation between samples is usually performed in order to obtain optimum results [6][7].

2.2.2 The Digital Receiver

There are two main types of receivers: non-coherent, and coherent. Non-coherent receivers, such as a differential detection receiver [1], require no replica of the carrier to demodulate the signal, however they do require 3dB higher SNR to achieve the same BER as a coherent receiver. Of the coherent receivers, several variations of

carrier tracking/synchronization loops exist, such as squaring (or $(\cdot)^N$) loops, Costas loops and remodulator loops, all of which can operate on either simple BPSK or N-PSK. For any of these loops to function, the digital receiver has the following requirements:

Timing reference: a (solid) reference time is needed in order to keep the time/frequency analysis constant, as a changing time reference produces a shift in frequency. This can be implemented through clock synchronizers [3]

Down-conversion to below $\frac{F_s}{16}$: some receivers operate on aliasing, and do not need any downconversion [8]. However, for the digital receiver implemented, the center of the downconverted received signal must be lower than $\frac{F_s}{16}$ (this maximum carrier residual frequency limit results from a filter in the remodulator).

The functioning criteria of the digital receiver are as follows:

Search the $[0, \frac{F_s}{16}]$ range for the carrier frequency: if the signal has not been downconverted properly, it might not be located in this range. As well as finding the signal, it should be able to lock onto the carrier quickly for a fast acquisition time.

Lock onto the present signal: for coherent data recovery, a replicate of the carrier signal needs to be generated. This is generated through a phase locked loop.

Ability to demodulate BPSK, QPSK, OQPSK: These modulation formats require different processing, and as a multi-functional receiver, the receiver must be able to extract the correct information for any of these three modulation methods.

Achieve bit synchronization: bit synchronization is needed in order to sample the recovered data at the points of least inter-symbol interference, providing optimal data recovery.

2.2.3 Data Channel Specifications

The data channel that this receiver has been designed for is a BPSK/QPSK/OQPSK modulated transmitted signal with a data rate of 4800bps for BPSK and 9600bps for QPSK, OQPSK. This requires a *minimum* data channel bandwidth of $\pm 2400\text{Hz}$. The downconverted noise spectrum will typically occupy a bandwidth of $\pm 4800\text{Hz}$ about the carrier.

2.2.4 Receiver Flow

Getting the receiver from the initial stage of powering on, to the final stage of outputting valid data requires certain things to happen. Figure 2.5 depicts the control flow for the receiver. Each part of this figure will be explained in further detail.

Figure 2.5: Receiver Control Flow

Chapter 3

The Phase Locked Loop

The phase locked loop is the most vital component of any coherent detection receiver. It has 3 components: a phase detector, a loop filter and a voltage/numerically controlled oscillator (VCO/NCO), see Figure 3.1 [9].

Figure 3.1: Phase Locked Loop

A PLL works by comparing the phase of an input signal to that of a reference (output) signal (generated by the VCO/NCO). The result of this comparison is an error signal, which passes through the loop filter and modulates the VCO/NCO in an attempt to match the phase of the reference (output) signal to the phase of the input signal. Once matched, the loop is essentially tracking (locked onto) the incoming signal; this is known as phase lock.

3.1 Obtaining Carrier Phase

For the PLL implemented in this receiver, a digital tanlock loop is used. Proposed by Lee and Un [10], the main advantages of this loop is that the phase detector response is linear from $[-\pi, \pi]$, and there is no need for an automatic gain control circuit [11]. The DTL achieves the phase of the incoming signal as follows:

Suppose incoming QPSK signals are of the form seen in equations (2.2) with some arbitrary phase $\Theta = \omega_{RR}t + \theta$:

$$I_S = d_1 \cos \Theta - d_2 \sin \Theta$$

$$Q_S = d_1 \sin \Theta + d_2 \cos \Theta$$

As these signals do not contain any *pure* carrier component, a method of carrier extraction must be used. Multiplying the signals by a replica of the data signals, d_1 and d_2 (assume the data is known), as in equations (3.1) and (3.2), and performing the division results in $\frac{\sin \Theta}{\cos \Theta}$.

$$A = I_S \times d_1 + Q_S \times d_2 \tag{3.1}$$

$$= (d_1^2 + d_2^2) \cdot \cos \Theta$$

$$B = I_S \times d_2 - Q_S \times d_1 \tag{3.2}$$

$$= (d_1^2 + d_2^2) \cdot \sin \Theta$$

Performing $\arctan(\frac{B}{A})$ allows the determination of the phase of the signal between $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Furthermore, knowing the signs of A and B allow one to determine what angle (between $[-\pi, \pi]$) the incoming signal is at compared to the reference signal.

3.2 Phase Detector

There are several types of phase detectors. A plot of the phase difference between the phase detector input signals vs. the output of a phase detector is referred to as the characteristic S-curve of the phase detector, and will in general be either sinusoidal or linear. Sinusoidal S-curves provide a non-linear output to input phase differences, and are not able to track phase differences as far as linear S-curves. Some analog sinusoidal detectors have been modified to produce a longer linear section [3]. PLL loss of lock can occur when the phase difference of the detector's input signals exceeds the operating part of the S-curve. For sinusoidal detectors, the operating range is $[-\frac{\pi}{2}, \frac{\pi}{2}]$; for linear detectors, it can

be more. For maximum tracking ability (i.e. best performance in the presence of noise), PLLs are usually set nominally to operate in the middle of the linear range.

For the phase detector implemented in this receiver, the output is directly proportional to the difference between the two inputs, and is linear over the whole 2π range, providing it with a linear S-curve for non-noisy signals.

In the presence of noise, the phase detector's S-Curve due to the DTL becomes degraded, and it loses its linear characteristic. In Figure 3.2, the S-curve for various SNR ratios are plotted. For low SNR, the curve becomes sinusoidal, which limits the operating range of the phase detector to the area between the points where $\frac{\partial S(\theta)}{\partial \theta} = 0$, i.e. roughly $\pm\frac{\pi}{2}$. For high SNR, the points $\frac{\partial S(\theta)}{\partial \theta} = 0$ encompass almost the entire $\pm\pi$ range. A com-

Figure 3.2: DTL S-Curve with Various SNR

parison between a sinusoidal DPLL and a DTL [12][13] shows that phase variance/jitter performance is comparable for low SNR. Thus the extended linear characteristics give added advantage to the DTL for higher SNR.

3.3 VCO, NCO

The voltage or numerically controlled oscillator (VCO, NCO) is either an analog or digital (respectively) version of the same principle: an oscillator whose output frequency is linearly proportional to an input control voltage.

In the case of the VCO, an analog control signal changes the capacitance in the resonant circuit, therefore changing the frequency of oscillation. With an NCO, a control signal is actually an increment of phase which is added to the nominal phase increment of the NCO, producing a change in the frequency of the output signal. A positive addition of phase causes an increase in frequency, while a negative addition of phase causes a decrease.

3.3.1 Analog (VCO) Phase Increment

The output frequency of the VCO, shown in equation (3.3a), is a function of a nominal (resonant) frequency and a filtered error signal (which modulates the capacitance of the resonant tank). The phase relation is derived (using $\int_0^t x(\lambda) \cdot d\lambda = \frac{1}{s}X(s)$ and $2\pi \int_0^t f_x \cdot dt = \Theta_x(t)$), shown in equation (3.3b), and indicates the presence of an integrator.

$$\begin{aligned}
 f_{VCO} &= f_{NOM} + K_{VCO} \cdot \epsilon_{FILTERED} & (3.3a) \\
 2\pi \cdot \int_0^t f_{VCO} \cdot dt &= 2\pi \cdot \int_0^t f_{NOM} \cdot dt + K_{VCO} \cdot 2\pi \cdot \int_0^t \epsilon_{FILTERED} \cdot dt \\
 \Theta_{VCO}(t) &= \Theta_{NOM}(t) + K_{VCO} \cdot 2\pi \cdot \int_0^t \epsilon_{FILTERED} \cdot dt
 \end{aligned}$$

taking Laplace Transform :

$$\Theta_{VCO}(s) = \Theta_{NOM}(s) + \frac{K_{VCO}}{s} \cdot \epsilon_{FILTERED} \quad (3.3b)$$

where f_{NOM} is the nominal frequency of the VCO, f_{VCO} is the output frequency of the VCO, K_{VCO} is the modulation sensitivity of the VCO, and $\epsilon_{FILTERED}$ is the input error signal.

3.3.2 Digital (NCO) Phase Increment

The output phase of the NCO, shown in equation (3.4a), is a function of the previous phase, the nominal increment of phase (resonant NCO frequency), and an error

signal, which in this case has the units of phase. The frequency relation is derived, shown in equation (3.4b), and can be closely compared to its analog counterpart in equation (3.3a).

$$\Theta_{NCO}(n) = \Theta_{NCO}(n-1) + 2\pi \frac{f_{NOM}}{F_S} + \epsilon_{FILTERED} \quad (3.4a)$$

$$\begin{aligned} \text{since } f &= \frac{1}{2\pi} \frac{\partial \Theta}{\partial t} \\ &= \frac{1}{2\pi} F_S [\Theta(n) - \Theta(n-1)] \\ f_{NCO} &= f_{NOM} + \frac{1}{2\pi} \epsilon_{FILTERED} \cdot F_S \end{aligned} \quad (3.4b)$$

where f_{NCO} is the output frequency of the NCO, f_{NOM} is the nominal frequency of the NCO, and $\epsilon_{FILTERED}$ is the (filtered) phase difference, i.e. $[\Theta_I(z) - \Theta_O(z)]$.

3.4 Loop Filter

The loop filter determines the operation characteristics of the PLL. Loop ‘type’ corresponds to the number of perfect integrators in the loop (or the number of poles at the origin of the open loop transfer function), while loop ‘order’ corresponds to the highest order of s in the loop transfer function denominator. Most analog loops are of type 1 (sometimes modelled to look like type 2), 2^{nd} order. Higher order loops can cause instability, and much care should be taken in the design, especially with the unpredictable variations in analog components and responses. Fortunately in ‘digital land’, responses are more easily predicted and should contain no surprises.

First order PLLs can track phase steps: with one integrator, the steady state phase error for phase steps is 0. Second order loops can track frequency and/or phase steps: with two integrators, the steady state phase error for both phase and frequency steps is 0. Third order loops can track frequency/phase steps and frequency ramps: with three integrators, the steady state phase error for

frequency/phase steps and frequency ramps is 0. All loops are of at least 1st order, due to the perfect integrator inherent in the VCO.

Przedpelski and Gardner [14] introduced two complicated high order loops, 3rd order, type 2, and 3rd order, type 3 (respectively). Przedpelski's loop filter function is of the form:

$$H_f(s) = \frac{1 + \tau_a s}{\tau_b s(1 + \tau_c s)} \quad (3.5)$$

With the filter providing a 2nd perfect integrator, this PLL is able to track all frequency and phase steps. Although the steady state error is not 0 for frequency ramps, this loop filter was chosen in order to be able to track doppler frequencies to a certain extent.

The loop filter is the most important component in determining the PLLs operational parameters, such as noise bandwidth, acquisition time and acquisition range. There are advantages to having both wider and narrower filters. A wider bandwidth creates more (replicate) carrier phase jitter (more noise is passed through the system) and therefore a higher probability of error, however it allows for faster acquisition times and wider locking ranges. A narrower bandwidth will restrict the amount of noise passing through the loop, lowering phase jitter, but increasing acquisition time and decreasing lock-in range.

Several Przedpelskis filters of varying bandwidths were designed in order to take advantage of their different qualities. All filters were designed in the analog (S) domain, and then transformed into the digital (Z) domain by use of the Bilinear Transform.

3.4.1 Design of the DPLL Loop Filter

A PLL is simply a feedback loop (see Figure 3.3), with a transfer function of:

$$\frac{\Theta_o}{\Theta_i} = \frac{1}{1 + A} \quad (3.6)$$

where $A = K_{PHASE} \cdot F(s) \cdot \frac{K_{VCO}}{s}$, K_{PHASE} is the phase detector constant (V/π), K_{VCO} is the modulation sensitivity of the VCO ($\pi/s \cdot V$), and $F(s)$ is the loop filter response.

Stability of a PLL must always be ensured.

From equation (3.6), it can be seen that as $|A| \Rightarrow 1$ and $\arg(A) \Rightarrow -180^\circ$, the loop transfer function tends towards ∞ : this is the characteristic of an unstable loop. By forcing A to never come close to the above limits, stability of the loop can be ensured. By examining A , it can be seen that the total phase shift is due to the

VCO $\frac{1}{s}$ term (90°), and $\arg(F(s))$. Although more integrators in the filter mean better tracking ability (in the phase/frequency steps, frequency ramp sense), each additional integrator produces an additional phase shift of 90° . With one integrator due to the VCO, and one from Przedpeksis filter, we already have a phase shift of 180° . Therefore, the combination of zeros/poles must be realized in order to make the filter transfer function have *less* than 90° phase shift at a gain of $\frac{1}{K_{PHASE} \cdot K_{VCO}}$ (condition of instability).

Figure 3.3: PLL Basic Model

In order to make a digital loop filter, an analog filter needed to be designed with a good corresponding phase margin. In order to do this, the constants needed to be determined:

$K_{PHASE} = 1$: signal phase is based on a 2π scaling to $[-PI, PI]$ (see Appendix A).

The phase detector output is simply the difference between the input signal phase and the LO phase, i.e. no multiplier/constant

$K_{VCO} = F_S$: output of the filter (i.e. filtered phase difference) is directly added to the phase of the nominal NCO frequency at the sampling rate of F_S

Some considerations had to be taken into account in the filter design. Wiener filters can be designed for optimal signal tracking performance in specific situations [9]: the filters are optimized for a single SNR and input stimulus (i.e. phase or frequency step). Carrier synchronizers are able to lock at data-related sidebands at frequencies spaced $\frac{k}{NT}$ from the carrier [3], so loop bandwidth should be small enough to filter these sidebands out.

In the end, four Przedpelski loop filters were designed, of bandwidths 70Hz, 35Hz, 18Hz, and 8Hz. For the analog response of the filter, educated guesses were made as to where the approximate poles and zeros should be situated in order to obtain a suitable phase margin at the ‘cutoff’ frequency of the filter. *MATLAB* was

Figure 3.4: 70Hz Filter Magnitude and Phase Response

used to perfect the response (see Figure 3.4). For a perfect match in converting an analog filter to a digital filter, an extra zero had to be applied in the analog filter

Figure 3.5: Pole and Zero Plot of 70Hz Filter

at $2F_S$ [15]. The bilinear transformation was then performed, and a digital filter was realized. The coefficients of the filter had to be quantized (for use in the digital receiver, see Appendix A) and were done so to 1/128. The magnitude and phase responses of the digital filters were then verified to be in close agreement with their analog equivalents.

Filter stability can be determined by looking at its *pole-zero* plot. Figure 3.5 plots the poles and zeros of the 70Hz filter. It can be seen that the analog filter is stable (all poles in the left hand of the S-plane) and that the corresponding digital filter is also stable (all the poles are inside or on the unit circle). A gain factor is also included in the filter in order to provide unity gain at the maximum phase margin in the PLL, i.e. gain of filter at maximum phase margin is equal to $K_{PHASE} \cdot K_{VCO}$.

Table 3.1 presents the final results of the filter synthesis for a sampling frequency of 24kHz.

Filter	Analog $H(s) = \frac{(s-z_1)(s-z_2)}{(s-p_1)(s-p_2)}$	Quantized Coef. $H(z) = \frac{b_0+b_1z^{-1}+b_2z^{-2}}{a_0+a_1z^{-1}+a_2z^{-2}}$	Digital $H(z) = \frac{\beta(z-z_1)(z-z_2)}{(z-p_1)(z-p_2)}$
1	<p><i>poles & zeros:</i></p> <p>$p_1 = 0$</p> <p>$p_2 = -246 \cdot 2\pi$</p> <p>$z_1 = -20 \cdot 2\pi$</p> <p>$z_2 = -2 \cdot F_S$</p>	<p>$b_0 = 0.0625000$</p> <p>$b_1 = 12.000000$</p> <p>$b_2 = -12.000000$</p> <p>$a_0 = 1.0000000$</p> <p>$a_1 = -1.9375000$</p> <p>$a_2 = 0.9375000$</p>	<p><i>poles & zeros:</i></p> <p>$p_1 = 1.000000$</p> <p>$p_2 = 0.937500$</p> <p>$z_1 = 0.994845$</p> <p>$z_2 = -192.995$</p> <p>$\beta = 2^{-13}$</p>
2	<p><i>poles & zeros:</i></p> <p>$p_1 = 0$</p> <p>$p_2 = -121 \cdot 2\pi$</p> <p>$z_1 = -10 \cdot 2\pi$</p> <p>$z_2 = -2 \cdot F_S$</p>	<p>$b_0 = 0.0312500$</p> <p>$b_1 = 12.000000$</p> <p>$b_2 = -12.000000$</p> <p>$a_0 = 1.0000000$</p> <p>$a_1 = -1.9687500$</p> <p>$a_2 = 0.9687500$</p>	<p><i>poles & zeros:</i></p> <p>$p_1 = 1.000000$</p> <p>$p_2 = 0.968750$</p> <p>$z_1 = 0.997409$</p> <p>$z_2 = -384.997$</p> <p>$\beta = 2^{-15}$</p>
3	<p><i>poles & zeros:</i></p> <p>$p_1 = 0$</p> <p>$p_2 = -60 \cdot 2\pi$</p> <p>$z_1 = -5 \cdot 2\pi$</p> <p>$z_2 = -2 \cdot F_S$</p>	<p>$b_0 = 0.0156250$</p> <p>$b_1 = 12.000000$</p> <p>$b_2 = -12.000000$</p> <p>$a_0 = 1.0000000$</p> <p>$a_1 = -1.9843750$</p> <p>$a_2 = 0.9843750$</p>	<p><i>poles & zeros:</i></p> <p>$p_1 = 1.000000$</p> <p>$p_2 = 0.984375$</p> <p>$z_1 = 0.998701$</p> <p>$z_2 = -768.998$</p> <p>$\beta = 2^{-17}$</p>
4	<p><i>poles & zeros:</i></p> <p>$p_1 = 0$</p> <p>$p_2 = -30 \cdot 2\pi$</p> <p>$z_1 = -2 \cdot 2\pi$</p> <p>$z_2 = -2 \cdot F_S$</p>	<p>$b_0 = 0.0078125$</p> <p>$b_1 = 12.000000$</p> <p>$b_2 = -12.000000$</p> <p>$a_0 = 1.0000000$</p> <p>$a_1 = -1.9921875$</p> <p>$a_2 = 0.9921875$</p>	<p><i>poles & zeros:</i></p> <p>$p_1 = 1.000000$</p> <p>$p_2 = 0.992187$</p> <p>$z_1 = 0.999349$</p> <p>$z_2 = -1536.999$</p> <p>$\beta = 2^{-18}$</p>

Table 3.1: Filter Coefficients, Zeros and Poles

3.5 Z-Domain Model

The digital PLL can be modelled in the Z-Domain, just as an analog PLL can be modelled in the S-Domain. Equations (3.7), (3.8) and (3.9) govern the PLL:

Phase detector:

$$\epsilon(n) = \Theta_i(n) - \Theta_o(n) \quad (3.7)$$

Filter:

$$\begin{aligned} F(n) = & b_0 \cdot \epsilon(n-1) + b_1 \cdot \epsilon(n-2) + b_2 \cdot \epsilon(n-3) \\ & - a_1 \cdot F(n-1) - a_2 \cdot F(n-2) \end{aligned} \quad (3.8)$$

This is expressed in a recursive form, and can be represented by the direct form second-order block diagram in Figure 3.6 [16]. The coefficients are taken from the form: $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$

NCO:

$$\Theta_o(n) = \Theta_o(n-1) + 2\pi \frac{f_Z}{F_S} + \epsilon_{FILTERED} \quad (3.9)$$

where $\epsilon_{FILTERED} = \frac{F(n)}{\beta}$, β is the attenuation factor used to achieve unity gain at the maximum phase margin of the loop.

From these equations, we can derive the loop equations:

$$\begin{aligned} H_C(z) &= \frac{\Theta_o}{\Theta_i} \\ &= \frac{1}{\beta} \cdot \frac{a_1 z^2 + a_2 z + a_3}{z^3 + (a_1/\beta - a_4 - 1)z^2 + (a_2/\beta - a_5 + a_4)z + (a_3/\beta + a_5)} \end{aligned} \quad (3.10)$$

$$\begin{aligned} H_e(z) &= 1 - H_C(z) \\ &= \frac{\epsilon}{\Theta_i} \\ &= \frac{z^3 + (-a_4 - 1)z^2 + (-a_5 + a_4)z + a_5}{z^3 + (a_1/\beta - a_4 - 1)z^2 + (a_2/\beta - a_5 + a_4)z + (a_3/\beta + a_5)} \end{aligned} \quad (3.11)$$

Figure 3.6: IIR Filter Block Diagram

Figure 3.7: Block Diagram of Implemented Digital PLL

The Z-Domain model of the PLL is shown in Figure 3.7.

3.5.1 Stability

To check for unconditional stability, the Jury Test was applied. The Jury Test is the digital equivalent of the Routh-Hurwitz test [17], which look at the denominators of a transfer function to see whether it falls within the constraints of stability. Through these constraints, Osborne [18] [19] derived optimal parameters for specific stimulus inputs for simple 1st order and more complex 2nd and 3rd order loops.

The Jury test is performed on the denominator of equation (3.6): $1+A = F(z) = c_3z^3 + c_2z^2 + c_1z + c_0$. Conditions for stability of our 3rd order transfer function are:

1. $F(1) > 0$
2. $F(-1) < 0$
3. $|c_0| < c_3$
4. $|c_0^2 - c_3^2| > |c_0c_2 - c_1c_3|$
5. $|c_0^4 + 2c_0c_1c_2c_3 - c_1^2c_3^2 + c_3^4 - c_0^2(c_2^2 + 2c_3^2)| > |c_0^3c_1 + c_2c_3^2(c_3 - c_1) - c_0^2c_2(c_1 + c_3) + c_0c_3(c_1^2 + c_2^2 - c_1c_3)|$

In the case of our DPLL Z-Domain model, the coefficients of $F(z)$ reduce to:

$$\begin{aligned} c_3 &= 1 & c_2 &= a_1/\beta - a_4 - 1 \\ c_1 &= a_2/\beta - a_5 + a_4 & c_0 &= a_3/\beta + a_5 \end{aligned}$$

The 5 stability conditions were investigated for each loop filter and root locus analyses were plotted. With each set of loop filter coefficients, the PLL successfully passed the Jury Test.

3.5.2 Noise Bandwidth

The Z-Domain model allows the determination of the noise bandwidth and (consequently) the SNR of the PLL through [20]:

$$\frac{2B_L}{B_i} = \frac{1}{j2\pi} \oint H_C(z)H_C(z^{-1})z^{-1}dz \quad (3.12)$$

where B_i is the bandwidth of the pre-filter to the PLL, and B_L is the one-sided loop noise bandwidth.

This integral is evaluated with the Cauchy Residue Theorem and computed in *MATLAB*. The bandwidth of the filter preceding the PLL is 1200Hz, and the resulting noise bandwidths are listed in Table 3.2.

Filter Number	Residue	2 Sided Noise BW
1	0.016085	19.3Hz
2	0.007925	9.5Hz
3	0.003934	4.7Hz
4	0.001960	2.4Hz

Table 3.2: Noise Bandwidths

The phase error variance of the PLL for an *unmodulated* carrier can be related to the input SNR by [20]:

$$\sigma_{\Theta}^2 = B_L \frac{N_O}{P} \quad (3.13)$$

$$= \frac{1}{(\frac{S}{N})_L} \quad (3.14)$$

where B_L is the equivalent noise bandwidth, P/N_O is the input signal SNR with a noise bandwidth of 1Hz, and $(\frac{S}{N})_L$ is the loop SNR.

From equation (3.13) and Table 3.2, it can be seen that the phase error variance of our PLL decreases by a factor of 2 for each successively narrower filter.

With N-PSK modulation on the input signal to the PLL and jitter due to noise intermodulation from the multipliers, further terms are needed to quantify the jitter variance of the PLL [3]:

$$\sigma_{\Theta}^2 = (B_L \frac{N_O}{P}) L_N(\rho_i) \quad (3.15)$$

where N represents the number of phase ambiguities in the signal, and $L_N(\rho_i)$ is the loop squaring loss, which describes the degradation in the loop SNR due to signal and noise intermodulation with the non-linearity of carrier regeneration [21]. ρ_i is the input signal to noise ratio to the loop, i.e. $\rho_i = \frac{P}{N_O B_i}$:

$$L_2(\rho_i) = 1 + \frac{1}{2\rho_i}$$

$$L_4(\rho_i) = 1 + \frac{9}{\rho_i} + \frac{6}{\rho_i^2} + \frac{3}{2\rho_i^3}$$

Gardner's has a formula similar to equation (3.15), with the difference of an N^2 term which represents the N -fold phase magnification of the PLL. Through the complex mixing present in our PLL, we have (ideally) a simple carrier tracking loop, with no phase magnification.

Furthermore, a sinusoidal (multiplicative) phase detector will generate an output error proportional to N times the phase error between the input signals in a typical N-PSK remodulator/Costas/squaring loop. This reduces the effective range of the phase detector by a factor of N , and results in this type of loop losing lock $10 \log(N^2)$ dB sooner than a simple carrier tracking PLL. Since we are using a differencer phase detector instead of a multiplier, our phase detector output signal is

equal to the phase error between the input signals. As previously mentioned, this gives our phase detector operational limits as $[-\pi, \pi]$, whereas with sinusoidal phase detectors, the operational limits would be $[-\frac{\pi}{N}, \frac{\pi}{N}]$. This choice of architecture allows this PLL to have the same (approximate) performance for BPSK, QPSK or N-PSK.

Figure 3.8: PDF of PLL Frequency Error

Figure 3.8 is a probability density function of the PLLs frequency error. This shows the effects of the different filters, and how each narrower filter is less responsive to noise, producing less frequency error. This run was performed with a QPSK signal at 3dB E_b/N_O .

3.5.3 Stimulus Response

The Z-domain model can also be given stimuli, to determine the loop response. To determine the steady state error of the PLL, one must evaluate:

$$\lim_{z \rightarrow 1} \left[\frac{\varepsilon(z)}{\Theta_i(z)} \cdot \Phi(z) \right] \quad (3.16)$$

where $\Phi(z)$ is the input stimulus to the loop, i.e. a phase step, a frequency step or a frequency ramp.

Table 3.3 provides a summary of the results. The loop was simulated with such inputs, and these results were verified in *MATLAB*.

Stimulus Type	Time Domain $\phi(t)$	Z-Transform $\Phi(z)$	$\lim_{z \rightarrow 1} \left[\frac{\varepsilon(z)}{\Theta_i(z)} \cdot \Phi(z) \right]$
Phase Step	$\Delta\theta \cdot u(t)$	$\frac{\Delta\theta z}{z-1}$	0
Frequency Step	$\Omega t u(t)$	$\frac{\Omega z}{(z-1)^2}$	0
Frequency Ramp	$\frac{\Omega}{2} t^2 u(t)$	$\frac{\Omega}{2} \frac{z(z+1)}{(z-1)^3}$	$\frac{\Omega\beta}{4} \frac{(14-10BmA_4-6BmA_5)}{(BmA_1+BmA_2+BmA_3)}$

Table 3.3: PLL Steady State Response to an Input

The frequency ramp stimulus will cause the PLL to lose lock when $\lim_{z \rightarrow 1} [\cdot] > PI$. This enables the loop to theoretically track (in the widest filter) a frequency ramp of 4.096kHz/s, which is more than adequate for any type of application.

3.6 Implemented Algorithm

The actual implemented algorithm of the PLL is shown below:

for each sample:

```

xy3(n) = xy2(n-1);                                % update buffer values
xy2(n) = xy1(n-1);
xy1(n) = ε(n-1);
F(n) = [xy1(n), xy2(n), xy3(n), xy4(n-1), xy5(n-1)] *
        [b0, b1, b2, -a1, -a2];
xy5(n) = xy4(n-1);
xy4(n) = F(n) / β;
fNCO(n) = fZ + 8·F(n) / β;1
LOPHASE(n) = LOPHASE(n-1) + fNCO(n)/8;
ε(n) = RFPHASE(n) - LOPHASE(n);

```

where xy_{1-5} is a buffer, containing the elements to be processed by the IIR filter, f_{NCO} is the instantaneous frequency of the NCO, f_Z is the nominal NCO frequency, RF_{PHASE} is the phase of the input signal (Θ_i), LO_{PHASE} is the computed output phase of the NCO (Θ_o), ϵ is the error signal between those two signals ($\Theta_i - \Theta_o$), F is the output of the filter, β is the scaling factor for unity gain at the maximum phase margin and $[b_0, b_1, b_2, -a_1, -a_2]$ are the filter coefficients.

¹A phase increment is $\Delta\theta = 2\pi ft$. If $f = f_{NCO}$, and $t = T_S$ (for digital systems), $\Delta\theta = 2\pi \frac{f_{NCO}}{F_S}$. With the scalings from Appendix A, $\Delta\theta = f_{NCO}/8$, so knowing the instantaneous frequency allows the computation of the required phase increment. Conversely, the instantaneous frequency can be calculated from the known phase increment.

Chapter 4

Remodulator

One of the methods of demodulating a signal is through the use of a remodulator (see Figure 4.1). A modulated signal is made up of a combination of two signals: a data signal and a carrier signal. A remodulator is based on a double feedback loop: a loop for data retrieval and a loop for carrier retrieval.

The carrier retrieval loop performs a complex mix between the recovered data and the incoming signal in order to generate a pure carrier. This carrier is then used to perform a complex mix with the incoming signal to recover the data. Although this is similar to a ‘chicken or the egg’ (which comes first) scenario, the proper use of delays and the fact that the carrier is not a stationary signal allows for this architecture to work.

Figure 4.1: Block Diagram of BPSK Remodulator

The remodulator is similar to the Costas loop, however it generates a pure carrier component, and is usually implemented at lower frequencies than the Costas loop. The squaring (or $(\cdot)^N$) loop requires a divide by 2 (or N) which adds a extra

complexity in the circuit, as well as a PLL which operates at that multiple of the carrier frequency.

4.1 BPSK Remodulator

The remodulator shown in Figure 4.1 is the basic form of the remodulator, and is used for demodulation of BPSK signals.

4.1.1 BPSK Components

FIR 1 LPF: This first filter serves to eliminate interference outside of the data channel. As the carrier must be below $\frac{F_s}{16}$, the maximum frequency the data can spread to is $\frac{F_s}{16} + F_b$ Hz. The signal should not be filtered before this frequency, so a conservative low-pass bandwidth of 4800Hz was chosen. An important note is that all digital filters have a processing delay of half its number of taps. More taps equals a smoother response. FIR filters were chosen over IIR filters for predictable phase delay. This filter was implemented as a 20 tap FIR raised cosine filter of $\alpha = 0.35$, 4800Hz roll-off frequency.

Data Wipeoff: As shown in the phase detector section, the data can be ‘wiped’ off from the incoming signals $I_S = d_1 \cos \Theta$, $Q_S = d_1 \sin \Theta$:

$$\begin{aligned} A &= I_S \times d_1 \\ &= d_1^2 \cdot \cos \Theta \\ B &= Q_S \times d_1 \\ &= d_1^2 \cdot \sin \Theta \end{aligned}$$

Then $\arctan(B/A)$ gives the carrier phase. Multiplying by d_1 eliminates the phase reversal of the data (i.e. $(\pm 1)^2 = 1$) and ‘wipes’ off the data.

FIR 2 LPF: This second filter serves to eliminate signals and noise past the maximum carrier frequency ($\frac{F_s}{16}$). The carrier is the only piece of useful information at this point, everything else is just noise. This filter was implemented as a 20 tap FIR raised cosine filter of $\alpha = 0.35$, 1200Hz roll-off frequency.

ATAN: This block computes the carrier phase between the two branches of incoming signals through the *arctan* function, and the output phase is fed into the PLL.

PLL: Tracks the phase of the carrier, and produces a replicate of the output carrier phase for coherent demodulation. This replicate phase is in the form of a $\text{mod}(2\pi)$ accumulator, and is used to generate the LO signals.

Sin/Cos: This block converts the output phase from the PLL into quadrature LO signals.

Carrier Wipe-off: As in the case of data wipe-off, carrier wipe-off is also needed in order to down-convert the data to 0Hz. The same type of equations as in (3.1) and (3.2) are used:

$$\begin{aligned} A &= I_S \times \cos \theta + Q_S \times \sin \theta \\ &= d_1 \cos^2 \theta + d_1 \sin^2 \theta \\ &= d_1 \end{aligned}$$

The $\cos \theta$ and $\sin \theta$ are the LO signals obtained from the PLL.

FIR RRC: The transmitted signal was filtered with an RRC filter, and so for minimum inter-symbol interference, the received data must also be filtered with an RRC filter. This filter was implemented as a 20 tap FIR RRC filter of $\alpha = 0.35$, 2400Hz roll-off frequency.

Limiter: This renders the filtered data to purely digital data, either 1s or 0s.

Delay: In order for data wipe-off to occur, the synchronous data must be known. The first branch (data wipe-off) has a delay equal to half the number of taps of the first FIR filter. The second branch (carrier wipe-off) has a delay equal to half the number of taps of the FIR RRC filter. When these delays are equal, the demodulated data is presented to the data wipe-off component at

the correct time. If the filter delays are not equal, an extra delay must be present in order to make them equal.

Although the remodulator architecture allows for data retrieval, with N-PSK modulation, there is N times ambiguity as to the polarity of the received data, the data could be locked on any of the $\frac{360}{N}$ phases. A data preamble, a predetermined expected pattern, is usually included in order to determine the correct polarity. Bit synchronization is also needed in order to sample the data at the points of minimum inter-symbol interference.

In the case of the CDGPS receiver, a synchronizing code was imbedded at known locations in the data, i.e. every 20 data bits. As well as providing polarity information, this code also allows potential phase inversions (PLL cycle slipping) to be caught. With a buffer of recently received data, the data can be re-inverted to its original state from the status of the recently received fast synch code bits.

4.2 QPSK, OQPSK Remodulator

The QPSK remodulator is similar to the BPSK remodulator, except the carrier and data ‘wipe-off’ operations are more complex. The feedback data (for the data wipe-off) requires a limiter [4] to produce non-linearity. The QPSK remodulator is shown in Figure 4.2.

Figure 4.2: Block Diagram of QPSK Remodulator

The OQPSK remodulator architecture is essentially the same as the QPSK remodulator, with the difference that the bit transitions of the two BPSK data streams occur at staggered times.

As shown in the phase detector section, the data wipe-off is performed as in equations (3.1) and (3.2). The carrier wipe-off is performed as follows:

$$\begin{aligned}
A &= I_S \times \cos \theta + Q_S \times \sin \theta \\
&= d_1(\cos^2 \theta + \sin^2 \theta) + d_2(\cos \theta \sin \theta - \cos \theta \sin \theta) \\
&= d_1 \\
B &= I_S \times \sin \theta - Q_S \times \cos \theta \\
&= d_2(\cos^2 \theta + \sin^2 \theta) + d_1(\cos \theta \sin \theta - \cos \theta \sin \theta) \\
&= d_2
\end{aligned}$$

4.3 Remodulator Algorithms

4.3.1 Search Algorithm

The incoming signal is downconverted to a frequency somewhere between $[0, \frac{F_s}{16}]$ Hz by the receiver hardware. Due to timing errors (crystal output) and frequency offsets (Doppler shift), it is not known exactly where in that range the signal is. If the PLL was left running, the loop would eventually lock onto the signal, however it might take a long time. To speed up this process, an algorithm was determined to find the right frequency bucket.

A frequency range was divided into N buckets (odd number), and a histogram was set up, one slot for each bucket (see Figure 4.3, where f_C is the center of the range, and $\pm f_L$ are the limits of the frequency range). The PLL's nominal

Figure 4.3: Histogram Frequency Bucket Setup

frequency of the NCO (f_Z) was then set to the middle of the first bucket, and the PLL was allowed to run. After a period of time, the bucket in which the NCO *output* frequency falls into gets an increment. This is repeated for the same bucket M times, and then the nominal frequency of the NCO is increased to the center of the next bucket. The PLL is allowed to run again, and then after some time, the appropriate bucket gets incremented. This process is repeated for all N buckets. The biggest bucket is the one where the carrier frequency is found; this works due to the tendency of the PLL to inch towards the correct frequency. This is much

quicker than just letting the PLL run uncontrolled for two main reasons. If there are other interfering signals that the PLL could lock onto, a free running PLL could lock onto a ‘wrong’ signal and just stay there, whereas with this technique, all possible signal locations can be found and tested. Also, as mentioned previously, a large frequency offset could take the PLL a considerable amount of time to lock onto.

A sample log of the NCOs instantaneous frequencies for a 0dB QPSK signal from this algorithm is shown in Figure 4.4, and the actual bucket histogram is shown in Figure 4.5.

Figure 4.4: Histogram Frequency Sweep

Figure 4.5: Histogram Results

When analyzing the histogram, the bucket with the most entries is selected, and the PLL is set up to have its nominal NCO frequency in that bucket. If this was not the right bucket (interfering signal), the histogram has information on the next likely case, and can set up the nominal NCO frequency to be in that bucket.

The center frequency, f_C , the range of frequencies, $\pm f_L$, the number of buckets, N , the number of trials per bucket, M , and the length of an individual bucket trial, X samples, should all be predetermined. The length of time this algorithm will take is simply $\frac{N \times M \times X}{F_S}$ seconds. As will be seen with the locking results (Chapter

7: Simulations), BPSK and QPSK have different mean time to lock results, and X should be adjusted accordingly.

4.3.2 Locking Algorithm

When a PLL is in lock, the NCO is ‘locked’ onto a frequency, and tracks it through an error signal. A good indicator of whether the loop is locked is by how much the NCO frequency is changing. Ideally, the frequency would not deviate at all in lock. Unfortunately, noise present on the error signal causes the frequency to vary, and this is referred to VCO/NCO jitter/phase noise. Unlike analog devices, it is theoretically possible to obtain a digital frequency output with no phase noise. As with all analog oscillators, the NCO control signal is corrupted by noise, however quantization can reduce the effect of this noise. Another factor that could cause a deviation in the frequency output is frequency drift or acceleration, due to changes in crystal temperature, Doppler shifts, etc. Most of these non-noise related factors occur relatively slowly, and do not have a large effect on the proposed method of locking.

One technique for establishing lock in typical sinusoidal phase detector PLLs is taking the cosine of the error signal coming out of the phase detector [22]. This is a valid method when the steady state error is 0. With our PLL, the steady state phase error is zero for both a phase and frequency step, but a frequency ramp provides a non-zero steady state error. If the stimulus input were an accelerating frequency, the loop would still be in lock, but this cosine detector would fail.

Variance plays a key role in the determination of the PLL lock status. In a simulation environment, where memory is not a premium, the following scheme could be used: a circular buffer of length N is filled with a new estimate of the instantaneous frequency for every sample that is computed (see Figure 4.6). This buffer holds the last N estimates of the NCO frequency, and computing the variance of the buffer is an estimate of how good/close the lock is. N should be large enough to provide a good ‘average’ over a period of time. This

method is computationally expensive every time the variance needs to be calculated.

Figure 4.6: Circular Buffer

As is the case with chips with a limited amount of memory, this buffer of length N is most likely too extravagant. A second method of computing a reasonable estimate of the variance is by taking instantaneous variances of input data and averaging them, as opposed to taking the variance of a buildup of input data. This requires only one memory location for the frequency (no buffer), and it spreads the computations over every sample, as opposed to a blitz every N samples.

Samples are clocked into the receiver at the sampling rate. After the first M samples, the frequency mean (f_{PREV}) is calculated from the instantaneous NCO frequencies (f_{NCO}). This mean frequency is used to compute the instantaneous variance (between f_{PREV} and f_{NCO}) for the next M samples. These instantaneous variances are averaged for M samples, and a new average frequency is computed, and used for the variance computation of the next M samples.

Since we know we are averaging over M samples, we can simply accumulate the instantaneous frequencies and variances, and then divide by M at the end. If M is a multiple of 2, shifting can be performed (instead of division) to save processing time.

The locking algorithm implemented is as follows:

for each sample:

```

counter = counter + 1;           % average over M samples
fnow = fnow + fnco;             % accumulating estimates
varf = varf + (fnco-fprev)^2;
if (counter >= M)
    counter = 0;
    fnow = fnow/M;              % average fnow
    varf = varf/M;              % average varf
    varf = round(varf* $\gamma$ );    % normalize varf1
    EVALUATE varf                % determine if locked
    fprev = fnow;                % new estimate for fprev
    varf = 0;                    % reset averages
    fnow = 0;
Endif

```

Once the average variance is calculated at M , it is compared against 2 levels, VARFTHRESHHI and VARFTHRESHLO (see Figure 4.7). If the variance is below VARFTHRESHLO, the loop can be considered to be in lock for the last M samples; if it is above VARFTHRESHHI, the loop is not considered in lock. Due to the random nature of noise, and the possibility of excess noise occurring for M samples, the loop should not come out of lock for a single bad average of the variance. Instead, a counter is used to determine whether it is in lock or not. Each time the evaluated variance is below the VARFTHRESHLO, the counter gets incremented, and each time

¹variance normalization is explained in Section 4.3.3: Adaptive Filtering Algorithm

the variance is above the `VARFTHRESHHI`, the counter gets decremented. When the counter reaches a high limit, the loop is deemed to be in lock. If the loop is in lock, it can come out of lock when the counter reaches 0. This will be further explained in the adaptive filtering section, where the adaptive filtering is closely related to the locking procedure algorithm.

Figure 4.7: Variance Thresholds

This counter allows the loop to stay in lock with short bursts of noise or a change of frequency. For frequency drifting, a maximum drift rate exists (see Appendix (C.3) and (C.4)), below which the loop will remain in lock (excluding noise considerations).

A sample run of this algorithm is shown in the next section, Figure 4.8.

4.3.3 Adaptive Filtering Algorithm

Performance of a phase locked loop is related to the loop filter. As mentioned previously, the loop filter affects the acquisition time, the lock range and the amount of phase jitter present on the oscillator. The use of a variety of filters can increase the performance of the phase locked loop [23]. For instance, a wide bandwidth loop filter would be able to lock to a signal with a greater frequency offset much faster than a narrow bandwidth loop filter. On the other hand, a narrow bandwidth loop filter will filter out most the noise in the loop, allowing for a much smoother NCO frequency output. On this basis, several loop filters were designed.

With a narrower filter, the amount that the PLLs oscillation frequency can differ from the ‘average’ frequency (calculated in the locking algorithm) is smaller than with a wider filter. This creates unequal frequency variances between filters. In order to overcome this (to keep everything normalized to one variance), a scaling factor was determined by allowing the loop to lock on the same generated signal with different filters. The average frequency variance of a locked signal for each filter was calculated, as well as the mean frequency offset, and the ratios were tabulated in Table 4.1. The ratios between filters are constant, however the absolute value of the variance varies with input SNR and type of modulation. The `VARFTHRESHHI` and `VARFTHRESHLO` limits were then set relative to the absolute values of low QPSK and BPSK SNR. With high SNR, the loop’s absolute variance would be smaller, but we want to be able to lock onto the lower SNR signals.

The widest filter is used for initial acquisition, and the filters narrow if warranted. If the loop comes out of lock, the filter resets itself to the widest filter for quick reacquisition. A second option is for the (just) unlocked loop to remain with the narrowest filter (in case the signal was blocked only for enough samples for the

Filter Number	$\overline{f_{OFFSET}}$	$\overline{var}$	Ratio (2^n)
1	0.2	19473.9	8.7
2	0.2	1781.3	5.2
3	0.0	637.5	3.7
4	0.0	48.1	0.0

Table 4.1: Locking Variances due to Filters

loop to come out of lock). If the signal reappears, ideally the frequency should still be centered in the narrowest filter's bandwidth.

Three counters are used for switching between these filters: *filterUp*, *filterDown*, *loseLock*. Each of these are used in determining whether the loop filter should become wider or narrower, and if the PLL is 'in lock' or should come out of lock. These counters work on the same principle as the locking mechanism: *filterUp* increments if the variance is lower than FVARTHRESHLO, *filterDown* and *loseLock* increment if the variance goes higher than FVARTHRESHHI. Each counter has its own limit, and once it is reached, the counter is reset, and the appropriate action is taken place.

filterUp: serves first to determine if the loop is in lock, and then to narrow the bandwidth of the loop

filterDown: serves to widen the bandwidth of the loop and to indicate loss of lock if widening is not possible (tries to step down from the widest filter)

loseLock: serves to indicate loss of lock (a faster indicator than filterDown). *loseLock* takes into account the complete loss of a signal, whereas *filterDown* will eventually lose lock when trying to step out of the widest filter

Upon achieving lock and changing the filter bandwidth, the nominal frequency of the NCO (f_Z) is updated to the new frequency average of the PLL in order to keep the loop operating in the center of its bandwidth. The PLL loop filter taps are also zeroed.

Figure 4.8: Locking and Adaptive Filtering Algorithm Sample Run

Shown in Figure 4.8 is a sample run of the locking process with a BPSK modulated carrier of $0\text{dB } E_b/N_0$ at 20Hz offset from the nominal NCO frequency. Variances are computed every $M = 32$ samples. At sample 320, the *filterUp* counter has reached its limit, indicating that the loop should now be in lock. At samples 640, 864, and 1120 the filter narrows down successively to the smallest bandwidth. The various frequencies can be seen from the figure as well, such as the NCO nominal frequency f_Z , the instantaneous frequency f_{NCO} , and the ‘average’ frequency used for variance calculations f_{PREV} .

Chapter 5

Bit Synchronization

Once the PLL has achieved lock, bit synchronization needs to be established, otherwise the data could be read at the wrong instant, i.e. at a point of high ISI. In this realization, the sampling frequency is at a higher rate than the data frequency, so each data bit is represented by several samples. A new method has been implemented in which the correct sampling instant shall be chosen.

A stream of samples comes from the output of the demodulator. For each changing bit of data, the output must flip between ± 1 , i.e. the output must cross zero. This information can be used in order to determine the most likely sampling instant for the data. Knowing where the zero crossings occur provides the bit transition area, and by knowing this, the point of lowest ISI can be found.

This bit synchronization algorithm is based on the accumulation of 3 separate histograms, each composing of N bins, where N is the number of samples per bit. The flow diagram of the bit synchronizer is shown in Figure 5.2.

Histogram #1: ZBUCK(1:N) N element array which gets incremented for every zero crossing, and is reset once a certain limit, ZBUCKMAX, is reached.

Histogram #2: ZACC(1:N) N element array which gets incremented when ZBUCK overflows. Whichever ZBUCK bin overflows is the same bucket that gets incremented in ZACC. Once a bucket has reached a limit, ZACCMAX, bit synchronization stage 1 is reached, and that ZACC bucket is reduced by 2 while all the others are reduced by 1. This is a nonlinear average, which relies on the fact that the probability (including noise) of the right bucket being chosen is at least twice that of the other buckets. This is dependant upon N .

Histogram #3: ZTOT(1:N) N element array which is an accumulation of ZBUCK. It is an ongoing average which is reset after a certain period of time. Bit synchronization stage 2 is reached if the maximum bucket of ZTOT is the same as the doubly decremented ZACC bucket.

For the bit synchronizer to return a valid sampling instant, two things must happen: the same ZACC bucket must ‘overflow’ ZACCMAX times (stage 1 bit synchronization), and upon evaluation (after COUNT iterations), ZTOT must have its maximum slot as the same bucket index as ZACC. If they are different, no bit synchronization occurs, but if they are the same, stage 2 bit synchronization (full synchronization) is deemed to have been found. Due to the double penalty for the winner, only the correct bucket should be chosen.

A sample bit synchronization run for a demodulated 0dB E_b/N_O , $F_B = \frac{F_s}{5}$ BPSK signal is shown in Figure 5.1. At sample 3051, the ZBUCKs have overflowed COUNT times, and so ZTOT is evaluated. Bucket 1 has the highest count, so that is tentatively labeled the zero crossing bucket. ZACC

Sample		Bucket 1	Bucket 2	Bucket 3	Bucket 4	Bucket 5
221	Zacc	1	0	0	0	0
386	Zacc	2	0	0	0	0
577	Zacc	2	1	0	0	0
806	Zacc	3	1	0	0	0
1002	Zacc	3	2	0	0	0
1204	Zacc	3	2	0	1	0
1360	Zacc	3	2	0	1	1
1572	Zacc	3	3	0	1	1
1720	Zacc	3	3	0	1	2
1956	Zacc	4	3	0	1	2
2147	Zacc	4	4	0	1	2
2366	Zacc	5	4	0	1	2
2567	Zacc	5	5	0	1	2
2740	Zacc	5	5	0	1	3
2896	Zacc	6	5	0	1	3
3051	Zacc	7	5	0	1	3
	Ztot	122	121	72	101	118
		center = 122, leftsum = 219, rightsum = 193				
3270	Zacc	7	5	0	1	4
3451	Zacc	8	5	0	1	4
3585	Zacc	8	5	0	1	5
3715	Zacc	8	5	0	1	6
3932	Zacc	8	6	0	1	6
4092	Zacc	8	7	0	1	6
4296	Zacc	9	7	0	1	6
4436	Zacc	10	7	0	1	6
		ZACC HITS ZACCMAX BITSYNC STAGE 1 ACHIEVED				
4676	Zacc	9	6	0	0	5
4820	Zacc	9	6	0	0	6
5035	Zacc	9	6	0	0	7
5166	Zacc	10	6	0	0	7
		ZACC HITS ZACCMAX				
5449	Zacc	8	5	0	1	6
5649	Zacc	8	5	0	2	6
5786	Zacc	9	5	0	2	6
5972	Zacc	9	6	0	2	6
	Ztot	125	101	86	88	120
		center = 125, leftsum = 208, rightsum = 187 BITSYNC STAGE 2 ACHIEVED				
6121	Zacc	10	6	0	2	6
		ZACC HITS ZACCMAX				
6276	Zacc	9	5	0	1	5
6421	Zacc	10	5	0	1	5
		ZACC HITS ZACCMAX				
6596	Zacc	9	4	0	0	4
6736	Zacc	10	4	0	0	4
		ZACC HITS ZACCMAX				
6896	Zacc	9	3	0	0	3
7041	Zacc	10	3	0	0	3
		ZACC HITS ZACCMAX				
7196	Zacc	9	2	0	0	2
7351	Zacc	10	2	0	0	2
		ZACC HITS ZACCMAX				
7562	Zacc	8	2	0	0	1
7565	Zacc	8	2	0	0	2
7720	Zacc	8	2	0	0	3
7936	Zacc	9	2	0	0	3
8175	Zacc	9	2	0	0	4
8372	Zacc	9	3	0	0	4
8572	Zacc	9	4	0	0	4
8574	Zacc	9	4	0	1	4
8745	Zacc	9	4	0	1	5
	Ztot	137	97	76	80	115
		center = 137, leftsum = 195, rightsum = 173 BITSYNC STILL VALID				
8918	Zacc	9	4	1	1	5
9070	Zacc	9	4	1	1	6
9176	Zacc	10	4	1	1	6
		ZACC HITS ZACCMAX				
9440	Zacc	8	3	0	0	6
9632	Zacc	8	4	0	0	6
9801	Zacc	9	4	0	0	6

Figure 5.1: Bit Sync Table

must overflow first, so nothing happens until sample 4436. Here, the overflowing ZACC is decremented by 2, and all others are decremented by 1. This represents bit synchronization stage 1. At sample 5166, ZACC hits the overflow again, and is decremented again. At sample 5972, ZTOT is again evaluated, and the highest ZTOT matches the overflown ZACC, which is the criteria for bit synchronization stage 2, so full bit synchronization occurs. ZACC continues to overflow, and ZTOT is evaluated again at sample 8745, where bit sync is proved to be still valid.

Once bit synchronization has been determined, the histograms also provide the advantage of timing synchronization. If the ZTOT buckets distribution seem to be drifting one way (i.e. going from one dominant bucket to two dominant (adjacent) buckets), it can be an indication that the clock timing is off.

In a noiseless system, one bucket would always win. However, noise can corrupt the recovered data, creating false zero crossings. These might be characterized by a zero crossing, and then a 2^{nd} zero crossing, coming from an immediate reversal (shot noise). These can be removed in a simple algorithm which cancels adjacent samples' successive zero crossings. The bit synchronizer for QPSK signals (2 binary baseband signals) works in the same way, except that the zero crossings accumulation happens at twice the speed.

Figure 5.2: Bit Synchronization Logic Flow

Chapter 6

Simulations

6.1 Signal Generation

For the testing of the digital receiver, synthesis of the output signal of the receiver hardware is required. Depending on the performance and accuracy of the receiver hardware, the downconverted signal could be at any frequency between $[0, \frac{F_s}{16}]$ with any SNR. In a world without noise and interference, the downconverted signals take the form of equations (2.1) or (2.2). However, as the world is not perfect, noise is present everywhere. Therefore with any signal generated, noise must be modelled upon it.

In order to provide the proper SNR (Signal to Noise Ratio), scaling between the signal component and the noise is required.

$$SNR(dB) = 10 \log_{10} \frac{var(signal)}{var(noise)}$$

Variance is a ‘power’ estimation of both signal and noise strength. It is a measure of the deviance of a signal from the mean, and as the mean for both the signal and noise (no dc terms) is 0, the bigger the deviance, the bigger the signal.

Figure 6.1 shows how the noisy data signal will be generated.

6.1.1 Noise Generation

For simulation purposes, it is important to be able to model noise correctly, otherwise results may be biased. When dealing with down-converting signals in quadrature, noise generation becomes complicated.

To create noise on a carrier frequency, the complete noise spectrum must be uncorrelated. The simplest way of achieving noise about a carrier is to multiply the carrier by baseband noise. However, when mixing signals, the baseband frequency

Figure 6.1: Generation of a BPSK Signal

spectrum is imposed symmetrically about the up-conversion frequency, producing a correlated spectrum. Therefore a method is needed to ensure the noise is not mirrored about the carrier frequency.

High frequency noise can be represented by:

$$n_1(t) \cdot \cos(\omega_C t) + n_2(t) \cdot \sin(\omega_C t) \quad (6.1)$$

where $n_1(t)$, $n_2(t)$ are low frequency random processes which are stationary and statistically independent, having a null mean value and uniform spectral density. It is assumed that a bandpass filter is used in the receiver, limiting the noise to $\pm B_i$ Hz.

Equation (6.1) is slightly misleading, as a normal multiplication of the $n_1(t)$ and $n_2(t)$ signals about the *sin* and *cos* carriers would provide a symmetrical frequency response about the carrier, ω_C . In this equation, $n_1(t)$ and $n_2(t)$ are representations of non-correlated noise between $[\omega_C - B_i, \omega_C + B_i]$ Hz.

The theory behind setting up the noise is to create wideband noise around 0Hz, filter it to B_i Hz (see Figure 6.2), and then shift it up to the appropriate frequency. A complex mix is needed to ensure that the noise signal is not mirrored about the carrier frequency.

Figure 6.2: Filtered Noise

The downconverted (by $\sin(\omega_{LO} + \theta)$ and $\cos(\omega_{LO} + \theta)$) quadrature noise signals can be represented as:

$$I_{NOISE} = -\frac{1}{2}n_1(t) \cdot \sin(\phi) + \frac{1}{2}n_2(t) \cdot \cos(\phi) \quad (6.2a)$$

$$Q_{NOISE} = \frac{1}{2}n_1(t) \cdot \cos(\phi) + \frac{1}{2}n_2(t) \cdot \sin(\phi) \quad (6.2b)$$

where ϕ is $\omega_{RR}t + \theta$, where $\omega_{RR} = (\omega_C - \omega_D)$.

$n_1(t)$ can be modelled as two noise signals, $n_A(t)$ and $n_B(t)$. These signals allow for a completely random frequency spectrum of noise from $[-B_i, B_i]$ Hz. To generate the above signals, a complex mix is performed with both noise signals ($n_A(t), n_B(t)$) to create the $n_1(t)$ spectrum (see Figure 6.3¹), and the same is done with a second set of $n_C(t)$ and $n_D(t)$ signals to form $n_2(t)$ (see Appendix B for the complete noise equations).

The synthesis of noise was aimed to produce a noise variance equal to 1. When the noise vector is synthesized by *MATLAB*, it has a wideband frequency spectrum

¹It should be noted that a *hilbert* transform imparts a phase shift of 90° on its argument, i.e. $\text{hilbert}(\sin) = -\cos$, $\text{hilbert}(\cos) = \sin$.

and a variance of 1. The noise is filtered to narrowband ($\pm B_i$), but filtering changes the variance, so a factor must be computed in order to return the variance to its initial value. This was determined to be $\sum_{n=1}^N \overline{H_F^2(w)}$, where $H_F(w)$ is the frequency response of the filter (see Appendix (C.2)).

Figure 6.3: Up-Conversion of Noise

6.1.2 Data Generation

Unlike noise, data is symmetrically modulated onto the carrier. A binary (± 1) impulse stream is created, containing either random or patterned data. This impulse data stream is then filtered by a root raised cosine filter (see Figure 6.4) and is subjected to the same scaling factor as the noise to render it to a unity variance signal. A further scaling factor of $sigratio = 10^{SNRdb/20}$ provides the signal with the correct amplitude compared to the noise.

Figure 6.4: RRC Filtered Data Signal

Inter-Symbol Interference

In the transmission of data signals, pulse spreading occurs due to channel elements that introduce reactances. This pulse spreading can lead to inter-symbol interference, and can corrupt an entire signal if not dealt with. Two methods of fixing this problem are to slow down the transmission bit rate, to a level where pulse spreading becomes a non-factor, or to implement a filter system that results in having no adjacent (interfering) symbol energy at the sampling point of a symbol. These filters are known as Nyquist filters, and are characterized by their symmetrical frequency response about a frequency equal to $0.5 \times \frac{1}{T_S}$, where T_S is the period of a symbol.

The most commonly used Nyquist filters are the raised-cosine and root-raised-

cosine filters. The raised-cosine filter provides the total response needed for ISI, while the root-raised-cosine filters are used in pairs to form a raised-cosine filter response. One of the root-raised-cosine filters is usually found in the transmitter, and the other in the receiver.

For this receiver/transmitter combination, $\alpha = 0.35$ was chosen. α is known as the ‘extra’ bandwidth of the signal and is a tradeoff between bandwidth efficiency and a simpler filter with less sensitivity to timing accuracy [24].

Relationship between BPSK, QPSK, OQPSK

Communication efficiency is usually related to E_b/N_O , which is the bit energy per noise power spectral density. This is related to the input SNR of a data stream by [25]:

$$\begin{aligned}\frac{S}{N} &= \frac{P_S}{N_O} \frac{1}{B_i} \\ &= \frac{E_b}{N_O} \frac{F_b}{B_i}\end{aligned}\tag{6.3}$$

where P_S is the power of the signal, N_O is the power of the noise in a 1Hz bandwidth, B_i is the signal bandwidth and F_b is the bit rate.

In BPSK, every symbol received represents 1 bit. For QPSK and OQPSK, every symbol received represents 2 bits. Therefore, for the same signal SNR, the $(\frac{E_b}{N_O})_{QPSK}$ will be half that of $(\frac{E_b}{N_O})_{BPSK}$. For the same SNR, the QPSK scaling factor needs to be reduced by a factor of $\sqrt{2}$ (amplitude, not power) due to the presence of 2 BPSK signals, therefore, it is *sigratio*/ $\sqrt{2}$.

Quantization

Quantization noise occurs when a signal is not represented fully due to lack of precision of samples (i.e. when transformed from the analog to digital world). The maximum SNR that can be achieved from a limited number of n bits is SNR_Q , which is equal to $3 \cdot 2^{2n-1}$ [24]. This is considered as a secondary noise source to the signal.

6.2 Simulation Results

To validate the performance of the receiver, many criteria were looked at, including the relative performance of BER between BPSK, QPSK and OQPSK, and the effects of quantization and pattern sensitivity. Actual obtained results from the realized version of the receiver are also included.

BER simulations were generally run for SNR between 0 and 8dB. Below ≈ 0 dB, the receiver achieved/lost lock intermittently. Above 8dB SNR (i.e. $BER \approx 2 \cdot 10^{-4}$), on the order of 10^6 samples need to be processed¹, and the receiver was assumed to have no problems. When a carrier offset is mentioned, it means offset from the PLL nominal frequency (f_Z).

Lock acquisition was also investigated for the same criteria as above. Locking and false locking percentages, and mean time to lock were plotted for various signals. It should be noted that an upper limit of 2000 samples was allowed for the receiver to achieve lock. With a longer running time, the locking percentage would increase.

This system was tested on unencoded data, to test the practical bounds of the remodulator. Encoded data would serve to reduce the probability of bit error at the cost of increasing bandwidth.

More BER results can be found in Appendix D.

¹if $F_b = 2400$ bits/s, at least $10 \times (\frac{1}{BER})$ bits need to be received, which is $\frac{10}{BER} \cdot \frac{F_s}{F_b}$ samples

6.2.1 Receiver Performance

Receiver performance of the 3 types of modulation (BPSK, QPSK, OQPSK) was investigated.

Locking

Figure 6.5 is a plot of the locking probabilities for different types of modulation at a 20Hz carrier offset, and Figure 6.6 is a plot of the locking probabilities for varying carrier offsets for a QPSK signal.

Figure 6.5: Locking % vs. Modulation Schemes for a 20Hz Carrier Offset

These curves all share the same shape, however from inspection, it looks as though the QPSK and OQPSK signals are more likely to lock than the plain carrier and BPSK. This can be attributed to the variance thresholds for lock detection being set to the same values for each type of modulation. With lower thresholds, the incoming signal would have a harder time achieving lock since the variance threshold is stricter. In the case of BPSK and the carrier, the limits were too strict, and this point is reinforced by looking at the number of samples the remodulator will take (on average) to lock for these signals, as compared to QPSK, OQPSK (see

Figure 6.6: Locking % vs. Frequency for a QPSK Signal

Figure 6.7). Figure 6.8 is a plot of the false locking percentage for different types of modulation at a carrier offset of 20Hz.

As a characteristic of all PLLs, a carrier is more likely to lock within a given period of time with a decreasing carrier offset[2].

Figure 6.7: Time to Lock vs. Modulation for a 20Hz Carrier Offset

Figure 6.8: False Locking vs. Modulation for a 20Hz Carrier Offset

BER

Figure 6.9 is the BER performance of the receiver for the different forms of modulation. It can be seen that above 1dB E_b/N_O , the receiver follows the ideal BER curve perfectly. For the errant BPSK data points, it is supposed that the remodulator lost lock briefly, and then regained lock. This loss of lock can again be explained through the variance level thresholds being too strict for this type of modulation. If the PLL recovered lock on the carrier with a 180° phase shift, the recovered data would be inverted to the 'known' data, resulting in bad BER.

Figure 6.9: BER for Different Modulated Signals at a 0Hz Carrier Offset

6.2.2 Effects of Quantization

As seen in the signal generation section, quantization can have an effect on the sensitivity of the receiver. Quantization was assumed from the A/D, and so the performance of the receiver with 8 through 3 bit representation was investigated.

Locking

Figure 6.10 is a plot of the locking percentages versus the number of quantization bits. These results come from a QPSK signal at a 10Hz carrier offset. It can be seen that there is no discernable difference between 256 quantization levels and 8 quantization levels.

Figure 6.10: Locking % vs. Quantization for a QPSK Signal with a 10Hz Carrier Offset

BER

Figure 6.11 is a plot of the BER performance of a QPSK carrier with varying levels of quantization. It can be seen that with only 3 bits of quantization, the BER performance of the remodulator is slightly degraded. Above 3 bits of quantization, and the receiver performance follows the ideal curve.

Figure 6.11: BER for Different Quantizations of a QPSK Signal with a 0Hz Carrier Offset

6.2.3 Pattern Sensitivity

Receivers are often sensitive to certain patterns. If the data is not random, i.e. a periodic data signal, the modulating waveform will consist of sidebands. A periodic data signal is not a sign of an efficiently coded stream of data. Scrambling serves to randomize the data signal, which eliminates discrete sidebands, and error correction coding enhances data recovery, but broadens the frequency spectrum.

In an effort to determine this digital receivers sensitivity, four patterned data signals were evaluated. These patterns were as follows:

pattern 1: [0 1]

this has a frequency of F_b

pattern 2: [0 0 1 1]

this has a frequency of $\frac{F_b}{2}$

pattern 3: [0 0 0 0 1 1 1 1]

this has a frequency of $\frac{F_b}{4}$

pattern 4: [0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1]

this has a frequency of $\frac{F_b}{8}$

The frequencies were determined by the data rate of $F_b = 4800$ bit/s. In the case of QPSK, and OQPSK, the same pattern was shown to both streams of data.

Locking

Figure 6.12 is a plot of the locking percentages versus the different patterns of a QPSK signal with a 20Hz carrier offset. It can be seen for these patterns, that there is no deviance from the average curve, which would suggest the patterned data has no effect on locking. Lower frequency patterned data might produce sidebands which fall within the loop bandwidth and present problems. This is unlikely to ever happen with error correction coding.

Figure 6.12: Locking % vs. Patterned Data on a QPSK Signal with a 20Hz Carrier offset

BER

Figure 6.13 is a plot of the BER performance of a QPSK carrier with varying patterned data. Again, with each pattern, each response follows the ideal BER curve.

Figure 6.13: BER for Patterned Data on a QPSK signal with a 0Hz Carrier Offset

6.2.4 Real life Data

For over-the-air signal testing of this implemented demodulator, the CDGPS terminal was rigged to perform the hardware down-conversion and sampling of the received signal.

Figure 6.14 is a plot of the BER performance of a BPSK carrier downconverted by the CDGPS hardware.

Figure 6.14: BER for BPSK Modulated Satellite Data

A frequency stability issue arose with the real life testing. Initially the loop kept coming out of lock when in the narrowest bandwidth filter. This was eventually deemed to be because the loop could not track the frequency drift, due to the crystal reference used in the CDGPS hardware, in the narrowest filter. This moderately accurate crystal allowed its frequency to vary substantially with the fluctuation in temperature.

Otherwise, under observation, the locking/adaptive filtering algorithms functioned well and the bit synchronizer exceeded expectations. If the loop was not in lock and the bit synchronizer was allowed to run, the synchronization results were usually able to indicate PLL lock before the lock detection algorithm; the bit synchronizer histogram would have returned a meaningful distribution (one big bin, the rest small).

Chapter 7

Conclusions

A receiver which is capable of tracking BPSK, QPSK and OQPSK signals in a low SNR environment has been implemented, simulated and tested. A coded version of the BPSK demodulator was included in the Mobile Knowledge product CDGPS and found to be very successful in its ability to track and demodulate low SNR signals.

Each of the three new and novel implemented algorithms performed very well. The locking algorithm proved to be successful, as the remodulator achieved lock at low E_b/N_O values with a high degree of certainty. The adaptive filtering algorithm was also very successful, narrowing and widening the loop bandwidth as needed. The bit synchronization exceeded expectations, and provided the added bonus of timing information for clock synchronization.

In terms of receiver performance, signals with E_b/N_O s of 0dB were confidently demodulated and tracked through frequency drift. The demodulation of all 3 types of modulation performed equally well, following the ideal BER curve. The digital receiver was not found to be susceptible to patterned data, and able to work with quantized signals down to 4bits without compromising performance.

In terms of locking/false locking probabilities, this receiver provided excellent results. Carrier offset proved to be the biggest reason for both false locking and mean time to achieve lock. However with the frequency searching algorithm implemented, this effect should be kept to a minimum. Quantization and patterned data had no effect on these probabilities.

Although the implemented version on the CDGPS had problems with oscillator stability, it performed up to expectations, and yielded good (near ideal) results.

7.1 Suggestions for Further Work

The results presented in this thesis raise a number of interesting topics for future research:

1. expanding the receiver to include $\pi/4$ and differential N-PSK modulation schemes.
2. expanding the receiver to allow for no knowledge of system (clock) timing, and including the possibility of using interpolation to recover data if the data rate is not an integer multiple of the sampling rate.
3. further analysis of the receiver, including immunity to adjacent channel interference and jamming.

Appendix A

Computer Implementation

As this receiver has been implemented on a DSP, care must be taken in optimizing instructions and data storage/arrays. Some of these techniques include methods to turn an otherwise decimal number into an int16 number:

phase: usually represented in degrees $[0, 360^\circ]$ or radians $[0, 2\pi]$. A 16 bit ‘signed’ integer was used to represent phase:

$$-\pi \Rightarrow -32768 = -PI$$

$$0 \Rightarrow 0$$

$$\pi \Rightarrow 32767 = PI$$

This had been used by Almeida [15] to provide a $\text{mod}(2\pi)$ function (as $32767 + 1 \Rightarrow -32768$).

Scaling factor: $\Theta(\text{int16}, [-PI, PI]) \Rightarrow \frac{(2^{15}-1)}{\pi} \Theta(\text{rad}, [-\pi, \pi])$

frequency: represented in Hz. Again, using a 16 bit integer provides the necessary precision (rather than using a decimal number). Frequencies above $\frac{F_s}{16}$ Hz (self imposed limit) are not useful, so frequencies are scaled from $0 \leftrightarrow \frac{F_s}{16}$ Hz.

Scaling factor: $f(\text{int16}, [0, PI]) \Rightarrow \frac{16 \cdot (2^{15}-1)}{F_s} f(\text{Hz}, [0, \frac{F_s}{16}])$

quantization: filter (FIR, IIR, RRC) coefficients were rounded to the closest power of 2 allowable for integer multiplication

attenuation, averages: attenuation were all in powers of 2, so that shifting could take the place of multiplication. Averages were also computed for 2^N items which allowed for simple shifting by N .

The organization of the *MATLAB* model was based on a hierarchy of functions:

main.m: the main program which directs the operation of the receiver

demodInit.m: sets up the demodulator for either the carrier frequency searching mode, or the demodulation of data

demodulator.m: performs the whole demodulator operation, including the locking and adaptive filtering algorithms

bitSync.m: takes the raw output sampled signal from demodulator.m, achieves bit synchronization and returns recovered data.

This was the implementation format of the coded *C* version on the CDGPS.

Appendix B

Further Noise Explanation

From the expansion of equations (6.2), the noise can be represented by a multiplication of simple terms – equations (B.1b) and (B.2b) represent how the noise was generated on the carrier. Equations (B.1a) and (B.2a) are similar to those found in Gardner [3].

Final noise signal vectors:

$$\begin{aligned}
I_{NOISE} &= -\frac{1}{2}n_1(t) \cdot \sin(\omega_{RR}t) + \frac{1}{2}n_2(t) \cdot \cos(\omega_{RR}t) \\
&= -\frac{1}{2}[\sin((\omega_{RR} - \omega_{n1A})t) + \sin((\omega_{RR} + \omega_{n1B})t)] \\
&\quad + \frac{1}{2}[\cos((\omega_{RR} - \omega_{n2C})t) + \cos((\omega_{RR} + \omega_{n2D})t)] \tag{B.1a} \\
&= -\frac{1}{2}[\sin(\omega_{RR}t) \cdot n_{1A} - \cos(\omega_{RR}t) \cdot \text{hilbert}(n_{1A}) \\
&\quad + \cos(\omega_{RR}t) \cdot \text{hilbert}(n_{1B}) + \sin(\omega_{RR}t) \cdot n_{1B}] \\
&\quad + \frac{1}{2}[\cos(\omega_{RR}t) \cdot n_{2C} + \sin(\omega_{RR}t) \cdot \text{hilbert}(n_{2C}) \\
&\quad - \sin(\omega_{RR}t) \cdot \text{hilbert}(n_{2D}) + \cos(\omega_{RR}t) \cdot n_{2D}] \\
&= -\frac{1}{2}[\sin(\omega_{RR}t) \cdot (n_{1A} + n_{1B}) + \cos(\omega_{RR}t) \cdot (-\text{hilbert}(n_{1A}) + \text{hilbert}(n_{1B}))] \\
&\quad + \frac{1}{2}[\cos(\omega_{RR}t) \cdot (n_{2C} + n_{2D}) + \sin(\omega_{RR}t) \cdot (\text{hilbert}(n_{2C}) - \text{hilbert}(n_{2D}))] \tag{B.1b}
\end{aligned}$$

$$\begin{aligned}
Q_{NOISE} &= \frac{1}{2}n_1(t) \cdot \cos(\omega_{RR}t) + \frac{1}{2}n_2(t) \cdot \sin(\omega_{RR}t) \\
&= \frac{1}{2}[\cos((\omega_{RR} - \omega_{n1A})t) + \cos((\omega_{RR} + \omega_{n1B})t)] \\
&\quad + \frac{1}{2}[\sin((\omega_{RR} - \omega_{n2C})t) + \sin((\omega_{RR} + \omega_{n2D})t)] \tag{B.2a} \\
&= \frac{1}{2}[\cos(\omega_{RR}t) \cdot n_{1A} + \sin(\omega_{RR}t) \cdot \textit{hilbert}(n_{1A}) \\
&\quad - \sin(\omega_{RR}t) \cdot \textit{hilbert}(n_{1B}) + \cos(\omega_{RR}t) \cdot n_{1B}] \\
&\quad + \frac{1}{2}[\cos(\omega_{RR}t) \cdot n_{2C} - \cos(\omega_{RR}t) \cdot \textit{hilbert}(n_{2C}) \\
&\quad + \cos(\omega_{RR}t) \cdot \textit{hilbert}(n_{2D}) + \sin(\omega_{RR}t) \cdot n_{2D}] \\
&= \frac{1}{2}[\cos(\omega_{RR}t) \cdot (n_{1A} + n_{1B}) + \sin(\omega_{RR}t) \cdot (\textit{hilbert}(n_{1A}) - \textit{hilbert}(n_{1B}))] \\
&\quad + \frac{1}{2}[\sin(\omega_{RR}t) \cdot (n_{2C} + n_{2D}) + \cos(\omega_{RR}t) \cdot (-\textit{hilbert}(n_{2C}) + \textit{hilbert}(n_{2D}))] \tag{B.2b}
\end{aligned}$$

Appendix C

Derivations

C.1 Variance of a Filtered Signal

Variance is calculated as:

$$var(f(x)) = \frac{1}{N} \sum_{n=1}^N [f(x) - \overline{f(x)}]^2 \quad (C.1)$$

For variables with a mean of 0, this formula simplifies to $var(f(x)) = \frac{1}{N} \sum_{n=1}^N [f(x)]^2$. Since the relationship between time and frequency variance is $var(signal(t)) = \frac{1}{N} var(signal(w))$, we can proceed to derive the variance of filtered noise:

$$\begin{aligned} var(Noise_{FILTERED}(t)) &= \frac{1}{N} var(Noise_{FILTERED}(w)) \\ &= \frac{1}{N} var(Noise(w) \cdot H(w)) \\ &= \frac{1}{N} \left[\frac{1}{N} \sum_{n=1}^N [Noise(w) \cdot H(w)]^2 \right] \\ &= \frac{\overline{H^2(w)}}{N} \left[\frac{1}{N} \sum_{n=1}^N [Noise(w)]^2 \right] \\ &= \frac{\overline{H^2(w)}}{N} var(Noise(w)) \\ &= \overline{H^2(w)} \cdot var(Noise(t)) \end{aligned} \quad (C.2)$$

where $\overline{H^2(w)}$ is the average of the square of the frequency response of the filter.

This is only valid as long as $Noise(w)$ is of mean 0, and of constant spectral density over all w , i.e. white noise. Because $H(w)$ is an unchanging function, and operates on a random variable, it can be extracted from the variance calculation. The scaling factor between the filtered and unfiltered noise is therefore simply $\overline{H^2(w)}$.

C.2 Variance due to Drift Rate

For a drift rate of $\frac{\Delta f}{T_s}$ Hz/s, there are 2 cases:

$\bar{f}$ is in the middle of the drift range

$$\begin{aligned} var &= \frac{2}{N} \sum_{n=1}^{\frac{N}{2}} [n \cdot \Delta f]^2 \\ &= \frac{2}{N} [1^2 + 2^2 \dots \frac{N^2}{2}] \Delta f^2 \end{aligned} \tag{C.3}$$

$\bar{f}$ is at the edge of the drift range

$$\begin{aligned} var &= \frac{1}{N} \sum_{n=1}^N [n \cdot \Delta f]^2 \\ &= \frac{1}{N} [1^2 + 2^2 \dots N^2] \Delta f^2 \end{aligned} \tag{C.4}$$

Appendix D

Additional Simulation Results

Eb/No	Modulation			
	ideal	BPSK	QPSK	OQPSK
0.0	0.07865	0.29706	0.32659	0.14065
0.5	0.06707	0.45864	0.06553	0.16763
1.0	0.05628	0.39765	0.06350	0.11398
1.5	0.04640	0.04212	0.04434	0.04373
2.0	0.03751	0.03630	0.03305	0.03440
2.5	0.02966	0.32208	0.02456	0.02644
3.0	0.02288	0.01843	0.02284	0.02354
3.5	0.01717	0.01319	0.01409	0.01726
4.0	0.01250	0.00935	0.01141	0.01159
4.5	0.00879	0.00729	0.00932	0.00736
5.0	0.00595	0.00819	0.00603	0.00574
5.5	0.00386	0.00226	0.00282	0.00429
6.0	0.00239	0.00262	0.00192	0.00246
6.5	0.00140	0.00129	0.00159	0.00172
7.0	0.00077	0.00058	0.00095	0.00132
7.5	0.00040	0.00040	0.00105	0.00107
8.0	0.00019	0.00026	0.00064	0.00039

Table D.1: BER for Different Modulations at a 0Hz Carrier Offset

Eb/No	BER (# of bits for quantization)						
	ideal	8	7	6	5	4	3
0.0	0.07865	0.32659	0.25703	0.23994	0.43138	0.11640	0.41102
0.5	0.06707	0.06553	0.07616	0.26000	0.07624	0.25944	0.07948
1.0	0.05628	0.06350	0.05283	0.05430	0.05453	0.21424	0.25585
1.5	0.04640	0.04434	0.04326	0.04564	0.04352	0.04730	0.26739
2.0	0.03751	0.03305	0.03238	0.03652	0.03693	0.03677	0.03999
2.5	0.02966	0.02456	0.02873	0.02531	0.02999	0.03471	0.03450
3.0	0.02288	0.02284	0.02236	0.02257	0.02269	0.02285	0.03305
3.5	0.01717	0.01409	0.01747	0.01603	0.01527	0.01678	0.02111
4.0	0.01250	0.01141	0.01035	0.01277	0.01168	0.01057	0.01864
4.5	0.00879	0.00932	0.01058	0.00802	0.00886	0.00672	0.01356
5.0	0.00595	0.00603	0.00506	0.00647	0.00608	0.00495	0.00956
5.5	0.00386	0.00282	0.00364	0.00405	0.00344	0.00319	0.00680
6.0	0.00239	0.00192	0.00309	0.00249	0.00244	0.00158	0.00577
6.5	0.00140	0.00159	0.00148	0.00190	0.00232	0.00192	0.00286
7.0	0.00077	0.00095	0.00135	0.00061	0.00120	0.00104	0.00234
7.5	0.00040	0.00105	0.00071	0.00060	0.00058	0.00060	0.00154
8.0	0.00019	0.00064	0.00041	0.00054	0.00081	0.00048	0.00147
8.5	0.00008	0.00040	0.00049	0.00048	0.00047	0.00023	0.00102
9.0	0.00003	0.00042	0.00025	0.00033	0.00038	0.00033	0.00081

Table D.2: QPSK BER for Different Quantizations at a 0Hz Carrier Offset

Eb/No	BER (# of bits for quantization)						
	ideal	8	7	6	5	4	3
0.0	0.07865	0.14065	0.27654	0.22634	0.39889	0.25082	0.38166
0.5	0.06707	0.16763	0.07374	0.14783	0.25250	0.07082	0.11640
1.0	0.05628	0.11398	0.04909	0.05599	0.14939	0.23870	0.13561
1.5	0.04640	0.04373	0.04821	0.04449	0.04466	0.04861	0.11295
2.0	0.03751	0.03440	0.03493	0.09790	0.15055	0.03504	0.04638
2.5	0.02966	0.02644	0.02710	0.02861	0.02934	0.02982	0.03807
3.0	0.02288	0.02354	0.01987	0.02399	0.02262	0.02339	0.02799
3.5	0.01717	0.01726	0.01665	0.01570	0.01662	0.01672	0.02025
4.0	0.01250	0.01159	0.01032	0.01252	0.01179	0.01331	0.01661
4.5	0.00879	0.00736	0.00906	0.00677	0.00941	0.00936	0.01304
5.0	0.00595	0.00574	0.00654	0.00480	0.00555	0.00654	0.00877
5.5	0.00386	0.00429	0.00435	0.00362	0.00329	0.00524	0.00784
6.0	0.00239	0.00246	0.00200	0.00299	0.00284	0.00375	0.00478
6.5	0.00140	0.00172	0.00191	0.00196	0.00222	0.00227	0.00345
7.0	0.00077	0.00132	0.00135	0.00074	0.00166	0.00164	0.00258
7.5	0.00040	0.00107	0.00036	0.00054	0.00127	0.00073	0.00161
8.0	0.00019	0.00039	0.00063	0.00087	0.00096	0.00069	0.00153

Table D.3: OQPSK BER for Different Quantizations at a 0Hz Carrier Offset

Eb/No	BER (# of samples per bit)			
	ideal	4	6	8
0.0	0.07865	0.32659	0.18756	0.17425
0.5	0.06707	0.06553	0.30153	0.05994
1.0	0.05628	0.06350	0.04986	0.06066
1.5	0.04640	0.04434	0.05383	0.04360
2.0	0.03751	0.03305	0.03394	0.03590
2.5	0.02966	0.02456	0.03288	0.03343
3.0	0.02288	0.02284	0.02433	0.02753
3.5	0.01717	0.01409	0.01441	0.01391
4.0	0.01250	0.01141	0.01406	0.01221
4.5	0.00879	0.00932	0.01005	0.00793
5.0	0.00595	0.00603	0.00600	0.00537
5.5	0.00386	0.00282	0.00494	0.00500
6.0	0.00239	0.00192	0.00428	0.00288
6.5	0.00140	0.00159	0.00203	0.00108
7.0	0.00077	0.00093	0.00199	0.00235

Table D.4: QPSK BER for Different Sampling Rates at a 0Hz Carrier Offset

Eb/No	BER Pattern				
	ideal	01	0011	00001111	100110001110101
0.0	0.07865	0.45005	0.44531	0.20835	0.23779
0.5	0.06707	0.27079	0.34363	0.06830	0.21470
1.0	0.05628	0.21086	0.05965	0.26598	0.06332
1.5	0.04640	0.03910	0.03962	0.03432	0.04135
2.0	0.03751	0.03457	0.06896	0.11034	0.03438
2.5	0.02966	0.02269	0.02150	0.02954	0.02165
3.0	0.02288	0.02061	0.02142	0.01735	0.02166
3.5	0.01717	0.01603	0.01629	0.01581	0.01718
4.0	0.01250	0.00835	0.00864	0.01284	0.00809
4.5	0.00879	0.00708	0.00659	0.00830	0.00491
5.0	0.00595	0.00550	0.00356	0.00482	0.00729
5.5	0.00386	0.00290	0.00191	0.00411	0.00617
6.0	0.00239	0.00170	0.00259	0.00271	0.00213
6.5	0.00140	0.00044	0.00183	0.00071	0.00109
7.0	0.00077	0.00097	0.00075	0.00057	0.00077

Table D.5: BPSK BER for Different Patterns at a 0Hz Carrier Offset

Eb/No	BER	Pattern			
	ideal	01	0011	00001111	100110001110101
0.0	0.07865	0.08152	0.07488	0.08494	0.06640
0.5	0.06707	0.07146	0.06537	0.06501	0.06564
1.0	0.05628	0.05921	0.05020	0.05347	0.05731
1.5	0.04640	0.05106	0.04338	0.04597	0.04245
2.0	0.03751	0.04865	0.02816	0.03719	0.03173
2.5	0.02966	0.02884	0.02519	0.02816	0.03003
3.0	0.02288	0.02032	0.02050	0.01972	0.02090
3.5	0.01717	0.01552	0.01653	0.01761	0.01922
4.0	0.01250	0.01240	0.01303	0.01071	0.01396
4.5	0.00879	0.01295	0.00627	0.00975	0.00687
5.0	0.00595	0.00960	0.00597	0.00553	0.00605
5.5	0.00386	0.00567	0.00373	0.00417	0.00414
6.0	0.00239	0.00254	0.00179	0.00311	0.00213
6.5	0.00140	0.00183	0.00195	0.00120	0.00095
7.0	0.00077	0.00100	0.00045	0.00057	0.00066

Table D.6: QPSK BER for Different Patterns at a 0Hz Carrier Offset

Eb/No	BER	Pattern			
	ideal	01	0011	00001111	100110001110101
0.0	0.07865	0.49862	0.22287	0.07896	0.46235
0.5	0.06707	0.49875	0.11176	0.06230	0.27262
1.0	0.05628	0.49768	0.05296	0.05634	0.23187
1.5	0.04640	0.49723	0.04530	0.03885	0.14771
2.0	0.03751	0.49568	0.04003	0.03083	0.30641
2.5	0.02966	0.49843	0.03248	0.02991	0.03378
3.0	0.02288	0.49153	0.02189	0.02225	0.02422
3.5	0.01717	0.49831	0.01455	0.01548	0.01890
4.0	0.01250	0.49830	0.01257	0.01283	0.01142
4.5	0.00879	0.49743	0.00640	0.00610	0.00661
5.0	0.00595	0.49533	0.00566	0.00635	0.00640
5.5	0.00386	0.49829	0.00354	0.00423	0.00401
6.0	0.00239	0.49699	0.00082	0.00194	0.00183
6.5	0.00140	0.49641	0.00117	0.00060	0.00136
7.0	0.00077	0.49267	0.00116	0.00094	0.00122

Table D.7: OQPSK BER for Different Patterns at a 0Hz Carrier Offset

Bibliography

- [1] J. Webber and N. Dahnoun. **Implementing a $\pi/4$ Shift D-QPSK Baseband Modem Using the TMS320C50.** Technical report, Texas Instruments, ESIEE Paris, September 1996.
- [2] S. Hinedi and B. Shah. **Acquisition Performance of Various QPSK Carrier Tracking Loops.** *IEEE Transactions on Communications*, 40(9):1426–9, September 1992.
- [3] Floyd M Gardner. ***Phase Lock Techniques*.** Wiley, New York, 2 edition, 1979.
- [4] J. Mark Steber. **PSK Demodulation (Part 1).** Technical Report Vol. 11 No. 2, Watkins-Johnson Company, March/April 1984.
- [5] Martin Oerder Heinrich Meyr and Andreas Polydoros. **On Sampling Rate, Analog Prefiltering, and Sufficient Statistics for Digital Receivers.** *IEEE Transactions on Communications*, 42(12):3208–14, December 1994.
- [6] Jussi Vesma and Tapio Saramaki. **Interpolation Filters with Arbitrary Frequency Response for All-Digital Receivers.** *IEEE*, pages 568–71, 1996.
- [7] Zhang Hang and Markku Renfors. **A Study of Performance of All-Digital Symbol Synchronizers with Polynomial Interpolators.** Technical report, Telecommunications Laboratory, Tampere University of Technology, 2000.

- [8] Jeffery A. Wepman. **Analog-to-Digital Converters and Their Applications in Radio Receivers.** *IEEE Communications Magazine*, pages 39–45, May 1995.
- [9] Alain Blanchard. ***Phase Locked Loops: Applications to Coherent Receiver Design.*** Wiley, Krieger, 1976.
- [10] Jae Chon Lee and Chong Kwan Un. **Performance Analysis of Digital Tanlock Loop.** *IEEE Transactions on Communications*, 30(10):2398–2411, October 1982.
- [11] Chong Kwan Un Hwa Jong Kim and Jae Chon Lee. **The N-Phase Digital Tanlock Loop for Tracking Suppressed-Carrier N-ary PSK Signals.** *IEEE Transactions on Communications*, 33(9):904–9, September 1985.
- [12] C. Young. **Application of PLLs**, January 1997.
- [13] Carlos A. Pomalaza-Raez. **Noise Analysis of a Digital Tanlock Loop.** *IEEE Transactions on Aerospace and Electronic Systems*, 24(6):713–8, November 1988.
- [14] Donald R. Stephens. ***Phase Locked Loops for Wireless Communications: Digital and Analog Implementation.*** Kluwer Academic, Boston, 1998.
- [15] Teresa M. Almeida and Moises S. Piedade. **High Performance Analog and Digital PLL Design.** *IEEE*, pages 394–7, 1999.
- [16] Emmanuel C. Ifeachor and Barrie W. Jervis. ***Digital Signal Processing: A Practical Approach.*** Addison-Wesley, Reading, 1993.
- [17] Edward W. Kamen and Bonnie S. Heck. ***Fundamentals of Signals and Systems.*** Prentice Hall, New Jersey, 1997.

- [18] Holly C. Osborne. **Stability Analysis of an Nth Power Digital Phase-Locked Loop – Part I: First Order DPLL.** *IEEE Transactions on Communications*, 28(8):1343–54, August 1980.
- [19] Holly C. Osborne. **Stability Analysis of an Nth Power Digital Phase-Locked Loop – Part II: Second- and Third-Order DPLLs.** *IEEE Transactions on Communications*, 28(8):1355–64, August 1980.
- [20] William C. Lindsey and Chak Ming Chie. **A Survey of Digital Phase-Locked Loops.** *Proceedings of the IEEE*, 69(4):410–31, April 1981.
- [21] P. K. Jain P. M. Rao and Padmavathy C.S. **Coherent Carrier Regeneration Using a Long Loop PLL Technique.** *Applied Microwave & Wireless*, pages 28–42, 2000.
- [22] William F. Egan. ***Phase Lock Basics.*** Wiley, New York, 1998.
- [23] Richard Tervo and Rogerio A. Enriquez. **Analysis of Digital Tanlock Loop with Adaptive Filtering.** *IEEE Pacific Rim*, pages 5–8, 1993.
- [24] Andy Bateman. ***Digital Communications.*** Addison-Wesley, Singapore, 1998.
- [25] Bernard Sklar. ***Digital Communications, Fundamentals and Applications.*** PTR Prentice Hall, New Jersey, 1988.