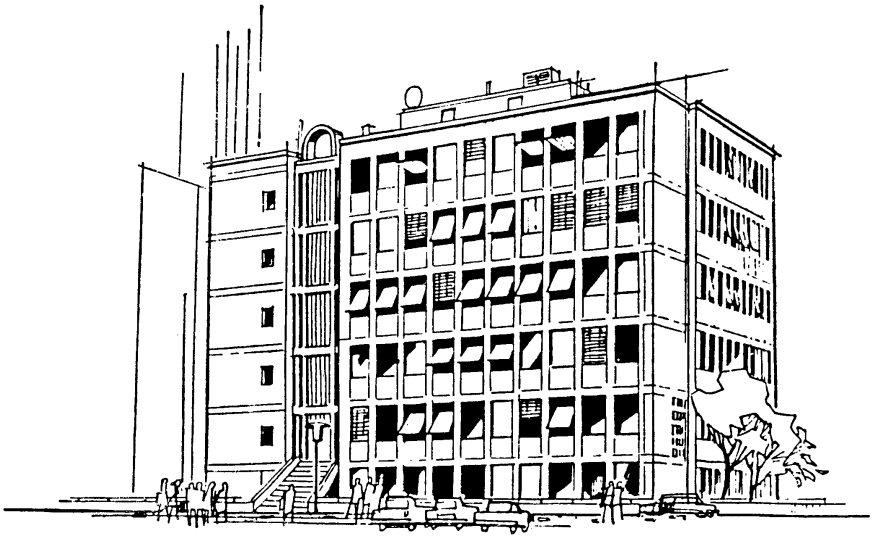
Master's Thesis

All-digital Online Timing Recovery with a Kalman Filter

Matti Ukkola



Professur für Leitungsgebundene Übertragungstechnik
Technische Universität München
Prof. Dr. -Ing. Norbert Hanik

Technische Universität München
Professur für
Leitungsgebundene Übertragungstechnik
Prof. Dr.-Ing. Norbert Hanik

Master's Thesis

All-digital Online Timing Recovery with a Kalman Filter

Vorgelegt von:
Matti Ukkola

München, Dezember 2024

Betreut von:
M.Sc. Bensu Baran Akin, TUM LÜT
M.Sc. Carl Valjus, DLR KN

Master's Thesis an der
Professur für Leitungsgebundene Übertragungstechnik (LÜT)
der Technischen Universität München (TUM)
Titel : All-digital Online Timing Recovery with a Kalman Filter
Autor : Matti Ukkola

Matti Ukkola
Rupprechstr. 8A
80636
matti.ukkola@tum.de

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

München, 18.12.2024

Ort, Datum (Matti Ukkola)

Contents

Abstract **1**

1 Introduction **3**
1.1 FSO DTE channel 5
1.2 Receiver architecture 6
1.3 Thesis structure 8

2 Timing recovery **11**
2.1 Timing error 11
2.2 Interpolation 14
2.3 Theoretical estimation limits 15
2.4 Timing error detection algorithms 18
2.4.1 Lee timing error detector 19

3 Kalman Filter **23**
3.1 Motivation 23
3.2 Derivation 23
3.2.1 System model 23
3.2.2 Observation model 25
3.2.3 A priori state estimate 25
3.2.4 A priori covariance estimate 26
3.2.5 Innovation 26
3.2.6 Innovation covariance 26
3.2.7 Kalman gain 27
3.2.8 State update 27
3.2.9 A posteriori covariance estimate 28
3.2.10 Summary 28
3.2.11 Initial state 29
3.3 Simulation 29

4 System **35**
4.1 Constraints . . . 35
4.2 Overview . . . 35

5 Implementation **37**
5.1 Timing constraints . . . 37
5.2 Fixed-point arithmetic . . . 38
5.3 Lee . . . 38
5.4 Buffer . . . 39
5.5 Kalman . . . 39
5.5.1 Estimating computational complexity . . . 39
5.5.2 State . . . 39
5.5.3 A priori covariance estimation . . . 41
5.5.4 Innovation . . . 41
5.5.5 Innovation variance . . . 42
5.5.6 Kalman gain . . . 42
5.5.7 State update . . . 44
5.5.8 A posteriori covariance estimate . . . 44
5.5.9 Summary . . . 45
5.5.10 Covariance limits . . . 45

6 Results **51**
6.1 Verification of VHDL implementation using GHDL . . . 51
6.2 FPGA resource usage and timing verification . . . 52
6.3 Experimental results using a fading testbench . . . 53

7 Conclusion and future outlook **61**
7.1 Future outlook . . . 61
7.2 Conclusion . . . 61

Appendix A **65**
7.1 MCRB constant derivation . . . 65
7.1.1 Evaluation of denominator . . . 65
7.1.2 Evaluation of numerator . . . 66
7.1.3 Equation for constant . . . 69
7.2 FPGA Resource utilization . . . 69

Bibliography **73**

List of Figures

1.1 Optical groundstation at DLR KN institute, [DLRb] 4
1.2 OSIRISv3 FSO satellite terminal, [DLRc] 4
1.3 Fading in a typical DTE FSO link, [Con17] 7
1.4 Optical dual polarization intradyne coherent receiver 9
1.5 Digital signal processing steps 10

2.1 Linear timing error $\theta(t)$ due to an initial positive phase offset and a positive frequency offset, causing a linear increase over time 11
2.2 Linear timing error $\theta(t)$ taken to modulo 1, the timing error becomes periodic 12
2.3 Timing error's effect on sampled values at receiver. Up: ADC frequency is lower than that of the DAC. Down: ADC sampling frequency is higher than that of the DAC. Crosses mark samples that are either skipped or repeated. 13
2.4 Raised cosine filter with varying β, $H(f) = |G(f)|^2$ as stated in Equation (2.9) 17
2.5 Timing recovery with a feedback structure, for example when using the Gardner TED algorithm 19
2.6 Timing recovery with a feedforward structure, for example when using the Lee TED algorithm 19
2.7 Lee noise sensitivity analysis against the MCRB 21

3.1 Python simulation of the phase of the timing error before and after filtering during fading. Top left: Phase of the timing error, or the timing error in modulo one, in samples. Top right: Observed timing error in samples, which is the timing error that has been distorted by an AWGN channel. . 32
3.2 Kalman filter behaviour during high noise. The maximum value of the phase is set to 100. 33

4.1 Timing recovery system with Kalman filter 36

5.1 Computational graph, weights along the longest path are underlined . . . 46
5.2 Covariance estimation limits, $p_{00,\mathrm{k}|\mathrm{k}-1}$ 49

5.3 Covariance estimation limits, $p_{01,k|k-1} = p_{10,k|k-1}$ 49
5.4 Covariance estimation limits, linear, $p_{01,k|k-1} = p_{10,k|k-1}$ 50
5.5 Covariance estimation limits, $p_{11,k|k-1}$ 50

6.1 Computational graph for FPGA, weights along the longest path are underlined 52
6.2 VHDL logic simulation showing the input and output phase of the Kalman filter. The phase, as expected, wraps around when reaching one. There is no noise, but observation variance is set to 1/4 resulting in slower convergence. 54
6.3 VHDL logic simulation showing the absolute value of the error between input and output phase of the Kalman filter. There is no noise, but observation variance is set to 1/4 resulting in slower convergence. 55
6.4 Conceptual diagram of the experimental setup. The fading testbench is depicted as an attenuator. 58
6.5 Experimental setup using a fading testbench 59
6.6 System performance during fading, observation variance=0, corresponds to trusting the output of the Lee algorithm completely. 60
6.7 System performance during fading, observation variance=1e5, corresponds to the Kalman filter using its internal linear model to predict the phase while largely ignoring its input. 60

7.1 MCRB constant $\epsilon(\beta)$ for $\beta \in [0, 1]$ 69

List of Tables

2.1 Timing error detection algorithms 18

5.1 Typical processor latencies 39
5.2 Operations required for a priori covariance estimation 41
5.3 Operations required for innovation 42
5.4 Operations required for innovation variance 42
5.5 Operations required for Kalman gain 42
5.6 Operations required for state update 44
5.7 Operations required for a posteriori covariance estimate 45
5.8 Estimated number of operations required for the Kalman filter update using latencies from Table 5.1 45
5.9 A priori covariance estimate ranges, k $\in [0, 1000]$ 48

6.1 FPGA resource utilization 53

7.1 FPGA resource utilization, part 1. Values in parenthesis are relative to the total amount of available resources. 70
7.2 FPGA resource utilization, part 2. Values in parenthesis are relative to the total amount of available resources. 70

Abstract

Free-space optical communication is a promising technology that enables secure and high throughput for inter-satellite and direct-to-Earth communication links and helps achieve global connectivity. One of the challenges for free-space optical communication systems with signals passing through the atmosphere is synchronizing the sender and receiver sampling times under strongly and quickly varying channel conditions, for example, due to scintillation caused by atmospheric turbulence. During periods of low signal quality, the sampling time should not be lost and in case it is, it should be recovered as soon as possible to limit data loss.

This thesis aims to provide a novel approach for all-digital timing recovery aimed at free-space optical communication systems by combining the Lee algorithm, a feed-forward timing error detector, with a Kalman filter followed by an interpolator. The approach combines information from the channel conditions under which the timing error measurement was made with the estimate based on a physical model and previous measurements. The Kalman filter is first derived from the general Kalman filter equations. This is followed by a Python implementation and verification of the filter against a simulated typical free-space optical channel resulting into distortions of the timing error detection. The simulations show improved performance of the timing error detection during fading and during low signal-to-noise ratio when using a Kalman filter.

The simulations are followed by an real-time implementation of the Kalman filter on an field programmable gate array (FPGA) as part of an all-digital timing recovery chain based on the Lee timing error detection algorithm. The FPGA is connected to an experimental test setup sending a single- and receiving a dual-polarized optical quadrature phase-shift keying (QPSK) signal at 2 GBaud with an optical fading testbench to validate timing recovery performance during signal fading.

The results show that the Kalman filter implementation does not constrain system speed, uses only 4 % of the total used resources for timing recovery and improves the bit-error-rate during fades when compared to using the Lee algorithm without the Kalman filter. The thesis therefore recommends using a Kalman filter when using the Lee algorithm in context of free-space optical communication with links passing through the atmosphere.

1 Introduction

As demand for global broadband connectivity keep increasing, the requirements for coverage, throughput, reliability and latency of communication networks also become more demanding. Terrestrial networks become less economically viable in sparsely populated areas as the installation costs of cable stay largely the same, but the number of customers decreases [DLRa]. Providing connectivity via satellite networks scales does not suffer from this limitation, as the same satellites can be used everywhere on the globe.
In terrestrial networks, optical fiber has increasingly replaced copper-based networks. For satellite networks, optical links between satellites and even in direct-to-earth (DTE) links provide an attractive option, complementing or in parts replacing radio frequency (RF) systems. An advantage of optical links when compared to RF links is magnitudes higher bandwidth, allowing for higher throughput. For DTE links using automatic repeat requests, 200 GBit/s throughput has been demonstrated [SRB+23].
Specifically in free-space optical communication (FSO), the links are highly focused. A model for optical beam divergence is the Gaussian beam. The radius W at which the beam intensity falls to $1/e$, and within which $1 - 1/e^2 = 86.5\ \%$ of the transmit power is concentrated, in distance z from the sender is given as

$$W(z) = W_0 \left[1 + \left(\frac{z}{z_0}\right)^2\right]^{1/2}, \quad z_0 := W_0^2 \frac{\pi}{\lambda} \tag{1.1}$$

where W_0 is the radius of the sender aperture, usually the optical lens. Plugging in z_0 yields for large distances due to $z \to \infty$

$$W(z) = W_0 \left[1 + \left(\frac{z\lambda}{W_0^2 \pi}\right)^2\right]^{1/2} \approx W_0 \left[\left(\frac{z\lambda}{W_0^2 \pi}\right)^2\right]^{1/2}$$

$$W(z) \approx \frac{\lambda}{W_0 \pi} z \tag{1.2}$$

Assuming a typical optical wavelength, $\lambda = 1500$ nm and a modestly sized aperture of 30 mm which is able to even fit a small CubeSat of $10 \times 10 \times 10$ cm^3 at a typical low Earth orbit (LEO) height of 500 km yields by Equation (1.2) a beam radius $W(z) =$

Figure 1.1: Optical groundstation at DLR KN institute, [DLRb]

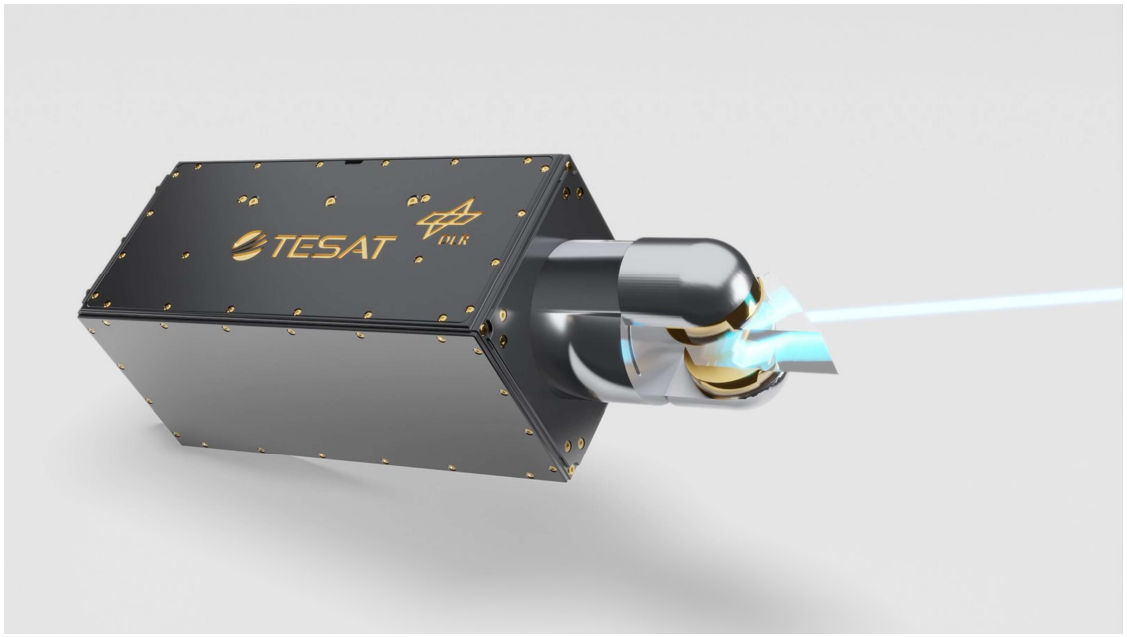


Figure 1.2: OSIRISv3 FSO satellite terminal, [DLRc]

$\frac{1.550\times10^{-6}\text{m}}{30\times10^{-3}\text{m}\pi}$ 500×10^3 m = 8.22 m. The low beam divergence means that interference to and from other systems is reduced. This is potentially one of the reasons why no frequency filing at International Telecommunications Union is to date required for free-space optical links.

To better understand how DTE optical free-space terminals can look like on space and on Earth, Figure 1.1 depicts an optical ground station and Figure 1.2 the OSIRISv3 FSO terminal that is the size of a 3U CubeSat, $30 \times 10 \times 10$ m^3.

Another consequence of the narrow beamwidth is increased security of the communication link. If a satellite sends down an optical signal to a ground station or to another satellite, in order for a bad actor to be able to listen in on the transmission, it would have to be located approximately within 10 m of the receiver. This means that the communication is more secure, as it is more difficult for a third party to listen in on the transmission.

1.1 FSO DTE channel

For FSO DTE links, in which light propagates through the atmosphere, the refractive index of the air changes due to atmospheric turbulence. This can be modelled by a lognormal distribution with mean $\mu = 1$ and variance σ^2 defined as

$$\sigma^2 = e^{\sigma_{\text{p}}^2} - 1 \tag{1.3}$$

according to [GPS+17] and illustrated in Figure 1.3 [Con17]. The coherence time of the channel disturbed by turbulence is in the order of milliseconds. Within the coherence time the channel is assumed to be constant. For symbol rates of several Gbaud this corresponds to millions of symbols having same channel conditions. Therefore, within the scope of this thesis the channel is assumed to be quasi-static within a burst of data containing up to a million symbols.

Another effect important for the link is the Doppler shift as this contributes to the amount of frequency difference that needs to be compensated by the receiver. The orbital speed of a satellite flying at height h is given as

$$v = \sqrt{\frac{GM}{R_E + h}} \tag{1.4}$$

with Earth radius $R_E = 6378$ km, gravitational constant $G = 66.7 \times 10^{-12}$ m^3/(kg s^2) and Earth mass $M = 5.97 \times 10^{24}$ kg.

Using the velocity, the relative Doppler shift can now be derived

$$\frac{f}{f_0} = \frac{v}{c} \tag{1.5}$$

with light speed $c = 300 \times 10^6$ m/s.
To get a worst-case estimate for the Doppler shift between a ground station (GS) and a satellite, the highest velocity difference is chosen. According to Equation (1.4), the velocity of a satellite increases the lower the satellite's orbit. A lower bound for a typical LEO satellite orbit height of 400 km is therefore chosen. This yields

$$v = \sqrt{\frac{66.7 \times 10^{-12} \cdot 5.97 \times 10^{24}}{6.378 \times 10^6 + 400 \times 10^3}}\ \text{m/s} = 7665\ \text{m/s}$$

$$\left.\frac{f}{f_0}\right|_{\text{GS-sat}} = \frac{7665\ \text{m/s}}{300 \times 10^6\ \text{m/s}} = 25.56\ \text{ppm} \tag{1.6}$$

When observing intersatellite links, the worst case is when the satellites are flying in opposite directions. The Doppler shift compared to a satellite-GS link is therefore doubled, resulting in

$$\left.\frac{f}{f_0}\right|_{\text{sat-sat}} = 2 \cdot 25.5651 \approx 51\ \text{ppm}. \tag{1.7}$$

As the Doppler shift is much slower than the fading, the assumption of a quasi-static channel still holds.

1.2 Receiver architecture

With FSO there are several different modulation techniques available. Using coherent modulation schemes based on phase modulation allows for higher sensitivity than amplitude modulation schemes while offering higher data rates. However, this requires a coherent receiver as the receiver to be synchronized with the sender [KT08]. This results in increased complexity of the communication system. Designing coherent receivers to work under the constraints set by the FSO DTE link is an ongoing field of research.
In this thesis, the receiver used is an optical intradyne dual polarization receiver, as depicted in Figure 1.4. The received optical signal is first divided by a polarization beam splitter (PBS) into its x and y components. They enter their corresponding 2x4 90° hybrid which in addition to the input signal component also takes in the optical local oscillator signal. The signal path continues to the balanced receiver (BD), which feeds the output of the hybrid first into four diodes and consequently adds the signals. The BD outputs the I and Q components as two electrical signals, which are amplified and

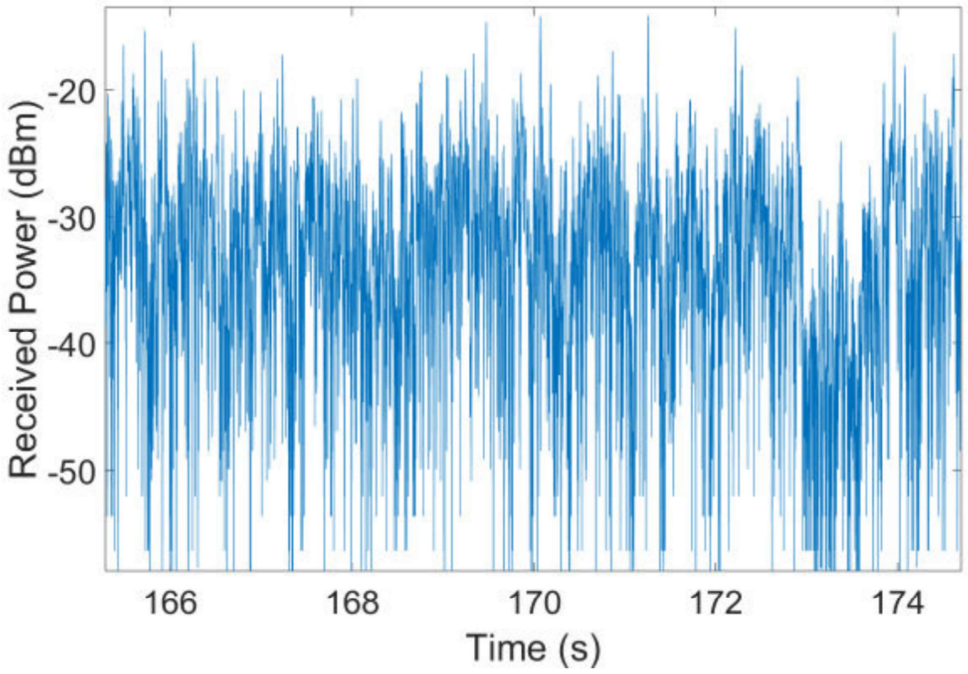


Figure 1.3: Fading in a typical DTE FSO link, [Con17]

finally converted into the digital domain by an analog-to-digital converter (ADC). As the receiver is intradyne, compensation for distortions is done in the digital domain, after the ADCs.

In the digital domain, it is commonly assumed that the symbols are sampled at the optimal points. However, this might in practice not be true. In the context of this thesis, the ADCs are integrated into the digital signal processing (DSP) unit, the field programmable gate array (FPGA), and the sampling time of the ADCs can not be adjusted. Instead, they run freely and therefore the sampling time is not ideal. In addition, the sampling time also changes over time, as the ADC sampling frequency does not exactly match that of the sender in real systems. Consequently, the assumption of ideal sampling time does not hold.

Thus, the DSP must also take care of timing recovery, a process after which the symbol timing is close to optimal. In theory the timing recovery does not need to be placed at the first step, however the amount of samples usually decreases by a factor of two or more, depending on the timing error detection method used. Therefore, it is common to place the timing recovery towards the beginning of the DSP chain. The method used also determines the minimum sampling rate of the ADC. More details on timing error detection can be found in Chapter 2. An overview of the whole DSP chain is shown in

Figure 1.5. The focus of this thesis is on the timing recovery, which is highlighted in the diagram.

1.3 Thesis structure

The thesis is structured as follows. First, the concept of timing error is introduced in Chapter 2. The chapter continues with an overview of timing error detection algorithms and follows up with a brief introduction to using interpolation to correct the timing error. Afterwards, the Lee timing error detection algorithm that is taken as a basis for this thesis, is analyzed in more detail with corresponding simulations in Python. The introduction of timing error detection and correction is followed by an analysis of the bounds for timing error detection accuracy in Chapter 2.3. After obtaining a better understanding of the timing error and its estimation as well as the limits of the estimation, the Kalman filter is introduced in Chapter 3. Here, the general form Kalman filter equations are specialized for the specific case for timing error filtering based on a linear system model for the timing error. The behaviour during a fade as well as in general under noisy conditions is simulated in Python and the results showcased. The full timing recovery system is presented in Chapter 4, which also specifies the constraints under which the timing recovery must function. After the overview, Chapter 5 focuses on the implementation considerations for an FPGA. The Kalman filter equations in matrix formulation are broken down into their components and simplified. This allows the computational complexity of the individual operations to be estimated and the results are illustrated in a computational graph. Additional care is taken to analyze possible fixed-point division algorithms needed to implement the Kalman filter. In Chapter 6, the very high speed integrated circuit hardware description language (VHDL) tools used are introduced. The implementation is functionally verified by using the free, open-source VHDL simulation software G Hardware Design Language (GHDL). After the functional standalone verification of the Kalman filter, the resource usage and timing of the Kalman filter as part of the full DSP system are analyzed. Finally, an experimental setup using a fading testbench is showcased, analyzing the performance in a running system during fades. The central findings are summarized in Chapter 7 together with a future outlook of applying a Kalman filter in the context of DSP for FSO.

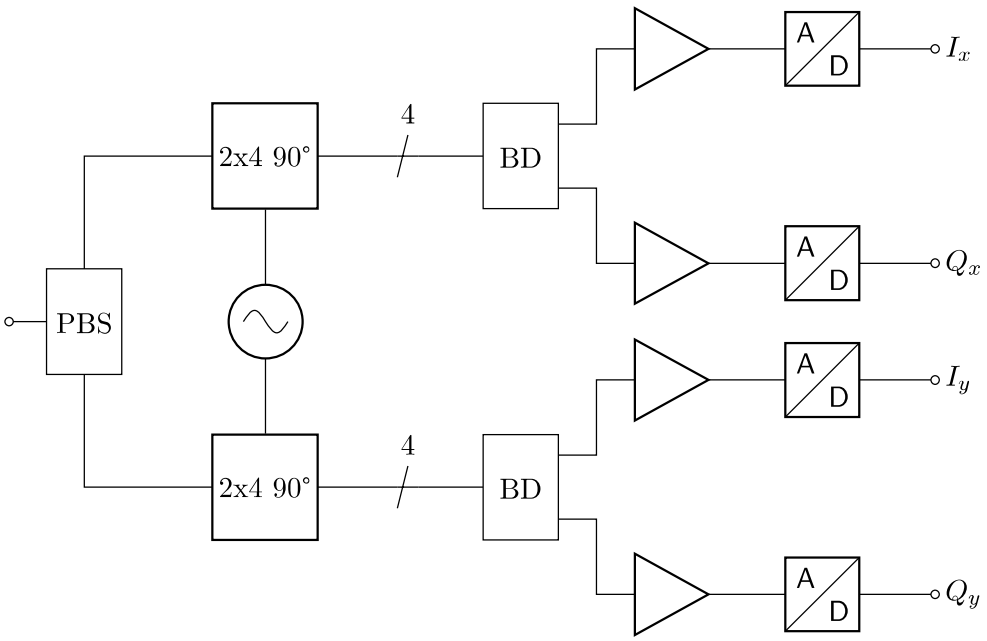


Figure 1.4: Optical dual polarization intradyne coherent receiver

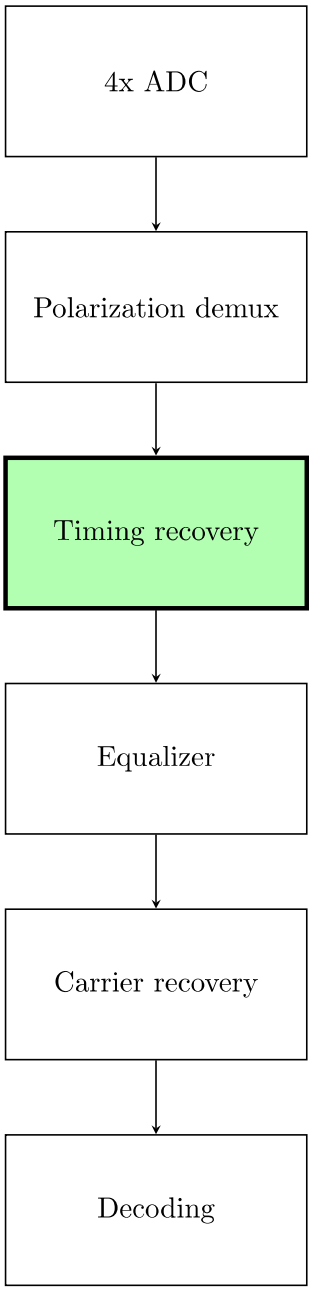


Figure 1.5: Digital signal processing steps

2 Timing recovery

2.1 Timing error

The purpose of a timing error detector (TED) is to estimate the timing error between a sender and a receiver. In digital receivers, the timing error can be interpreted as the misaligment of the receiver's ADC sampling time compared against the digital-to-analog converter (DAC) sampling time of the sender. This initial misalignment changes primarily due to the ADC sampling frequency offset with respect to the DAC of the sender. This causes a linear change in the timing error over time. A qualitative example of the timing error evolution can be seen Figure 2.1.

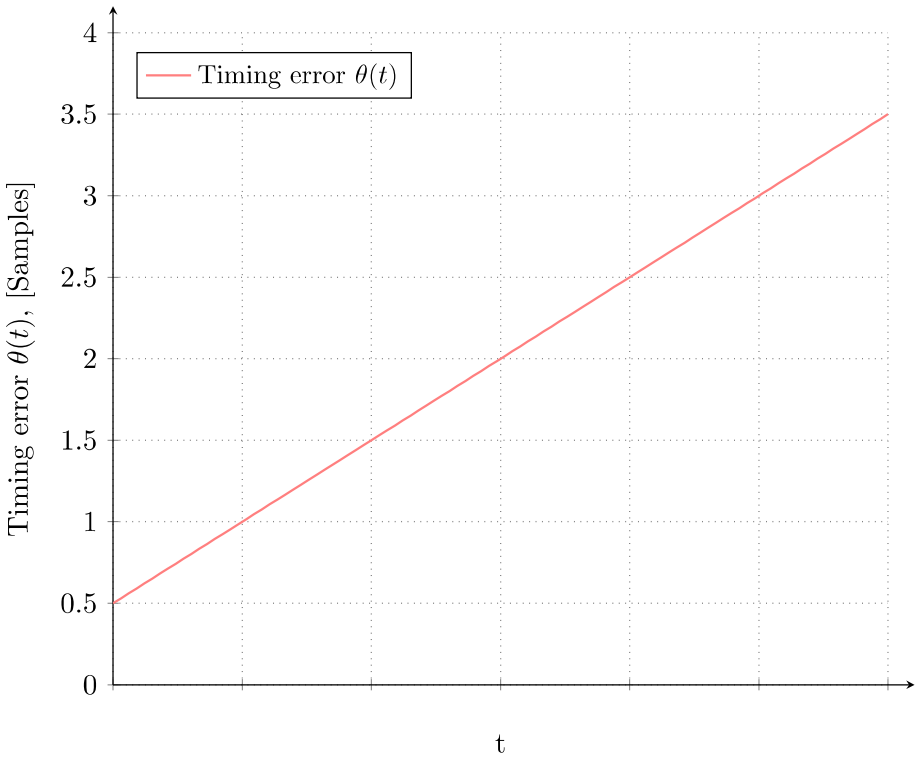


Figure 2.1: Linear timing error $\theta(t)$ due to an initial positive phase offset and a positive frequency offset, causing a linear increase over time

When the timing error increases over 1, the currently processed sample may be skipped by the receiver. Consequently, the timing error goes back one sample. Therefore, the evolution of the timing error is periodic and behaves much like a phase, as depicted in Figure 2.2.

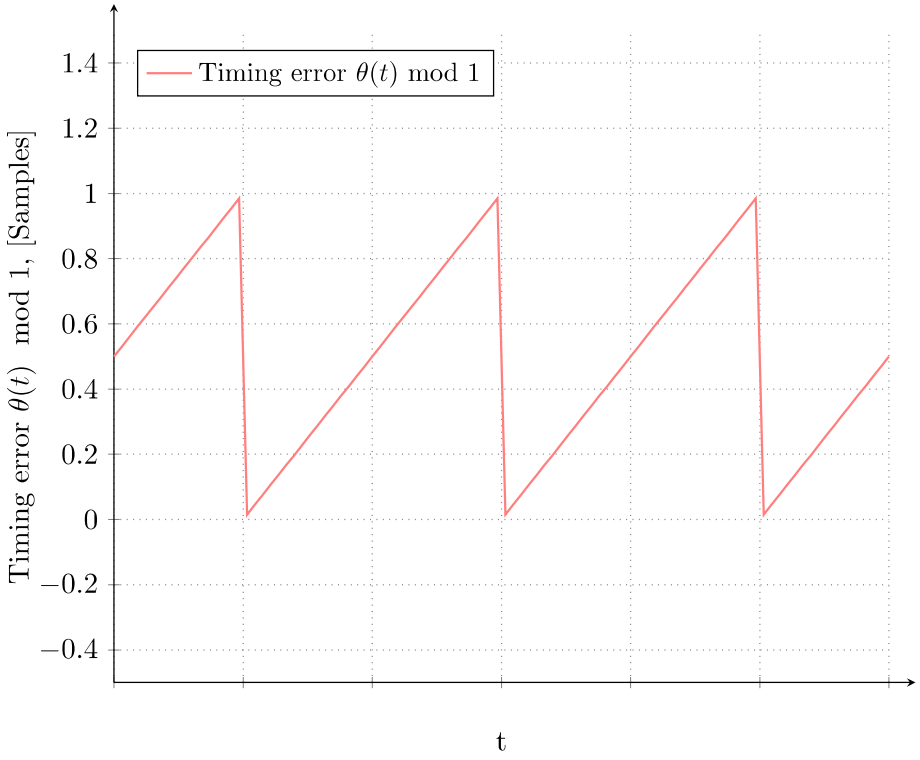


Figure 2.2: Linear timing error $\theta(t)$ taken to modulo 1, the timing error becomes periodic

The effect of the timing error on the sampled values is illustrated in Figure 2.3 with a binary phase-shift keying signal consisting of a $\{+1, -1, +1, -1, ...\}$ symbol sequence. Moreover, the timing error is influenced potentially due to the Doppler shift typical to satellites in LEO as discussed in Chapter 1. Finally, the timing error is influenced by the frequency drift due to temperature variation. As these effects change slowly when compared to the evolution of the timing error due to the frequency offset, they can be considered constant for the purpose of the timing error detection window which is measured in micro- or milliseconds. These effects are thus of secondary importance and not considered in the timing error detection process within the scope of this thesis.
The timing error can then be used to correct the timing error in the received samples for example by using interpolation as discussed in Section 2.2. For the TED, there are different algorithms to choose from. The design criteria considered are

- Required number of samples per symbol (SPS)

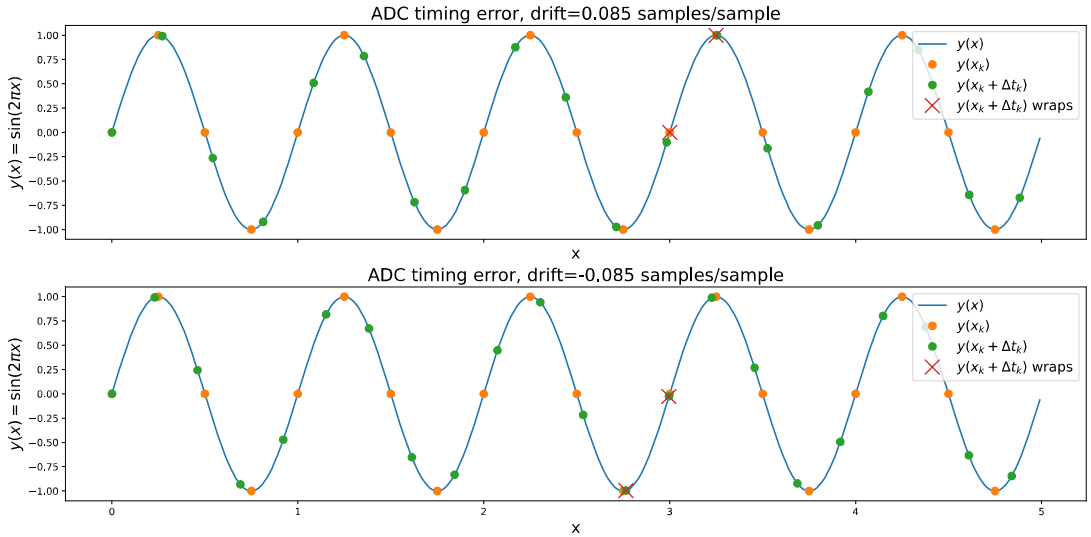


Figure 2.3: Timing error's effect on sampled values at receiver. Up: ADC frequency is lower than that of the DAC. Down: ADC sampling frequency is higher than that of the DAC. Crosses mark samples that are either skipped or repeated.

- Robustness to fading
- Robustness to noise
- Data-aided / non-data aided
- Computational complexity

The design criteria are motivated in the following. First of all, the required number of SPS. The higher the required number of SPS, the faster the ADC must run. This becomes especially relevant at high symbol rates where the ADC speed commonly is a bottleneck. In the context of optical communication and therefore by extension FSO, this is especially relevant due to the inherently high data rates.
Robustness to fading is the second crucial factor when choosing the TED algorithm for timing recovery. As discussed in 1, the FSO channel is highly susceptible to fading. The system must therefore be able to minimize performance degradation during the fade and as soon as the fade is over and the signal comes back, continue nominal operation as quickly as possible.
The third design criteria considered is the timing recovery's robustness to noise. As with any channel and especially with wireless channels, the FSO channel is noisy. Therefore, the TED must be able to cope with noise, which in first approximation is assumed to be zero-mean Gaussian.

With a data-aided TED, training symbols or equivalent must be sent in addition to the useful data. This reduces the throughput of useful information, also known as the goodput, when compared against a non-data aided approach where any sufficiently random data can be used. Therefore, non-data aided TEDs are preferred.
The last design criteria is computational complexity. The selection of algorithms can be optimized more heavily on accuracy rather than speed when the algorithms are run in an offline context. Practically, this is done by measuring samples for a certain period of time, storing them and processing them later with a potentially different signal processor with more power and a less-constrained environment. An example for such an environment is as a computer running a MATLAB script or a Python program. Notably, the time for the evaluation is not strictly bounded. However, when running timing recovery online, the algorithm must operate quickly and reliably enough to process the incoming sample stream in real-time. The deadline is given by the next incoming sample or batch of samples and ultimately determined by the speed of the ADC. In addition, the execution environment is often more constrained, for example the algorithms may be implemented on an FPGA or on an application-specific integrated circuit. Therefore, low computational complexity is preferred.

2.2 Interpolation

A classical textbook approach to correcting the timing error is by adjusting the sampling speed of the ADC. As discussed previously, in some systems such as the one presented within this thesis, adjusting the sampling of the ADC is not possible. For example, the ADC might run at a fixed speed that cannot be changed. In this case, the timing needs to be corrected via some other method. Instead of changing the ADC sampling time, the values of the samples can be adjusted as well to achieve the same effect [Gar93]. How this is done is by conceptually fitting a function to a set of received samples or interpolating between them. Then, by using the function, the value of each sample can be changed to either move forwards or backwards in time. The shift is limited by the increasing error, as at some point the fitted function is too far away from the received signal. Practically, when the shift in time increases to more then the time between two samples, then the indexes of the samples are increased or decreased and the shift reset.
Different methods can be used for interpolation. Piece-wise linear interpolation is the easiest, however using a higher order polynomial gives more accurate results. Trigonometric interpolation can also be used to further increase accuracy, however the computational effort is increased. Within this thesis, cubic interpolation with Lagrange polynomials is used as a middle-ground. A computationally efficient way of calculating this is to use the

Farrow structure [ZC11].
This means that the remaining aspect to remove the timing error is detecting it. After the detection, the error can be given to the interpolator which will invert the shift in time, resulting in optimally sampled symbols.
A practical note to interpolation is that the skipping and repeating of samples needs to be taken care of. The prevent design considerations for the interpolator, this logic is implemented after the TED and before the interpolator.

2.3 Theoretical estimation limits

In order to assess the quality of any parameter estimator, which includes any TED as well, their mathematical qualities such as bias and variance may be used. It is desired that the estimator is unbiased and that the variance is minimized.
In the case of the Lee TED, it is proven[Lee02] that the estimator is approximately unbiased for a large number of samples. As the symbol rates are in the order Gbaud, using even thousands of samples takes only microseconds during which the channel stays quasi-static. Therefore it is assumed that the Lee algorithm is sufficiently unbiased in the context of this timing error estimation problem.
For the variance of estimators, a popular tool is the Cramér-Rao bound (CRB), which provides a lower limit for the variance

$$Var[\lambda] \geq CRB(\lambda) \tag{2.1}$$

where λ is the parameter to be estimated. How difficult the equation for CRB bound is to evaluate, depends on the estimation problem. The estimation problem of the receiver is described in the following.
The received complex waveform within a time interval T_0 is described as

$$r(t) = s(t) + w(t)$$

with received signal $r(t)$, sent signal $s(t)$ and Gaussian noise $w(t)$. The received signal is mostly known, but has unknown parameters including carrier phase θ, symbol duration τ, carrier frequency error ν as well as the transmitted data.
It is common that only one of θ, τ and ν is estimated, the data is assumed to be a so-called unwanted parameter. A single parameter to be estimated is chosen from set $\{\theta, \tau, \nu\}$ and the chosen parameter is called λ. The two other parameters as well as the data are put inside vector $\mathbf{u}$ with probability density function $p(\mathbf{u})$ which is independent from λ.
It is assumed that $r(t)$ can be represented with enough accuracy using a finite-dimensional

vector $\mathbf{r}$. Given $\hat{\lambda}(\mathbf{u})$ is an unbiased estimator, a lower bound to the error is given by the CRB bound [DMR94]

$$CRB(\lambda) = \frac{1}{E_{\mathbf{r}}\left[\left(\frac{\partial \ln p(\mathbf{r}|\lambda)}{\partial \lambda}\right)^2\right]} \tag{2.2}$$

where the conditional probability $p(\mathbf{r}|\lambda)$ is given by the integral [DMR94]

$$p(\mathbf{r}|\lambda) = \int_{-\infty}^{\infty} p(\mathbf{r}|\mathbf{u}, \lambda)\, d\mathbf{u} \tag{2.3}$$

The difficulty in calculating this bound typically arises because an analytical solution for Equation (2.3) may not exist, or the expectation in Equation (2.2) may be computationally infeasible to evaluate. As evaluating the CRB can be challenging, alternative bounds such as the modified Cramér-Rao bound (MCRB) may be used to determine bounds of frequency, phase and timing error estimation. To this end, the MCRB bound is derived within the following.

Replacing the expectation $E_{\mathbf{r}} \rightarrow E_{\mathbf{r},\mathbf{u}}$ yields according to [DMR94]

$$MCRB(\lambda) = \frac{1}{E_{\mathbf{r},\mathbf{u}}\left[\frac{\partial \ln p(\mathbf{r}|\mathbf{u},\lambda)}{\partial \lambda}\right]^2} \tag{2.4}$$

To note is that $MCRB \leq CRB$ and therefore less strict as both are lower bounds. The discrepancy is highest at low SNR and becomes small for large SNR. Lower bound to obtain for carrier frequency ν yields

$$MCRB(\nu) = \frac{3T}{2\pi^2(LT)^3}\frac{1}{E_s/N_0} \tag{2.5}$$

with symbol duration T, number of symbols L, energy per symbol to noise power spectral density E_s/N_0 which for QPSK is $(E_s/N_0)_{\text{QPSK}} = E_b/N_0 \log_2(M = 4)$ [DMR94]. Similarly applied for carrier phase θ yields

$$MCRB(\theta) = \frac{1}{2L}\frac{1}{E_s/N_0} = \frac{B_L T}{E_s/N_0} \tag{2.6}$$

and finally for timing error

$$MCRB(\tau) = \frac{1}{2L}\frac{1}{E_s/N_0}\frac{T^2}{4\pi^2\epsilon} = \frac{B_L T}{4\pi^2\epsilon}\frac{T^2}{E_s/N_0} \tag{2.7}$$

with

$$\epsilon = \frac{\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2\, df}{\int_{-\infty}^{\infty} |G(f)|^2\, df} \tag{2.8}$$

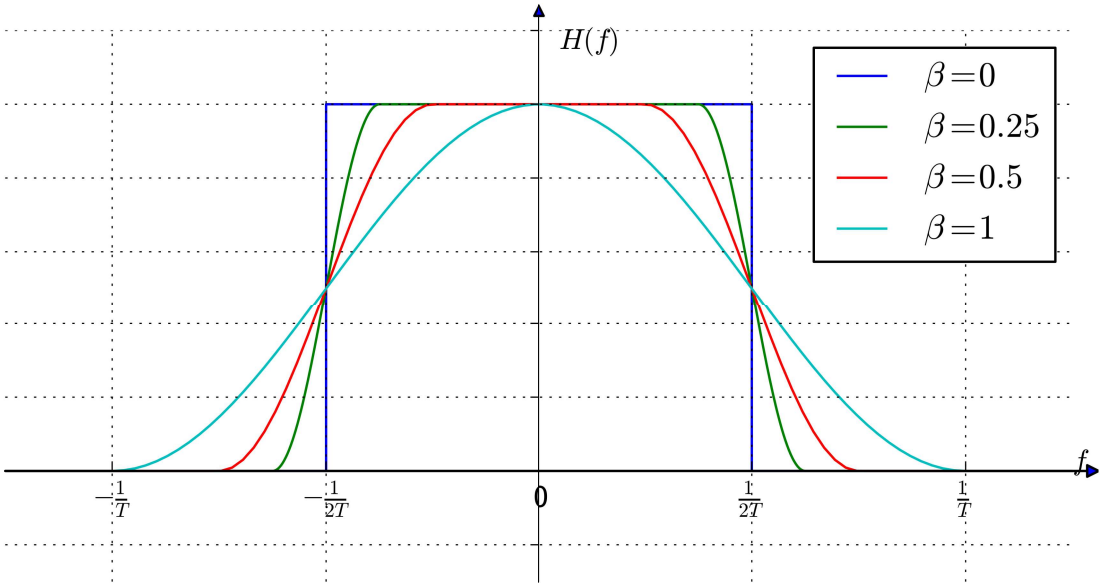


Figure 2.4: Raised cosine filter with varying β, $H(f) = |G(f)|^2$ as stated in Equation (2.9)

defined as a constant which depends on the pulse shape. For our use case the pulse shape $|G(f)|^2$ is defined as a raised cosine function

$$|G(f)|^2 = \begin{cases} 1, & |f| \leq \frac{1-\beta}{2T} \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right], & \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \\ 0, & else \end{cases} \tag{2.9}$$

with pulse shaping parameter $\beta \in [0, 1]$. The function for different β is depicted in Figure 2.4. The resulting ϵ is derived in the Appendix in Section 7.1 and the result in Equation (7.9) is stated as

$$\epsilon(\beta) = \frac{1}{12} + \beta^2\left(\frac{1}{4} - \frac{2}{\pi^2}\right)$$

which also matches the equation stated in [D'A15].

Summarizing the results from Equation (2.7) and Equation (7.9) yields the MCRB for the timing error and therefore for any timing error estimator as

$$MCRB(\tau) = \frac{1}{2L}\frac{1}{E_s/N_0}\frac{T^2}{4\pi^2\epsilon}$$

with ϵ defined as

$$\epsilon(\beta) = \frac{1}{12} + \beta^2 \left(\frac{1}{4} - \frac{2}{\pi^2} \right) \tilde{\in} [0.083, 0.131]$$

2.4 Timing error detection algorithms

Table 2.1: Timing error detection algorithms

	SPS	Input	PLL?
Gardner	2	Received symbol amplitude	Yes
Mueller and Müller	1	Received and decided symbol amplitude	Yes
Oerder and Meyr	4	Received symbol phase	No
Absolute Value Nonlinear	4	Received symbol phase	No
LogN	4	Received symbol phase	No
Lee	2	Received symbol phase	No

Table 2.1 shows different existing timing error detection algorithms. As stated in Section 2.1, the required SPS is a central design criteria, as the ADC sampling rates often are a bottleneck within high data rate receivers. Consequently, the lower the required SPS, the better. From the required SPS, Gardner, Mueller and Müller and Lee stand out the most. Although Mueller and Müller offers the best performance in terms of data, it has a severe limitation of needing the decided symbol amplitudes, therefore implicitly requiring the decided symbols [Con17]. As the proceeding logic of the timing error correction should not be influenced to allow more freedom in research, the Mueller and Müller is not further considered.

The two algorithms left, Gardner and Lee, differ in the timing recovery structure around them. The Gardner algorithm requires a phase-locked loop (PLL), which in a digital system may be realized by a numerically controlled oscillator (NCO) and an interpolator which can be implemented as a digital resampling filter. These components are put into a feedback configuration, which is illustrated in Figure 2.5. The feedback structure implies that the algorithm needs some cycles to obtain the correct timing error. In contrast to the Gardner algorithm, the Lee algorithm gives out a phase directly, which can be directly used in the interpolator. Therefore, it does not require a PLL thus it does not need time to synchronize itself to the received signal. In order to account for the Lee TED delay, the incoming samples need to be buffered until the detector is ready. Due to the feedforward nature of the Lee algorithm and therefore not needing a PLL, it is chosen as the algorithm to be further evaluated within this thesis.

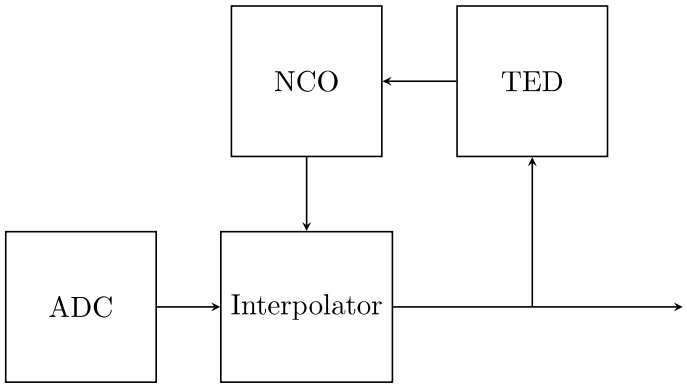


Figure 2.5: Timing recovery with a feedback structure, for example when using the Gardner TED algorithm

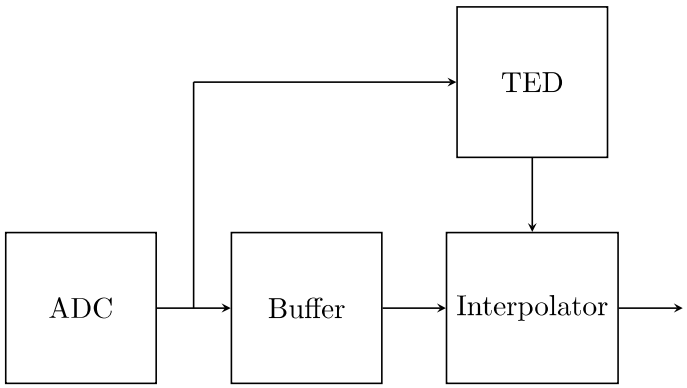


Figure 2.6: Timing recovery with a feedforward structure, for example when using the Lee TED algorithm

2.4.1 Lee timing error detector

The received complex signal $r(t)$ at time t is defined as

$$r(t) = x(t-\tau)e^{j\phi(t-\tau)} + n(t) \tag{2.10}$$

with the sent sample amplitude $x(t)$, signal delay τ, sample phase $\phi(t)$ and complex Gaussian random noise $n(t)$. Using these received samples, the Lee timing error estimator, which is operating at two samples per symbol, according to [Lee02] is defined as

$$\frac{\hat{\tau}}{T} = \frac{1}{2\pi} \arg\left\{ \sum_{k=1}^{2L} \left[|r(kT_s)|^2 e^{-jk\pi} + \mathrm{Re}\left[r(kT_s) r^*((k-1)T_s) \right] e^{-j(k-0.5)\pi} \right] \right\} \tag{2.11}$$

with the timing error estimate $\hat{\tau}$, sample duration $T_s = T/2$ and number of observed samples $2L + 1$.

Lee algorithm performance under varying signal-to-noise ratio (SNR) conditions is simulated by modeling an additive white Gaussian noise (AWGN) channel and implementing the algorithm in Python. The simulation results of the Lee TED show the normalized variances $\sigma^2 T^2$ plotted against different SNR levels for different input lengths L. The results are depicted in Figure 2.7 against the MCRB, which is discussed in more detail in Section 2.3. The modulation scheme used is quadrature phase-shift keying (QPSK) and the pulse shape is root-raised cosine with $\beta = 0.35$.

Potentially due to simulation inaccuracies in either the AWGN channel, the Lee algorithm, or the MCRB bounds, the graphs depicting the MCRB do not lie below the variance curves of the Lee algorithm for all SNR values. For lower SNR values the discrepancy can be explained due to the timing error being measured in samples and taken to modulo one. The error can never be larger than 0.5 and therefore the normalized variance is also limited to values below one. For the SNR range between -20 and 10, the bounds hold well. For high SNR values, the bounds do not hold for larger window size. Therefore, for SNR values above 10 dB, the results should be critically interpreted. As a suggestion for further work, these simulations should be verified for the higher SNR values.

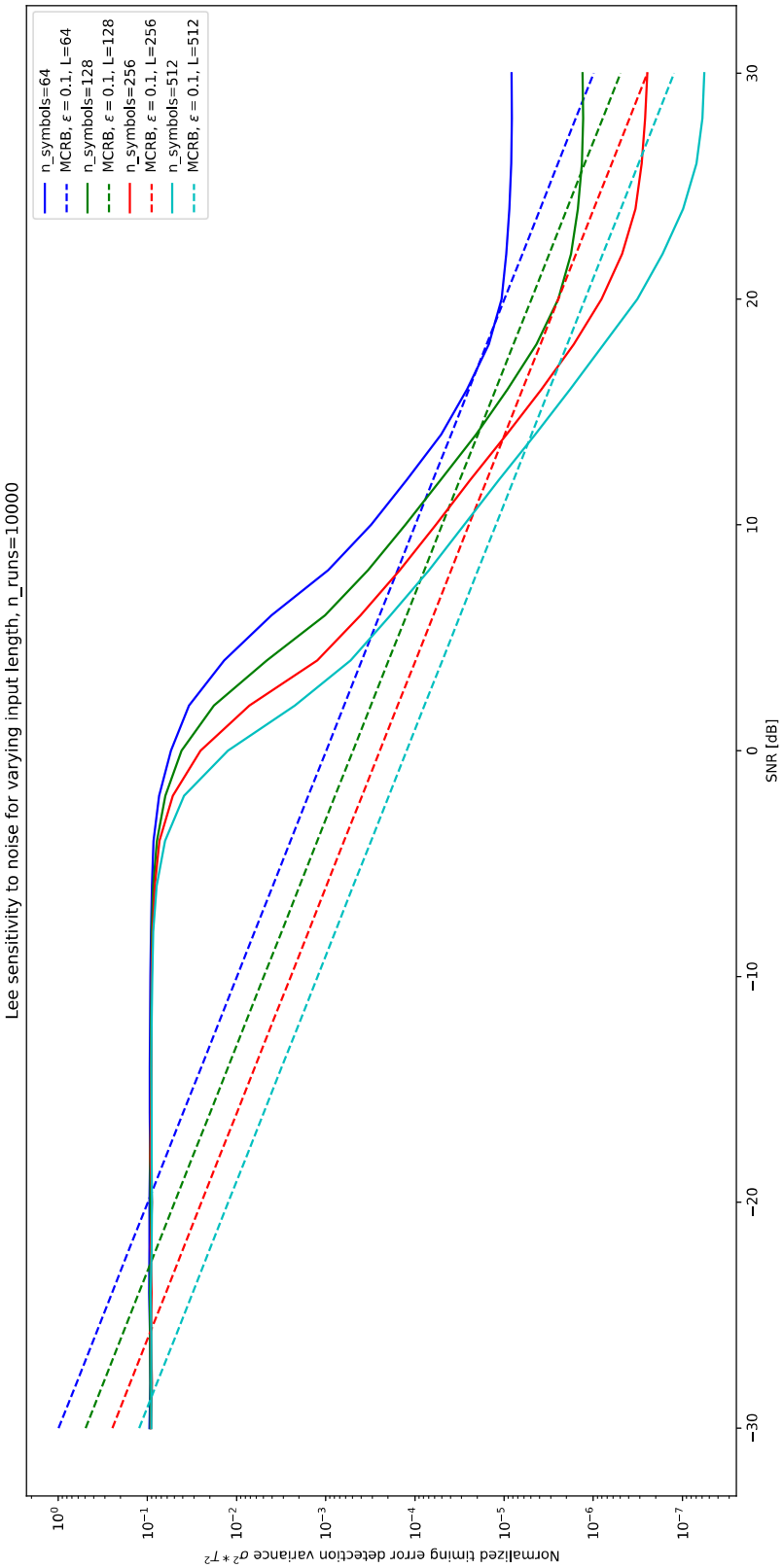


Figure 2.7: Lee noise sensitivity analysis against the MCRB

3 Kalman Filter

3.1 Motivation

As the timing recovery and by extension the timing error detection is run online under real-time constraints, the samples are received one after another, therefore iterative estimation is required. A Kalman filter is an estimator that takes in measurements observed over time, which matches the application. It is the best possible linear estimator in the sense that it minimizes the mean square error. In addition, if the system model is linear and known apriori, the measurement noise zero mean, independent and jointly Gaussian, the Kalman filter is also the optimal estimator overall in the minimum mean square error sense [MYXF19].

3.2 Derivation

3.2.1 System model

The generic system model for a Kalman filter is given by

$$\mathbf{x}_\mathrm{k} = \mathbf{F}_\mathrm{k}\mathbf{x}_{\mathrm{k}-1} + \mathbf{B}_\mathrm{k}\mathbf{u}_{\mathrm{k}-1} + \mathbf{w}_\mathrm{k} \tag{3.1}$$

with state $\mathbf{x}_\mathrm{k}$, state transition model $\mathbf{F_k}$, control input model $\mathbf{B}_\mathrm{k}$ and process noise $\mathbf{w}_\mathrm{k}$. In the use case of timing recovery as part of a digital receiver, the Kalman filter is used to track the timing error and its change over time. This corresponds to the timing error model as discussed in Section 2.1. As the timing error is used for aligning the sampling time, tracking the timing error phase with respect to the sample period T is sufficient. Consequently the error always lies within $[-T/2, T/2)$ or equivalently, due to its periodicity $t + T = t$, within $[0, T)$ where T is the sample period. The Kalman filter therefore tracks the timing error phase, noted as θ. It is important that when tracking the timing error phase instead of the whole timing error with the Kalman filter, the implementation skips or repeats samples as the timing error phase wraps.
The change of the timing error over time can be thought of as the first derivative with respect to time of the timing error phase, or simply put, the timing error frequency. The

timing error frequency is denoted as f and defined as $f = \frac{\partial \theta}{\partial t}$. The timing error frequency can be both positive and negative. A positive value corresponds to the ADC sampling slower than the sender, a negative value to the ADC sampling faster than the sender. Its absolute value is assumed to be small, within the range of 0.1 ppm to 100 ppm which should cover typical sources of error such as imperfect clock sources e.g. quartz crystals, temperature drift, as well as Doppler shift. Combining the timing error phase θ and the timing error frequency f, the state of the Kalman filter is obtained and denoted as

$$\mathbf{x}_\mathrm{k} = \begin{bmatrix} \theta_\mathrm{k} \\ f_\mathrm{k} \end{bmatrix} \tag{3.2}$$

Furthermore, a state transition is the change in frequency and phase transition model between the previous and current step. It is given by

$$\mathbf{F}_\mathrm{k} = \mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

which implies that the next frequency is equal to the frequency in the previous step, or in other words it stays constant. The matrix also implies that the phase depends is the previous phase of the system plus the change due to frequency. This makes sense physically, as with a constant frequency the phase changes linearly. Put into a differential form, the frequency is the first derivative of the phase w.r.t. time: $\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = f$. This formulation also shows that the system model does not change over time and therefore the dependency on the current step k is omitted.

As there is no control input, the control input model $\mathbf{B} = \mathbf{0}$ and can be left out.

The system noise $\mathbf{w}_\mathrm{k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_\mathrm{k})$ statistics do not change overtime, and therefore the covariance matrix of the process noise $\mathbf{Q}_\mathrm{k}$ stays constant

$$\mathbf{Q}_\mathrm{k} = \mathbf{Q} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_f^2 \end{bmatrix} \tag{3.4}$$

Consequently, $\mathbf{w}_\mathrm{k} = \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. The system model assumes that the temperature drift, over minutes or hours, and degradation for example due to radiation, over months or years, are modelled by the random walk of the timing error frequency. The system model is thus given by

$$\mathbf{x}_\mathrm{k} = \mathbf{F}\mathbf{x}_{\mathrm{k}-1} + \mathbf{w}_\mathrm{k} \tag{3.5}$$

3.2.2 Observation model

The generic observation model for the Kalman filter is given by

$$\mathbf{z}_\mathrm{k} = \mathbf{H}_\mathrm{k}\mathbf{x}_\mathrm{k} + \mathbf{v}_\mathrm{k} \tag{3.6}$$

with observation model $\mathbf{H}_\mathrm{k}$ and zero mean Gaussian observation noise $\mathbf{v}_\mathrm{k} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_\mathrm{k})$.

As only the phase is observed by the Lee timing error detector, the observation $\mathbf{z}_\mathrm{k}$ and its noise $\mathbf{v}_\mathrm{k}$ become scalar and the covariance matrix $\mathbf{R}_\mathrm{k}$ reduces to scalar variance $\sigma^2_{z,k}$.

$$\mathbf{z}_\mathrm{k} = z_\mathrm{k}, \quad \mathbf{v}_\mathrm{k} = v_\mathrm{k}, \quad \mathbf{R}_\mathrm{k} = \sigma^2_{z,k} \tag{3.7}$$

with $v_\mathrm{k} \sim \mathcal{N}(0, \sigma^2_{z,k})$. The observation matrix is constant $\mathbf{H}$ and simplifies to a row vector

$$\mathbf{H}_\mathrm{k} = \mathbf{H} = \mathbf{h} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.8}$$

This leads to the simplified observation model

$$z_\mathrm{k} = \mathbf{h}\mathbf{x}_\mathrm{k} + v_\mathrm{k} = \theta_k + v_\mathrm{k} \tag{3.9}$$

Furthermore, the observation noise is time-variant as the channel conditions change. This is modelled by keeping v_k dependent on k. In the real system, this observation noise will depend on the SNR.

$$v_\mathrm{k} := v(\mathrm{SNR}_\mathrm{k})$$

In the following sections, the steps of a single update Kalman filter are described and specialized for the specific case of timing error estimation.

3.2.3 A priori state estimate

The state is first predicted as follows

$$\hat{\mathbf{x}}_{\mathrm{k}|\mathrm{k}-1} = \mathbf{F}_\mathrm{k}\hat{\mathbf{x}}_{\mathrm{k}-1|\mathrm{k}-1} + \mathbf{B}_\mathrm{k}\mathbf{u}_{\mathrm{k}-1} \tag{3.10}$$

Again, as there is no external control, $\mathbf{B} = \mathbf{0}$. In addition, with the simplified state transition model from Equation (3.3) and the equation simplifies to

$$\hat{\mathbf{x}}_{\mathrm{k}|\mathrm{k}-1} = \mathbf{F}\hat{\mathbf{x}}_{\mathrm{k}-1|\mathrm{k}-1} \tag{3.11}$$

3.2.4 A priori covariance estimate

The Kalman filter approach first makes a prediction called the a priori state estimate using only information from the previous step.

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{F}_{\mathrm{k}}\mathbf{P}_{\mathrm{k-1|k-1}}\mathbf{F}_{\mathrm{k}}^{\top} + \mathbf{Q}_{\mathrm{k-1}} \tag{3.12}$$

which with Equation (3.3) and Equation (3.4) simplifies to

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{F}\mathbf{P}_{\mathrm{k-1|k-1}}\mathbf{F}^{\top} + \mathbf{Q} \tag{3.13}$$

3.2.5 Innovation

The innovation, which represents the error between the predicted and observed value, is given in its generic form as

$$\tilde{\mathbf{y}}_{\mathrm{k}} = \mathbf{z}_{\mathrm{k}} - \mathbf{H}_{\mathrm{k}}\hat{\mathbf{x}}_{\mathrm{k|k-1}} \tag{3.14}$$

the observation is scalar according to Equation (3.7). When in addition using the simplified observation matrix from Equation (3.8), the equation simplifies to a scalar form

$$\tilde{y}_{\mathrm{k}} = z_{\mathrm{k}} - \mathbf{h}\hat{\mathbf{x}}_{\mathrm{k|k-1}} \tag{3.15}$$

which when plugging in the a priori state estimate from Equation (3.11) yields

$$\begin{aligned}
\tilde{y}_{\mathrm{k}} &= z_{\mathrm{k}} - \mathbf{h}\mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} \\
&= z_{\mathrm{k}} - \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta_{\mathrm{k-1|k-1}}\\ f_{\mathrm{k-1|k-1}}\end{bmatrix} = z_{\mathrm{k}} - \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}\theta_{\mathrm{k-1|k-1}}\\ f_{\mathrm{k-1|k-1}}\end{bmatrix} \\
\tilde{y}_{\mathrm{k}} &= \underbrace{z_{\mathrm{k}}}_{\mathrm{TED_k}} - (\theta_{\mathrm{k-1|k-1}} + f_{\mathrm{k-1|k-1}})
\end{aligned} \tag{3.16}$$

For the observation, z_{k} any TED can be used. The choice of the TED is discussed in a later chapter.

3.2.6 Innovation covariance

$$\mathbf{S}_{\mathrm{k}} = \mathbf{H}_{\mathrm{k}}\mathbf{P}_{\mathrm{k|k-1}}\mathbf{H}_{\mathrm{k}}^{\top} + \mathbf{R}_{\mathrm{k}} \tag{3.17}$$

with Equation (3.8), Equation (3.7) and using notation

$$\mathbf{P}_{\mathrm{k|k-1}} := \begin{bmatrix}p_{\mathrm{k|k-1,00}} & p_{\mathrm{k|k-1,01}}\\ p_{\mathrm{k|k-1,10}} & p_{\mathrm{k|k-1,11}}\end{bmatrix}$$

the equation can be simplified

$$s_{\mathrm{k}} = \mathbf{h}\mathbf{P}_{\mathrm{k|k-1}}\mathbf{h}^{\top} + \sigma_{z,k}^2 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_{\mathrm{k|k-1,00}} & p_{\mathrm{k|k-1,01}} \\ p_{\mathrm{k|k-1,10}} & p_{\mathrm{k|k-1,11}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \sigma_{z,k}^2$$

$$= \begin{bmatrix} p_{\mathrm{k|k-1,00}} & p_{\mathrm{k|k-1,01}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \sigma_{z,k}^2$$

which finally yields the simplified innovation covariance, or simply innovation variance

$$s_{\mathrm{k}} = p_{\mathrm{k|k-1,00}} + \sigma_{z,k}^2 \tag{3.18}$$

3.2.7 Kalman gain

$$\mathbf{K}_{\mathrm{k}} = \mathbf{P}_{\mathrm{k|k-1}}\mathbf{H}_{\mathrm{k}}^{\top}\mathbf{S}_{\mathrm{k}}^{-1} \tag{3.19}$$

which simplifies with Equation (3.8) and Equation (3.16) to

$$\mathbf{k}_{\mathrm{k}} = \mathbf{P}_{\mathrm{k|k-1}}\mathbf{h}^{\top}s_{\mathrm{k}}^{-1}$$

Using

$$\mathbf{P}_{\mathrm{k|k-1}} := \begin{bmatrix} p_{\mathrm{k|k-1,00}} & p_{\mathrm{k|k-1,01}} \\ p_{\mathrm{k|k-1,10}} & p_{\mathrm{k|k-1,11}} \end{bmatrix}$$

gives

$$\mathbf{k}_{\mathrm{k}} = \begin{bmatrix} p_{\mathrm{k|k-1,00}} & p_{\mathrm{k|k-1,01}} \\ p_{\mathrm{k|k-1,10}} & p_{\mathrm{k|k-1,11}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} s_{\mathrm{k}}^{-1}$$

and thus

$$\mathbf{k}_{\mathrm{k}} = \begin{bmatrix} p_{\mathrm{k|k-1,00}} \\ p_{\mathrm{k|k-1,10}} \end{bmatrix} s_{\mathrm{k}}^{-1} \tag{3.20}$$

3.2.8 State update

This estimate is then updated based on a new scalar measurement $\tilde{y}_{\mathrm{k}}$ yielding the a posteriori state estimate. In our case there is no external control, thus

$$\hat{\mathbf{x}}_{\mathrm{k|k}} = \mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} + \mathbf{K}_{\mathrm{k}}\tilde{\mathbf{y}}_{\mathrm{k}} \tag{3.21}$$

which when using the Kalman gain to Equation (3.20) and the innovation from Equation (3.16) reduces to

$$\hat{\mathbf{x}}_{\mathrm{k|k}} = \mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} + \mathbf{k}_{\mathrm{k}}\tilde{y}_{\mathrm{k}} \tag{3.22}$$

3.2.9 A posteriori covariance estimate

After the update, the covariance estimation is updated

$$\mathbf{P}_{\mathrm{k|k}} = (\mathbf{I} - \mathbf{k}_{\mathrm{k}}\mathbf{h}_{\mathrm{k}})\,\mathbf{P}_{\mathrm{k|k-1}}$$

as $\mathbf{h} = \mathbf{h}_{\mathrm{k}}$, this simplifies to

$$\mathbf{P}_{\mathrm{k|k}} = (\mathbf{I} - \mathbf{k}_{\mathrm{k}}\mathbf{h})\,\mathbf{P}_{\mathrm{k|k-1}} \tag{3.23}$$

3.2.10 Summary

1. Predicted a priori estimate covariance

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{F}\mathbf{P}_{\mathrm{k-1|k-1}}\mathbf{F}^{\top} + \mathbf{Q} \tag{3.24}$$

with constant state transition model $\mathbf{F}$, constant system noise covariance matrix $\mathbf{Q}$ and the a posteriori covariance matrix estimate $\mathbf{P}_{\mathrm{k-1|k-1}}$.

$$\mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\ \mathbf{Q} = \begin{bmatrix} \sigma_{\theta}^2 & 0 \\ 0 & \sigma_{f}^2 \end{bmatrix}$$

2. Innovation

$$\tilde{y}_{\mathrm{k}} = z_{\mathrm{k}} - \mathbf{h}\mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} \tag{3.25}$$

with constant observation matrix $\mathbf{h} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and previously estimated state $\hat{\mathbf{x}}_{\mathrm{k-1|k-1}}$ with its initial state $\hat{\mathbf{x}}_{0|0} = \begin{bmatrix} \theta_0 \\ f_0 \end{bmatrix}$

3. Innovation covariance

$$s_{\mathrm{k}} = \mathbf{h}\mathbf{P}_{\mathrm{k|k-1}}\mathbf{h}^{\top} + \sigma_{z,k}^2 \tag{3.26}$$

with the observation noise variance $\sigma_{z,k}^2$ that depends on the SNR.

4. Kalman gain

$$\mathbf{k}_{\mathrm{k}} = \mathbf{P}_{\mathrm{k|k-1}}\mathbf{h}^{\top}s_{\mathrm{k}}^{-1} \tag{3.27}$$

5. State update

$$\hat{\mathbf{x}}_{\mathrm{k|k}} = \mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} + \mathbf{k}_{\mathrm{k}}\tilde{y}_{\mathrm{k}} \tag{3.28}$$

6. The a posteriori estimate covariance

$$\mathbf{P}_{\mathrm{k|k}} = (\mathbf{I} - \mathbf{k}_{\mathrm{k}}\mathbf{h}_{k})\mathbf{P}_{\mathrm{k|k-1}} \tag{3.29}$$

3.2.11 Initial state

For the initial state, there is no information about the timing error at the receiver, which could lie anywhere within the interval $[0, T)$. For this reason, the initial timing error distribution is assumed to be uniformly distributed between $[0, T)$ with variance $T^2/12$. The initial timing error is arbitrarily chosen to be 0, however any other value between $[0, T)$ also suffices.
The initial change in timing error over time, f, is unknown as well. The distribution of the initial value for the change in the timing error over time, f, should be zero mean and therefore assumed zero. The variance should lie within the constraints of the system, for example if the ADC runs at half the speed as it is supposed to, the Nyquist criterion would be violated. The intended operating range of the system is within range of tenths of parts per million up to parts per thousand. Larger deviations are assumed to be less common than small ones, and a Gaussian distribution is therefore chosen as a first approximation for the deviation.

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{P}_{0|0} = \begin{bmatrix} T^2/12 & 0 \\ 0 & T^2 f_{\Delta n}^2 \end{bmatrix} \tag{3.30}$$

where $f_{\Delta n}^2 = E[(f_\Delta / f_0)^2]$ is the normalized variance of the zero-mean distribution for the initial frequency difference f_Δ and $T^2/12$ is the variance of the uniform distribution of the initial timing error or the initial phase error of the system.

3.3 Simulation

The behaviour of the Kalman filter during fading can be seen in Figure 3.1. In this figure, the timing error is modelled by a simple linear equation and thought of as a phase and a frequency, as was similarly done in Section 2.1 in which the timing error is introduced as well as in Section 3.2.1, where the system model of the Kalman filter is derived. The true timing error phase is depicted as phase in Figure 3.1. It is measured in samples and has a range of [0, 1), due to it wrapping around. The true timing error is unknown to the receiver. Instead of seeing the true timing error, the receiver uses the TED to estimate the timing error. The quality of this estimate depends on the SNR, as shown in Figure 2.7 and can be modelled by an AWGN channel. The distorted phase, which in practice is the output of the Lee TED, is depicted in Figure 3.1 as the observed phase.
In the figure, the initial channel conditions are good, meaning that the SNR is high and consequently the estimation of the timing error by the TED is of good quality. The effect of the good SNR conditions can be seen in the low phase error of the Kalman filter

output when compared against the undistorted phase, which decreases over time. This is due to the more accurate synchronization of the Kalman filter as it converges towards the correct frequency and therefore the output gets more accurate. In addition, in this model it is assumed that the receiver knows that the channel is of good quality which increases the convergence speed. This can in practice be realized by a power detector which observes the strength of the input signal. This reading is converted to an SNR if the channel is understood well enough, which can then be mapped to an expected observation noise at the output of the Lee TED. One way to approach the mapping from SNR to the observation noise is to use the mapping as illustrated in Figure 2.7. After obtaining the observation noise, it is passed on to the Kalman filter.

After the initial period of good channel conditions, the conditions become worse. This is models a period of strong fading, which can be seen as the increased noise in the input signal. During this time, the phase error slowly increases, but still stays relatively low and importantly stays below the initial phase error, even though the fading lasts for a long time that is longer than the acquisition period. The third phase contains shows that as soon as the conditions improve, the error starts decreasing again.

What this effectively means is that a) errors are smaller during fading and b) the reacquisition time is decreased compared to not using a Kalman filter. This is a central takeaway of this thesis, as it suggests that adding a Kalman filter does indeed provide a benefit for timing error detectors within context of receivers placed after channels that are prone to strong fading. The trade off is computational complexity, which in context of hardware, such as an FPGA, translates into higher resource, such as power and size, usage. More concrete implications for a generic hardware implementation are discussed in the following section whereas more detailed analysis for a FPGA implementation are covered in Chapter 5.

Analyzing Figure 3.2 yields further insights to the Kalman filter functionality, especially during low SNR. The top row of the figure describes the constant, undistorted frequency as well as the phase, which depicts a sawtooth like wave due to it wrapping around at its maximum value, which during this simulation is arbitrarily set to 100 in order to make the wrapping less frequent while keeping the system frequency at 1, increasing readability. Below the frequency on the left hand side, the number of phase loops counts up every time a phase wraps around. Below the phase loops is the observation noise variance. This is both used at the input of the Kalman filter as well as the variance of the noise added to the phase.

The resulting distorted phase can be seen in the second top most plot on the right, which is labeled as observed phase. This is the phase seen by the Kalman filter and the distortions by noise are clearly visible.

Below the observed phase are the errors calculated by the Kalman filter between the prediction and the observed phase. These errors are labeled in the figure as Kalman errors. Notably, the scale is ± 50 as in a modulo 100 numerical system the maximum distance between two numbers is 50. Using modulo calculus for calculating the errors is crucial, as otherwise the system suffers from strong non-linear jumps on phase wraps which lead to severe degradation of the Kalman filter performance.
In the row below the observation noise variance and Kalman errors are the Kalman gains for the phase and frequency, respectively. The Kalman gains are responsible for weighting the Kalman errors, in one extreme cases trusting the errors completely, in the other, ignoring them. In the beginning, it can be seen that the gains increase and, after reaching their maximum, start decreasing again. The Kalman gain for the phase reacts much faster and thus the increase and decrease look more like a spike. The response of the Kalman gain looks smoother.
Below the Kalman gains are the states that are tracked by the Kalman filter. On the left side is the phase that is tracked by the Kalman filter, on the right is the frequency. It can be seen that the Kalman phase has again the same sawtooth form as the undistorted phase. There are some observable distortions during the first 400 updates, but the phase becomes increasingly smoother. The Kalman frequency can be seen to get get closer and closer to one, matching the original system frequency.
To more closely observe the differences between the undistorted phase and frequency with the filtered phase and frequency, the absolute phase errors and the relative frequency errors are plotted on the bottom row. The phase errors are taken as absolute values, as the phase regularly becomes zero, which requires special handling when using relative errors to avoid division by zero. After about four hundred updates, the phase error stays below 1, which is 1 % of the maximum phase value. If the maximum phase was 360°, this would correspond to 3.6° of error. The frequency error converges to around 100×10^{-6}.
All in all, the figure shows that given enough samples and that the noise is zero-mean, the Kalman filter converges to the given system frequency after approximately four hundred updates, even though the observed signal is significantly distorted.

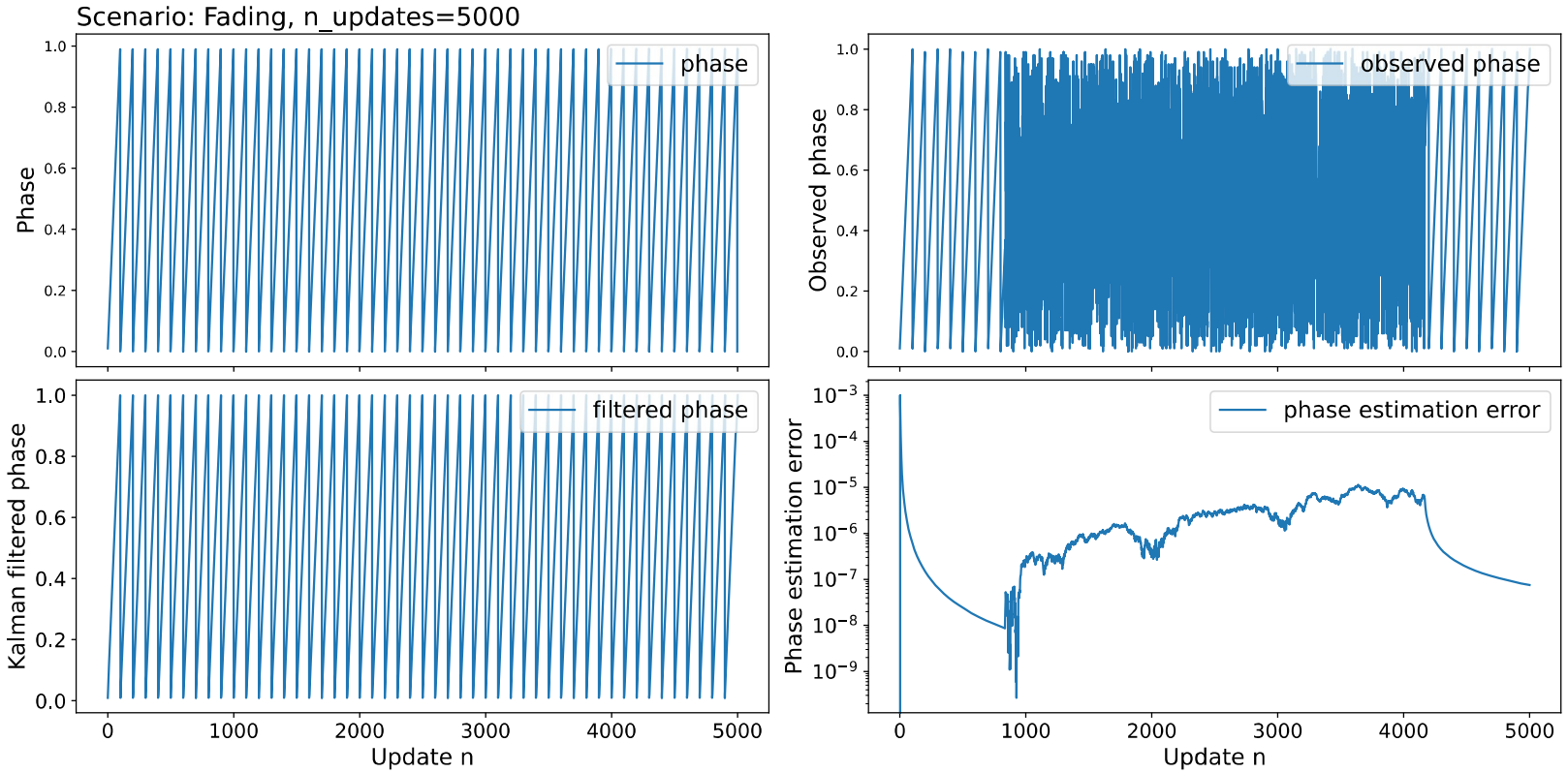


Figure 3.1: Python simulation of the phase of the timing error before and after filtering during fading. Top left: Phase of the timing error, or the timing error in modulo one, in samples. Top right: Observed timing error in samples, which is the timing error that has been distorted by an AWGN channel.

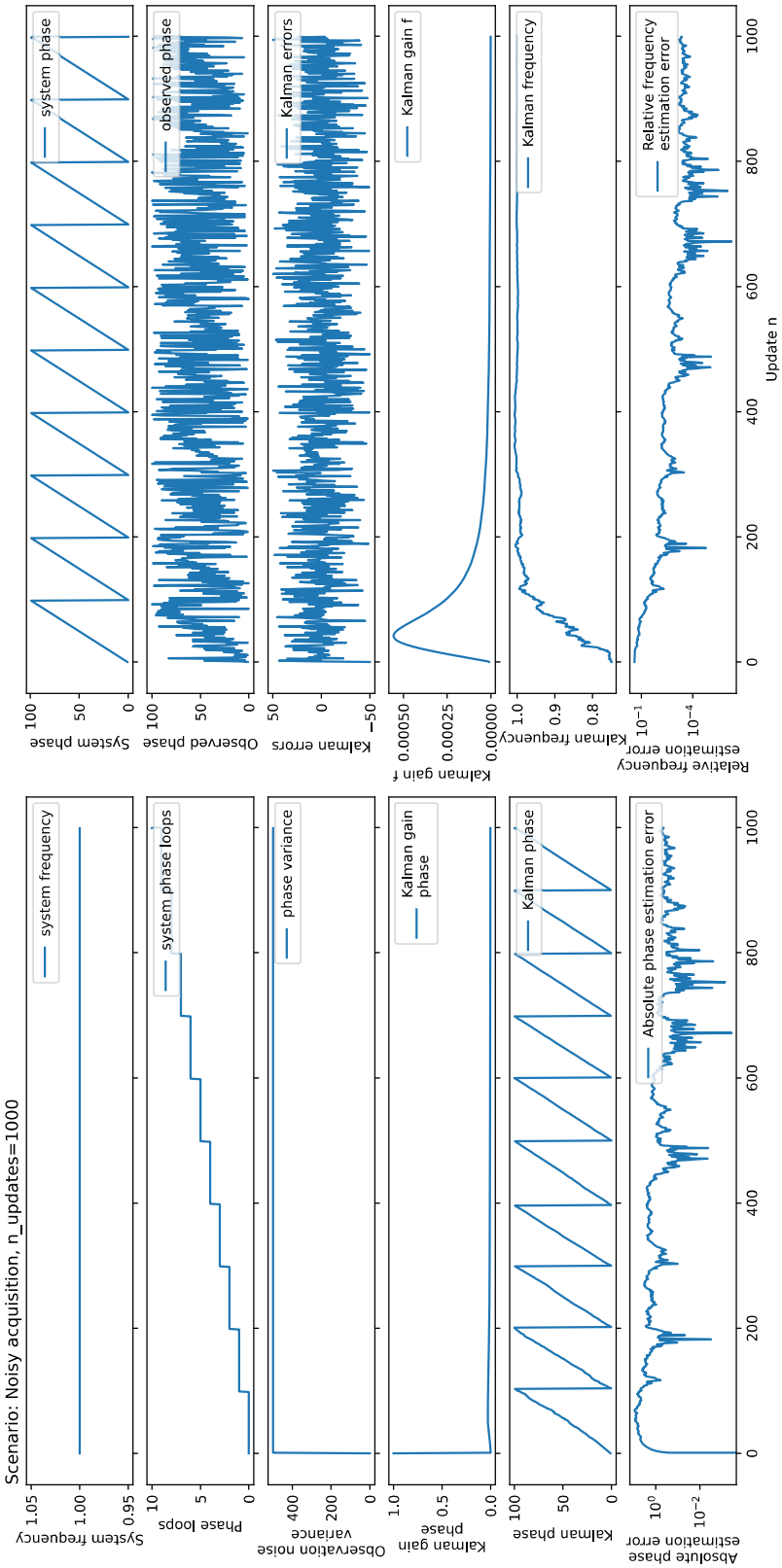


Figure 3.2: Kalman filter behaviour during high noise. The maximum value of the phase is set to 100.

4 System

The timing recovery system aims to detect the timing error and then apply the correction to the incoming samples.

4.1 Constraints

The system must fulfill the timing recovery under the following constraints which are motivated by the design criteria introduced in Section 2.1:

1. Each of two ADCs present on the FPGA used within this thesis produces 4×10^9 samples every second
2. The system is able to run continuously and process the samples in real-time
3. In case of distortions only caused by the sampling frequency offset, the bit-error-rate (BER) is 0 after acquisition
4. The filtered output is close to the Lee output during good channel conditions
5. The filtered output has a lower BER than the Lee algorithm during fading
6. The system handles ± 100 ppm frequency difference, 50 ppm coming from the Doppler shift and 50 from a pessimistic upper limit for the ADC clock offset

4.2 Overview

The system consists of three major parts

- Timing error detection
- Kalman filter
- Timing error correction via interpolation

The Kalman filter is designed to work as an add-on to the Lee TED. This means that the output of the Kalman filter should have an identical interface to that of the Lee algorithm to allow streamlined integration into the rest of the system. Moreover, values at the output of the filter should be identical to the Lee in case the SNR is high, meaning that the input signal quality is good and that the output of the Lee TED can be well trusted.

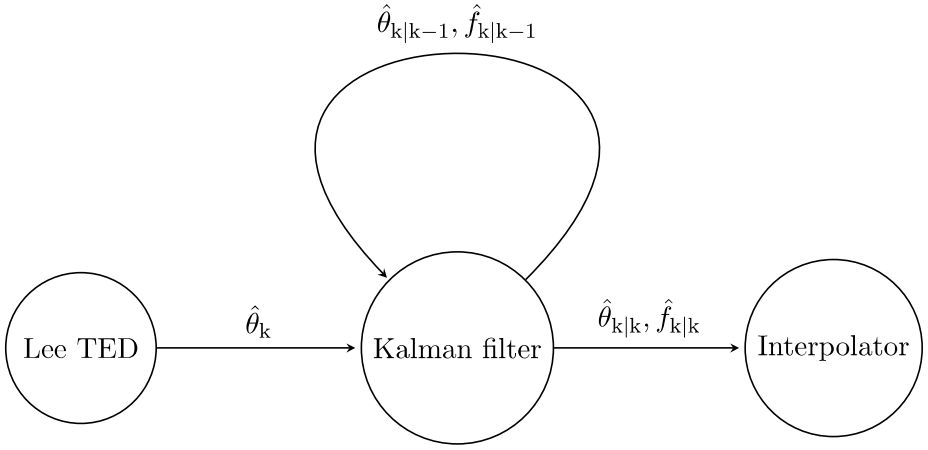


Figure 4.1: Timing recovery system with Kalman filter

The phase error typically changes over time. This often is due to a, potentially drifting, sampling frequency offset. Thus, when correcting a large amount of samples, the amount of correction also needs to change over time. One option is to apply the correction so often that the change of the timing error within the corrected window is negligible. The other possibility is to apply linear interpolation. This is done by taking the last, and therefore statistically most accurate timing error estimation of the Kalman filter and shifting it back in time according to the estimated change over time. In this thesis, the first mentioned approach is used due to its easier implementation. In the future, and especially for very high sampling frequency offsets due to which the timing error changes rapidly, it might be worth considering to switch to the linear interpolation approach. Another possibility to cope with faster changing timing errors is to do use a smaller number of samples for the Lee algorithm as well as updating of the Kalman filter more often, yielding to smaller time steps which in turn allow for larger changes as the change within a single time step is kept reasonably small.

5 Implementation

After obtaining an overview of the timing recovery system based on the Lee algorithm, Kalman filter and interpolation, the cost due to the implementation are analyzed in more detail within the following section. An FPGA in general offers an attractive platform to validate the implementation of the Kalman filter in an experimental setting due to its reconfigurability. Due its sophisticated parallel execution model and existing library support, VHDL is chosen as the language for the describing the hardware implementation of the Kalman filter for the FPGA. As a reference platform, a Xilinx UltraScale RFSoc FPGA is chosen, however the following analysis methodology is not limited to a specific hardware device.

5.1 Timing constraints

The ADC of the target FPGA runs at 4 GSPS. The interface of the ADC is designed such that on each clock cycle, 16 samples are delivered at once in parallel. In addition, the ADC provides the clock output signal which is fed into an asynchronous first in, first out (FIFO) inside the FPGA to allow transferring data from the ADC clock domain to the internal clock domain of the FPGA.
FPGA components can generally be designed using an asychronous or a synchronous design. Synchronous designs generally have the advantage of having timings that are simpler to model. As less complex systems are easier to verify and by extensions to make reliable, a synchronous design approach in chosen. The most straight forward approach is to run the FPGA design from a single clock to avoid additional clock synchronization apart from the necessary transformation between the ADC and the FPGA clock. Using a synchronous single clock design means that every operation within the FPGA logic must complete within a clock cycle. Especially computationally expensive operations such as divisions may be of concern. Notes on division are handled in a later section.
Using a synchronous single clock approach for the FPGA design, the FPGA needs to run at least at the ADC speed of 4 GSPS divided by 16, the number of samples taken in parallel into the FPGA per FPGA clock cycle, resulting in 4/16 GHz = 250 MHz clock speed. Adding a buffer to allow to react for slight deviations, a frequency of 330 MHz

is considered as the target goal for the Kalman filter, accounting for some buffer when integrating the Kalman filter into the rest of the system as the design tool will have less freedom to optimize timing of the circuits due to the FPGA design having to accomodate for more components.

5.2 Fixed-point arithmetic

For an online implementation running on an FPGA, numerical operations are more constrained. Floating point operations take longer and require more resources than integer operations. In addition, the values from the ADC are integers and would thus require conversion to floating-point numbers before further processing. Therefore, integer arithmetic is preferred over a floating-point implementation and consequently fixed-point arithmetic is chosen as the number format for the Kalman filter within the scope of this thesis. When doing fixed point arithmetic, careful consideration of the value ranges and bitwidths of numbers is needed. The VHDL 2008 language offers a fixed point package[1] to have an easier way of describing fixed-point numbers.

5.3 Lee

The implementation of the Lee algorithm is not part of the scope of this thesis. However, to show that the Kalman filter is capable of interfacing with the Lee algorithm, understanding the interfaces and timing of the Lee module is necessary. The algorithm is used twice, once for the x and once for the y component. For simplification, only the x component of the received signal is considered in this section. As the ADCs run at 4 GHz and the FPGA roughly at 250 MHz, each ADC outputs 16 samples per FPGA clock cycle. The resolution of the ADC is 12 bits, therefore yielding an interface to the Lee that takes in 16 I samples and 16 Q samples per clock cycle, with each sample having 12 bit resolution.

To be able to handle the throughput, the inputs are processed in parallel. As the Lee algorithm is a sum of multiplications, the multiplications are first done in parallel, and then summed up together using an adder tree.

The output of the specific Lee algorithm implementation within the scope of this thesis is within range of [-1, 1) samples and has a width of 10 bits.

[1] `https://freemodelfoundry.com/fphdl/Fixed_ug.pdf`

5.4 Buffer

The design uses buffers to implement skipping and repeating in addition to compensate for the TED delay. This means that shifts by one must be possible. However, shifting large amounts of data with high speed and granularity requires a lot of FPGA resources. Therefore, the tasks are split between two buffers of different sizes. The larger buffer is able to store hundreds or thousands of samples. However, it is only able to shift by 16 samples at a time. The larger buffer feeds into a smaller buffer, which while only being able to hold 16 samples, is able to shift by one sample at a time. The implementation uses a ring buffer for the larger one and a shift register for the smaller one.
The interface to the buffer requires signals to trigger the skipping and repeating logic. These signals are particularly relevant, because they must be triggered by the Kalman filter.

5.5 Kalman

As the Kalman filter evaluation and implementation is in the focus of this thesis, the different components of it are analyzed in more detail in the following sections in terms of numerical accuracy as well as estimating their computational complexity using the metric introduced in the following section.

5.5.1 Estimating computational complexity

A possible way to estimate more accurately how much computation different numerical operations take relative to each other is to compare the operations as if they were to run on a central processing unit (CPU). To this end, Table 5.1 lists how large the latencies are for different numerical operations executed on a typical CPU.

Table 5.1: Typical processor latencies

	CPU cycles
add	1
multiply	10
divide	70

5.5.2 State

As per Equation (3.2)

$$\mathbf{x}_\mathrm{k} = \begin{bmatrix} \theta_\mathrm{k} \\ f_\mathrm{k} \end{bmatrix}$$

where θ_k is the phase of the timing error and it is measured as a fraction of the sample period. This unit is chosen as it matches the output of the Lee algorithm implementation as introduced in Section 5.3. The output range of [-1, 1) samples of the Lee algorithm is linearly mapped to a convenient numerical range of

$$\theta_\mathrm{k} \in [0, 1) \tag{5.1}$$

avoiding signed integer numbers and being closer to the timing error model assumed in earlier Python simulations. The output of the Kalman filter is linearly mapped back to the range of [-1, 1) samples to preserve the original interface and to allow easier integration to the following components. The resolution of

$$\Delta\theta = 2^{-20} < 10^{-6} \tag{5.2}$$

is chosen in a way which allows for an efficient representation by extending the standard 18×18 bit signed integer multipliers from Xilinx by utilizing extension techniques as described in [Cha08]. Due to increasing relative errors when approaching $\Delta\theta$, the phase estimate of the Kalman filter is sufficiently accurate down to $100 \cdot \Delta\theta < 10^{-4}$, which is enough for the timing recovery even when including a magnitude of margin as the resolution of the interpolator input is 10 bits. Therefore the numerical inaccuracies of the Kalman filter timing error estimation are assumed to be negligible. The number of symbols per update determines what the frequency is scaled by, as in the end the phase change is frequency multiplied by the time between symbols. $\Delta\theta / N_\text{symbols-per-update}$ with

$$N_\text{symbols-per-update} \in [16, 1024] \tag{5.3}$$

The frequency f_k is in the range of

$$f \in [-100, 100] \text{ ppm} \tag{5.4}$$

as stated in the constraints in Chapter 4. The frequency has a resolution of

$$\Delta f = 2^{-10} \text{ ppm} < 1 \times 10^{-3} \text{ ppm} \tag{5.5}$$

however in practice the value is somewhat larger due to large relative numerical errors for numbers approaching 2^{-10} ppm. Allowing for some buffer, 0.1 ppm should be achievable using this resolution. Updating every $N_\text{symbols-per-update}$ symbols yields a maximum step size of

$$\Delta\theta_\text{max} = N_\text{symbols-per-update, max} \cdot f_\text{max} = 1024 \cdot 100 \times 10^{-6} = 0.124$$

Therefore, the given widths for the fields are sufficient to achieve 0.1 ppm frequency resolution with a maximum frequency compensation of ±100 ppm resulting to a maximum phase change per update of 0.124 or 12.4 % of the maximum normalized phase value.

5.5.3 A priori covariance estimation

From Equation (3.13)

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{F}\mathbf{P}_{\mathrm{k-1|k-1}}\mathbf{F}^{\top} + \mathbf{Q}$$

with convenience notation

$$\mathbf{P}_{\mathrm{k-1|k-1}} := \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

this yields

$$\mathbf{P}_{\mathrm{k|k-1}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \mathbf{Q} = \begin{bmatrix} A+C & B+D \\ C & D \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + \mathbf{Q}$$

$$= \begin{bmatrix} A+B+C+D & B+D \\ C+D & D \end{bmatrix} + \mathbf{Q}$$

and with $\mathbf{Q}$ as defined in Equation (3.4) this yields

$$\mathbf{P}_{\mathrm{k|k-1}} = \begin{bmatrix} A+B+C+D+\sigma_\theta^2 & B+D \\ C+D & D+\sigma_f^2 \end{bmatrix}$$

with the amount of operations stated in Table 5.2. To note is that $C + D$ needs to be only calculated once, therefore one operation is saved.

Table 5.2: Operations required for a priori covariance estimation

	add	multiply	divide
a priori covariance	6	0	0

5.5.4 Innovation

The simplified innovation is stated in Equation (3.16) as

$$\tilde{y}_{\mathrm{k}} = z_{\mathrm{k}} - (\theta_{\mathrm{k-1|k-1}} + f_{\mathrm{k-1|k-1}}) \tag{5.6}$$

where the predicted phase is the previous phase plus the change due to frequency. This is compared against the observed phase. To note is that here the time between updates

is assumed to be one symbol. In reality, this needs will by tens, hundreds or thousands of symbols. This frequency scaling leads to an additional multiplication.
As the phase is modulo 1, this leads to at most two further additions/subtractions as the input of the respective operation is assumed to be within the modulo range. The operations for prediction and innovation are listed in Table 5.3.

Table 5.3: Operations required for innovation

	add	multiply	divide
innovation	4	1	0

5.5.5 Innovation variance

As per Equation (3.18), the innovation variance is defined as

$$s_{\mathrm{k}} = p_{\mathrm{k|k-1},00} + \sigma_{z,k}^2$$

from which the single addition can directly be read

Table 5.4: Operations required for innovation variance

	add	multiply	divide
innovation variance	1	0	0

5.5.6 Kalman gain

Eq. Equation (3.20) defines the Kalman gain as

$$\mathbf{k}_{\mathrm{k}} = \begin{bmatrix} p_{\mathrm{k|k-1},00} \\ p_{\mathrm{k|k-1},10} \end{bmatrix} s_{\mathrm{k}}^{-1}$$

from which the required operations can directly be derived

Table 5.5: Operations required for Kalman gain

	add	multiply	divide
kalman gain	0	0	2

Division

The Kalman gain computation contains the numerically most expensive operation, the division. A first implementation that uses the default division operator '/' for fixed-point numbers, results into a maximum FPGA speed of 21 MHz as reported by the synthesis tool. This is an order of magnitude below the requirement. This could be due to the simulator not being able to pipeline the operation, resulting in a low maximum clock cycle.
To meet the requirements a simple binary division algorithm, the long division[2], is implemented. The long division algorithm consists of bitshifts, additions and subtractions and after first simulations runs at 400 MHz. The full algorithm as described in [AF09] is stated as follows

1. Align the most significant bit of the numerator with that of the denominator. The shifted numerator value is denoted as D. Set remainder R and quotient Q to zero and i to the number of bits shifted. T is a register large enough to hold the in-between calculation results.

2. $T = R - D$

   a) For $T \geq 0$, $Q(0) = 1$, $R = T$

   b) Else, $Q(0) = 0$

3. Shift Q by one bit to the right and D by one bit to the left. Subtract i by one.

4. If $i = -1$, Q is the resulting quotient and R is the resulting remainder.

However, the algorithm takes tens of cycles to complete due to its runtime of $2n$, where n is the numbers of bits in the numerator. For 20 bit wide numbers, the division delay results in approximately 40 cycles of delay using the long division algorithm implementation. In comparison, an addition and a multiplication of sufficiently small integers only takes a single cycle.
As part of future work, the division algorithm can at a later point be switched to a restoring, non-restoring or a Newton-Rhapson [PSC21] based division algorithm, which needs additional hardware resources due to multiplication [BK23]. A manufacturer specific hardware implementation may be used as well. Due to implementation complexity, these options are not further evaluated within the scope of this thesis.

[2] `https://en.wikipedia.org/wiki/Division_algorithm#Long_division`

5.5.7 State update

The simplified is earlier stated in Equation (3.22)

$$\hat{\mathbf{x}}_{\mathrm{k|k}} = \mathbf{F}\hat{\mathbf{x}}_{\mathrm{k-1|k-1}} + \mathbf{k}_{\mathrm{k}}\tilde{y}_{\mathrm{k}}$$

$$= \begin{bmatrix} \theta_{\mathrm{k-1|k-1}} + f_{\mathrm{k-1|k-1}} \\ f_{\mathrm{k-1|k-1}} \end{bmatrix} + \mathbf{k}_{\mathrm{k}}\tilde{y}_{\mathrm{k}}$$

Assuming the result from the prediction Equation (5.6) can be used, the number of operations amounts to

Table 5.6: Operations required for state update

	add	multiply	divide
state update	2	2	0

5.5.8 A posteriori covariance estimate

As per Equation (3.23)

$$\mathbf{P}_{\mathrm{k|k}} = (\mathbf{I} - \mathbf{k}_{\mathrm{k}}\mathbf{h})\,\mathbf{P}_{\mathrm{k|k-1}}$$

with

$$\mathbf{k}_{\mathrm{k}} := \begin{bmatrix} k_{\mathrm{k},0} \\ k_{\mathrm{k},1} \end{bmatrix}$$

$$= \left(\mathbf{I} - \begin{bmatrix} k_{\mathrm{k},0} \\ k_{\mathrm{k},1} \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix}\right) \mathbf{P}_{\mathrm{k|k-1}} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} k_{\mathrm{k},0} & 0 \\ 0 & k_{\mathrm{k},1} \end{bmatrix}\right) \mathbf{P}_{\mathrm{k|k-1}}$$

$$= \left(\begin{bmatrix} 1 - k_{\mathrm{k},0} & 0 \\ 0 & 1 - k_{\mathrm{k},1} \end{bmatrix}\right) \mathbf{P}_{\mathrm{k|k-1}}$$

with notation

$$\mathbf{P}_{\mathrm{k-1|k-1}} := \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

this can be rewritten as

$$\mathbf{P}_{\mathrm{k|k}} = \begin{bmatrix} 1 - k_{\mathrm{k},0} & 0 \\ 0 & 1 - k_{\mathrm{k},1} \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

and finally

$$\mathbf{P}_{\mathrm{k|k}} = \begin{bmatrix} (1 - k_{\mathrm{k},0})\, A & (1 - k_{\mathrm{k},0})\, B \\ (1 - k_{\mathrm{k},1})\, C & (1 - k_{\mathrm{k},1})\, D \end{bmatrix} \tag{5.7}$$

which yields the required operations as stated in Table 5.7.

Table 5.7: Operations required for a posteriori covariance estimate

	add	multiply	divide
a posteriori covariance	2	4	0

5.5.9 Summary

The total number of generic CPU cycles needed using mappings from Table 5.1 is listed in Table 5.8 and the computational graph is illustrated in Figure 5.1. The total number of cycles amounts to 189.

Table 5.8: Estimated number of operations required for the Kalman filter update using latencies from Table 5.1

	add	multiply	divide	cycles
a priori covariance	6	0	0	6
innovation	4	1	0	14
innovation variance	1	0	0	1
Kalman gain	0	0	2	140
state update	2	2	0	22
a posteriori covariance	2	4	0	42

5.5.10 Covariance limits

The a priori and a posteriori covariances are continuously updated. The covariance update step is integrative in nature, so there is potential of a rapid increase of the matrix elements. To better understand how fast the elements grow and potentially provide limits on the element value ranges, the change of these values is analyzed.

The element growth of the a posteriori covariance estimate, and thereby the a priori estimate, is determined by the Kalman gain. This allows determining the fastest and slowest rate of growth of the values by setting the extreme values, which are zero and one, for the respective Kalman gains.

For the first limit, Kalman gain is set to $\mathbf{k}_{\mathrm{k}} = \mathbf{k} = \mathbf{1}$. Thus, the a posteriori covariance

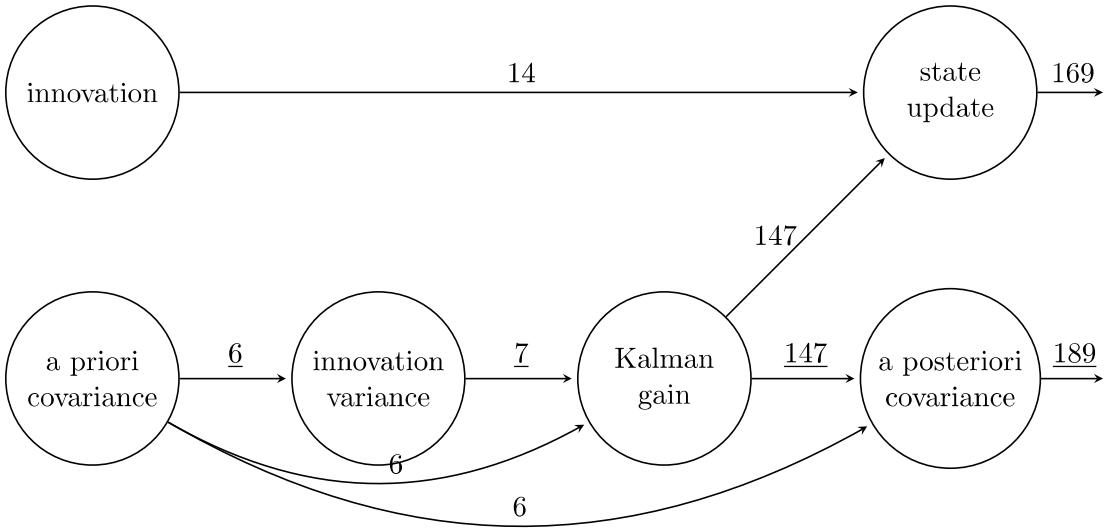


Figure 5.1: Computational graph, weights along the longest path are underlined

estimate as stated in Equation (3.23) becomes

$$\mathbf{P}_{\mathrm{k|k}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and consequently the a priori covariance from Equation (3.13) reduces to

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{0} + \mathbf{Q} = \mathbf{Q}.$$

This means that the matrix elements keep their initial values.

For the second special case, the Kalman gain is set to $\mathbf{k}_{\mathrm{k}} = \mathbf{0}$, the covariance matrix values grow the fastest. In addition, the a priori and a posteriori covariance estimates become equal.

$$\mathbf{P}_{\mathrm{k|k-1}} = \mathbf{P}_{\mathrm{k|k}}$$

Now using notation

$$\mathbf{P}_{\mathrm{k|k-1}} = \begin{bmatrix} p_{00,\mathrm{k|k-1}} & p_{01,\mathrm{k|k-1}} \\ p_{10,\mathrm{k|k-1}} & p_{11,\mathrm{k|k-1}} \end{bmatrix} := \begin{bmatrix} A_{\mathrm{k}} & B_{\mathrm{k}} \\ C_{\mathrm{k}} & D_{\mathrm{k}} \end{bmatrix}$$

$$= \begin{bmatrix} A_{\mathrm{k-1}} + B_{\mathrm{k-1}} + C_{\mathrm{k-1}} + D_{\mathrm{k-1}} + \sigma_\theta^2 & B_{\mathrm{k-1}} + D_{\mathrm{k-1}} \\ C_{\mathrm{k-1}} + D_{\mathrm{k-1}} & D_{\mathrm{k-1}} + \sigma_f^2 \end{bmatrix}$$

This allows for an easier analysis of the convergence of $\mathbf{P}_{\mathrm{k|k}}$.

The last element, D_{k} depends on its previous value so that $D_{\mathrm{k}} = D_{\mathrm{k-1}} + D_0$, $\mathrm{k} \in \mathbb{N}^+$, $D_0 = \sigma_f^2$. This results into a linear dependency of k

$$p_{11,\mathrm{k|k-1}} = D_{\mathrm{k}} = (k+1) \cdot \sigma_f^2 \tag{5.8}$$

The second and second last element are identical $B_{\mathrm{k}} = C_{\mathrm{k}}$ and equal to

$$\left\{ \frac{C_{\mathrm{k}}}{\sigma_f^2} \middle| \ \mathrm{k} \in \mathbb{N}_0 \right\} = \{0, 1, 3, 6, 10, 15, ...\}$$

corresponding to the triangular number sequence[3]. Therefore, it can be written in compact form as

$$p_{01,\mathrm{k|k-1}} = p_{10,\mathrm{k|k-1}} = B_{\mathrm{k}} = C_{\mathrm{k}} = \frac{k(k+1)}{2} \cdot \sigma_f^2 \tag{5.9}$$

Finally, A_{k} consists of the non-linear frequency contribution $A_{\mathrm{k},f}$ as well as the linear phase contribution $(k+1) \cdot \sigma_\theta^2$. As $A_{\mathrm{k},f} = B_{\mathrm{k-1}} + C_{\mathrm{k-1}} + D_{\mathrm{k-1}}$, it can be written as the sum of two tetrahedral number[4] series plus a triangular one. This can be written out as

$$\left\{ \frac{A_{\mathrm{k},f}}{\sigma_f^2} \middle| \ \mathrm{k} \in \mathbb{N}_0 \right\} = \{0, 1, 5, 14, 30, 55, ...\}$$

and the closed form corresponds to

$$A_{\mathrm{k},f} = \left[2 \cdot \frac{k(k-1)(k+1)}{6} + \frac{k(k+1)}{2} \right] \sigma_f^2$$

Combining the phase and frequency contributions yields

$$p_{00,\mathrm{k|k-1}} = A_{\mathrm{k}} = \left[2 \cdot \frac{k(k-1)(k+1)}{6} + \frac{k(k+1)}{2} \right] \sigma_f^2 + (k+1) \cdot \sigma_\theta^2 \tag{5.10}$$

The values for σ_θ^2 and σ_f^2 should be small, as it expected that the change in the timing error frequency is due to drift is slow and that the phase jitter of the sender is minimized. For the first implementation, it is assumed that $\sigma_\theta^2 = 2^{-10}$ and $\sigma_f^2 = 2^{-10}$ ppm^2.

$$\mathbf{Q} = \begin{bmatrix} 2^{-10} & 0 \\ 0 & 2^{-10} \end{bmatrix} \tag{5.11}$$

[3] https://en.wikipedia.org/wiki/Triangular_number
[4] https://en.wikipedia.org/wiki/Tetrahedral_number

The initial values for $\mathbf{P}_{\mathrm{k|k}}$ are given in Equation (3.30). Here, $T = 1$, and therefore

$$\mathbf{P}_{0|0} = \begin{bmatrix} 1/12 & 0 \\ 0 & f_{\Delta n}^2 \end{bmatrix}$$

which leaves the parameter $f_{\Delta n}^2$ to be determined. As $f_{\Delta n}^2 = E[(f_\Delta/f_0)^2] = Var[f_\Delta/f_0]$ due the distribution being zero mean. As a first estimation, the timing error frequency variance is set to a small value of 2^{-10} ppm$^2 = 1/1024$ ppm^2. For a Gaussian distribution this would mean a standard deviation of $2^{-5} = 1/32$ ppm. The initial a posteriori covariance matrix is thus given as

$$\mathbf{P}_{0|0} = \begin{bmatrix} 1/12 & 0 \\ 0 & 2^{-10} \end{bmatrix} \tag{5.12}$$

These values are to be refined after experimental verification.
In order to provide practical number ranges for the initial implementation, a number of assumptions are made. Firstly, the initial values are assumed as stated in Equation (5.12). In addition, it is assumed that 1000 update steps are enough for the frequency to converge and that the elements decrease afterwards. Under these assumptions, the values of the a priori covariance elements lie within the ranges as stated in Table 5.9. In the table, the column "min > 0" means the smallest non-zero value that needs to be represented. For the number of minimum fractional bits it is important to note that the actual numbers depends on the numerical error tolerated. The more bits are used, the lower the error. Visualization of the logarithmic value range can be seen in Figure 5.2, Figure 5.3 and Figure 5.5. Only the initial value is visible in Figure 5.3, as the values go to 0 after the first iteration. Therefore they are not visible on the logarithmic y-axis. For illustration, Figure 5.4

The values for the a posteriori elements are the same or lower as the a priori ones

Table 5.9: A priori covariance estimate ranges, k $\in [0, 1000]$

	min > 0	max	min fractional bits	min integer bits
$p_{00,\mathrm{k|k-1}}$	10^{-1}	10^{6}	4	20
$p_{01,\mathrm{k|k-1}}$	10^{-3}	10^{3}	10	10
$p_{10,\mathrm{k|k-1}}$	10^{-3}	10^{3}	10	10
$p_{11,\mathrm{k|k-1}}$	10^{-3}	10^{0}	10	1

according to Equation (5.7), therefore the same ranges can be used.

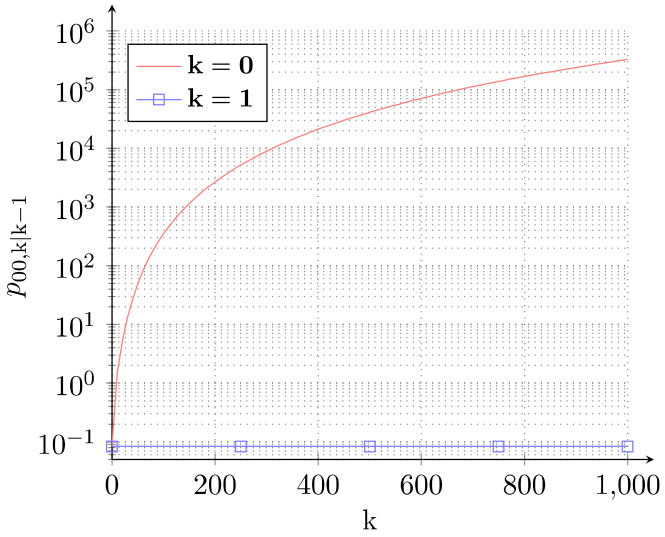


Figure 5.2: Covariance estimation limits, $p_{00,k|k-1}$

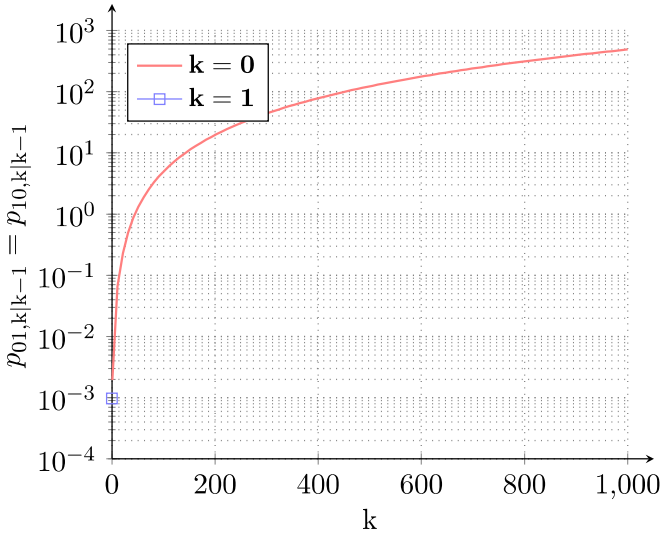


Figure 5.3: Covariance estimation limits, $p_{01,k|k-1} = p_{10,k|k-1}$

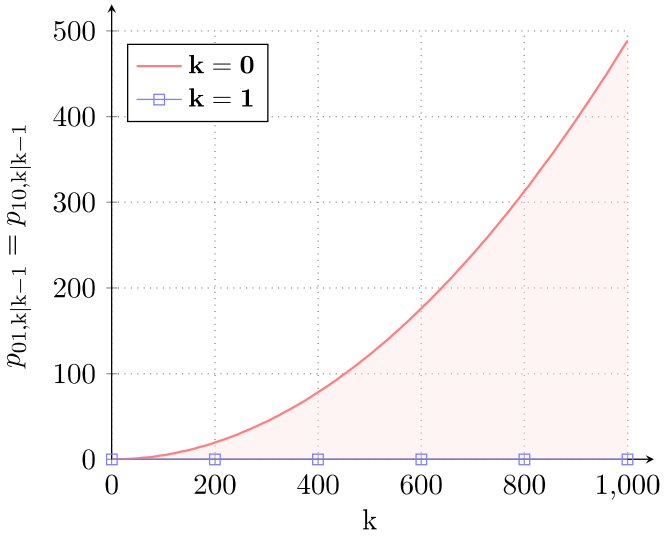


Figure 5.4: Covariance estimation limits, linear, $p_{01,k|k-1} = p_{10,k|k-1}$

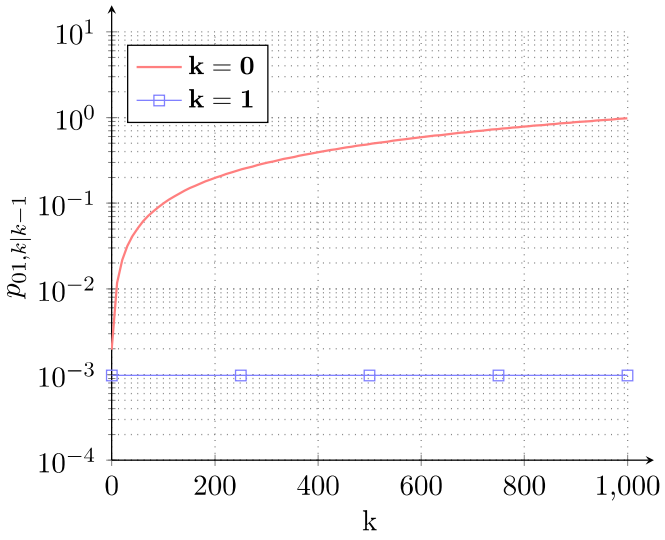


Figure 5.5: Covariance estimation limits, $p_{11,k|k-1}$

6 Results

6.1 Verification of VHDL implementation using GHDL

In order to test the components of the Kalman filter as well as the filter's functionality, the free open-source VHDL synthesis and simulation software GHDL[1] is used to together in combination with an open-source testing framework cocotb[2]. In practice, this allows writing testbenches in Python that test the VHDL code, utilizing the vast amout of Python libraries available as well as being easy to integrate to the existing Kalman filter Python simulations introduced earlier within this thesis. In addition, to later allow easy switching to other design and simulation software frameworks such as Vivado that are able to translate the VHDL description to an actual hardware platform, the free open-source VHDL package manager fusesoc[3] is utilized. As no license is required for any of these tools, this allows to run unit- and system level tests automatically in an easily reproducible continuous integration environment on code changes to prevent accidental breaking changes. Especially testing the interface of the Kalman filter is crucial in order to make the integration to the rest of the timing recovery between the Lee output and the interpolation input, to the full receiver and ultimately to the whole DSP system as streamlined as possible. In summary, this approach brings the VHDL development process closer to modern software development practices, increasing productivity especially with increasing system complexity. In the following, the most significant GHDL simulation results are summarized.

Figure 6.2 shows the input and output phase of the Kalman filter over 1000 updates. There is no noise added to the input signal, however the Kalman filter is set to believe that the input has a relatively large observation variance of 0.25, meaning that for the first tens of updates a small difference can be observed as the filter converges to the input frequency. Looking at Figure 6.3 reveals that the full convergence takes around 200 updates, after which the absolute value of the error between the input and output stays at around 10×10^{-6}, which is close to the numerical limits of 20 bits used by the implementation for tracking the phase.

[1] `https://ghdl.github.io/ghdl/about.html`
[2] `https://www.cocotb.org/`
[3] https://github.com/olofk/fusesoc

The design shows and the simulation results using GHDL verify that the required number of FPGA clock cycles for a single Kalman filter update is 50. The update includes the division and fits to the assumed 40 cycles in Section 5.5.6, as the integers used for the division are 20 bits wide. The duration for all the components are depicted in Figure 6.1. Notably, the the total number of cycles along the critical path reduces from 189 to 50 and the critical path goes through the state update, not through the a posteriori covariance when compared to 5.1.

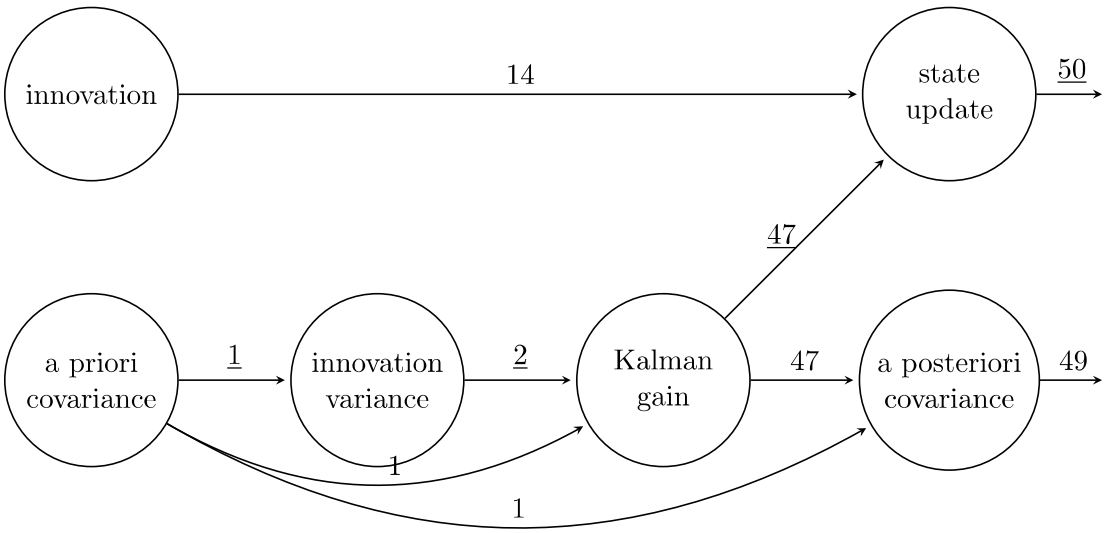


Figure 6.1: Computational graph for FPGA, weights along the longest path are underlined

6.2 FPGA resource usage and timing verification

After confirming the component level functionality, the FPGA resource usage is analyzed when the Kalman filter is integrated into the whole system. The system contains the digital sender and receiver on the same FPGA. The sender generates a random bit sequence, maps the bits to IQ symbols, shapes the pulses and interfaces to the DACs, which are integrated onto the same FPGA system-on-chip. The receiver includes interfacing with the on-chip ADCs, static equalization, polarization demultiplexing, timing recovery, carrier phase and frequency recovery, as well as bit and frame detection. Due to the matched-filter at the receiver degrading performance, it is not included. The reason for this could be that the pulses arriving have been distorted in some way and this they do not fit the matched filter. The first significant result is that the whole system passes

Table 6.1: FPGA resource utilization

	Total LUTs	DSP Blocks
System	110933 (26.08 %)	708 (16.57 %)
Receiver	81367 (19.13 %)	708 (16.57 %)
Timing recovery	32007 (7.53 %)	200 (4.68 %)
Kalman filter	658 (0.15 %)	8 (0.19 %)

the resource usage verification, meaning that the system does indeed fit onto the FPGA. The full resource utilization, both in absolute terms as well as relative to the total amount of available resources, can be seen in Table 7.1 and Table 7.2. The usage of the resource categories using the highest relative amount of resources are the look-up tables (LUTs) and the DSP blocks. Their usage is summarized in Table 6.1.
As part of the timing recovery, the Kalman filter uses $658/32007 \approx 2$ % of the LUTs and $8/200 \approx 4$ % of the DSP blocks. This means that not only does the Kalman filter fit onto the whole system, it also does not utilize a significant amount of resources even when compared against the utilized timing recovery resources. In other words, the cost of including a Kalman filter as part of a digital receiver are low.
When the synthesis tool, in this case Vivado, generates the bitstream, it verifies in addition to the resource constraints that the timing constraints are met. The final system containing the FPGA is able to be generated with the clock frequency constraint set to over 260 MHz, ultimately proving that it is possible to run a Kalman filter fast enough on an FPGA to be able to be utilized as part of a FSO DSP chain.

6.3 Experimental results using a fading testbench

To show that the Kalman filter integrates onto the whole FPGA logic and also provides an improvement to the Lee algorithm results in practice, a fading testbench is used. All in all, the setup is conceptually depicted in Figure 6.4. The QPSK signal is generated by the FPGA, which using the integrated DACs generates an x polarized signal with its corresponding I and Q components. The two signals from the DACs are connected to a Mach-Zehnder modulator (MZM) with an integrated laser. The MZM is connected via fiber to a fading testbench, which for the purpose of the test functions as a programmable attenuator taking in a power vector and is therefore marked as attenuator within Figure 6.4. Effectively, this allows to set periodic fades lasting up to several ten or hundred milliseconds. The fading testbench is further connected to an optical receiver outputting both polarizations, and its four outputs are connected to the same FPGA where the sender is located. As the fiber and components change the polarization, in order to

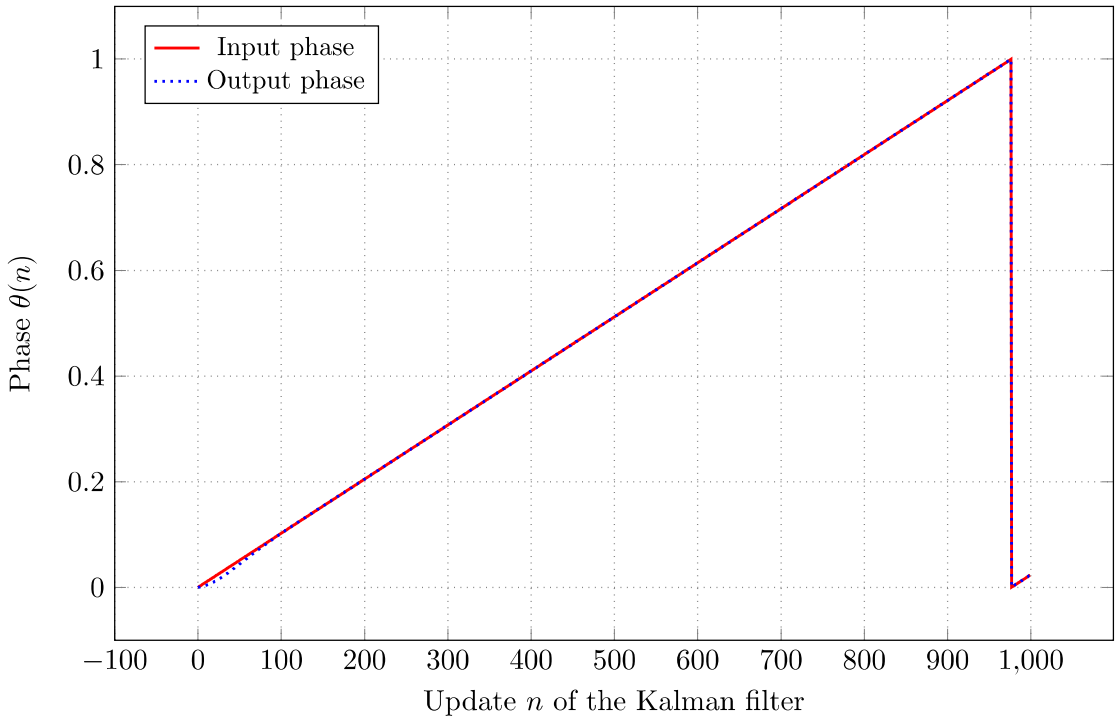


Figure 6.2: VHDL logic simulation showing the input and output phase of the Kalman filter. The phase, as expected, wraps around when reaching one. There is no noise, but observation variance is set to 1/4 resulting in slower convergence.

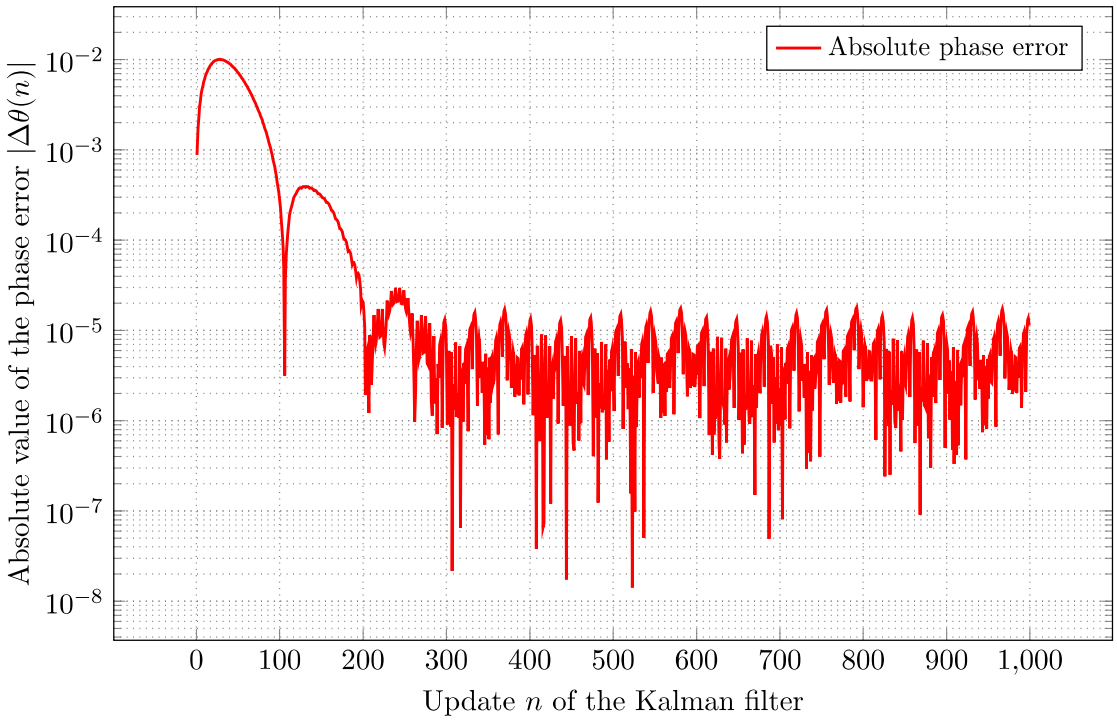


Figure 6.3: VHDL logic simulation showing the absolute value of the error between input and output phase of the Kalman filter. There is no noise, but observation variance is set to 1/4 resulting in slower convergence.

capture the changes, a dual-polarized receiver is used. The original, single polarization is recovered and therefore timing recovery results are only shown for one polarization.
The experiment is depicted in Figure 6.5. On the bottom center standing on the floor is the fading testbench. There is a screen connected to the fading testbench which is on top of the desk on the left side showing the currently programmed channel which for the purpose of the setup is set be a rectangular-shaped wave, creating a fade which lasts for 500 milliseconds and resulting in about three orders of magnitude less power being received at the ADC of the FPGA. On the top shelf on the right, there is a stack of two devices with eight cables running to and from the FPGA and two cables running to and from the fading testbench. The device on the bottom of the stack is the MZM, the top device is an optical receiver. The FPGA is placed in the middle of the desk. There are four cables coming from and to the FPGA, however only two of them are used to transmit information as only the signals x polarization is transmitted during the experiment. The FPGA board has a computer connected to it with a screen attached. The screen is placed on the right-hand side of the desk and showing the Vivado software used to create the bitstream for the FPGA as well as to program it. In addition, the computer is remote-controlled via a laptop which can be seen on top of the fading testbench. It runs interactive plots in Python which show a green-coloured IQ plot, a blue-colored BER plot as well as a red-coloured timing error plot. All the plots are updating in real-time and the data is fetched from the receiver running on the FPGA.
The interactive plots of the fading testbench and the computer connected to the FPGA further allow the adjustment and observation of the channel, receiver and sender parameters during in real-time. The exact SNR at the ADC is not available due to lacking time as well as measurement equipment during the testing, however as the SNR follows the power, the relative SNR difference is measured to be around 20 dB.
The results of the experiment are seen in the two screenshots of the computer connected to the FPGA, Figure 6.6 and Figure 6.7. The plot in the bottom right corner in both figures depicts the clock offset, which is the timing error in samples, over time. The plot in the middle on the right in turn depicts the BER over time. The timing error is observed to be close to zero as part of the experiment illustrated. However, setting the offset to another constant value by adjusting the sender delay digitally provides similar results. The timing error does not change over time, as this feature is not currently implemented within the scope of the experimental setup. In the future, this could be tested by utilizing an arbitrary waveform generator, by implementing the effect into the channel simulator, or by adding a digital implementation onto the FPGA.
The configuration of the observation variance differs between the two figures. The figures correspond to the two extreme cases of the observation variance for the Kalman filter.

The first configuration in Figure 6.6 shows a scenario where the observation variance of the Kalman filter is set to zero, its smallest value. Setting the observation variance to zero leads to the Kalman filter blindly following the timing error at its input and not relying on its internal prediction of the timing error. What this means in practice is that the output of the Lee TED is passed right through the Kalman filter. This allows to observe the performance of the Lee TED algorithm to provide a baseline to compare against when using the Kalman filter. This is done in Figure 6.7, where the observation variance is set close to 100×10^3, close to its maximum. In this case the input is largely ignored and the Kalman filter primarily relies on its internal prediction.

When looking at the clock offset in Figure 6.6, it can be seen that shortly after the start, the output starts to vary a lot, suggesting that the performance of the clock offset prediction is worse. This timepoint corresponds to a start of a fade generated by the fading testbench. In addition, when the clock offset becomes worse, at the same time the BER also goes up. The value of the BER is around 0.5. This means that the receiver is unable to receive any bits and suggests a total loss of synchronization. After the fade, the variance of the estimated timing error and the BER return back to their levels before the fade.

When in turn looking at the clock offset in Figure 6.7, it can be seen that after the start, the output starts to vary similarly to Figure 6.6. Again, this timepoint corresponds to a start of a fade generated by the fading testbench. However, this time the variance is not as strong. This implies that the performance of the clock offset prediction is still worse during a fade, but better than when not using the Kalman filter. At the same moment when the clock offset becomes worse, the BER also goes up. The value of the BER is notably below 0.5. This means that the receiver is still able to receiver at least some bits and means that the synchronization is not lost during the fade. After the fade, similarly to the case with zero observation variance, the variance of the estimated timing error as well as the BER return to their levels before the fade.

Comparing the two experiments suggests that using the Kalman filter improves the performance of the Lee algorithm during a fade while also recovering to the pre-fade BER levels after the fade.

Observing the IQ diagram deliveres additional insights. In both cases it has a slightly elliptic shape with the middle looking slightly different. This does suggest that the timing synchronization is at least in part working. However, as the IQ diagram collects samples over a longer duration, the samples may originate before, during and after the fade. In addition, the size of the opening is related to the number of samples captured. If the number goes to infinity, the figure would end up completely green. Thus, the size or lack of the opening is not directly related to the actual performance. However, the diagram

looks slightly tilted, which potentially could be due to the sub-optimal bias control of the MZM or some other systematic imperfection within the system likely unrelated to the timing recovery.
Finally, the setup is running continuously over several minutes. The only limiting factor is disk space of the computer connected to the FPGA, which if running the experiment over days will eventually fill up of measurement data collected from the FPGA. This suggests that the system is able to handle the high sampling rates of 4 GSPS per ADC.

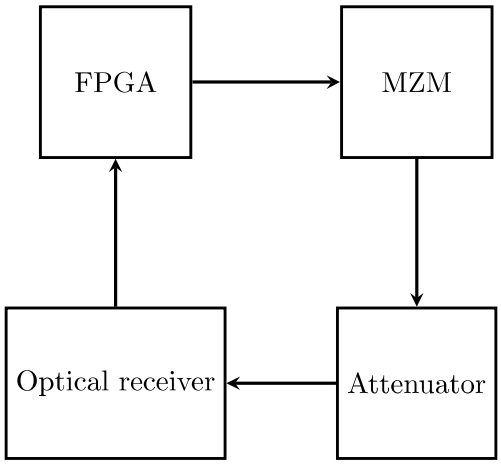


Figure 6.4: Conceptual diagram of the experimental setup. The fading testbench is depicted as an attenuator.

Figure 6.5: Experimental setup using a fading testbench

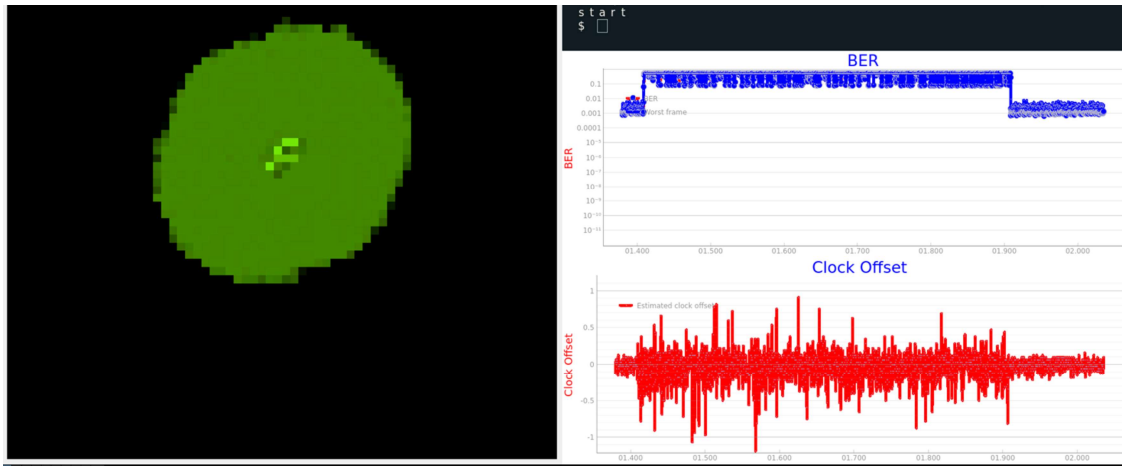


Figure 6.6: System performance during fading, observation variance=0, corresponds to trusting the output of the Lee algorithm completely.

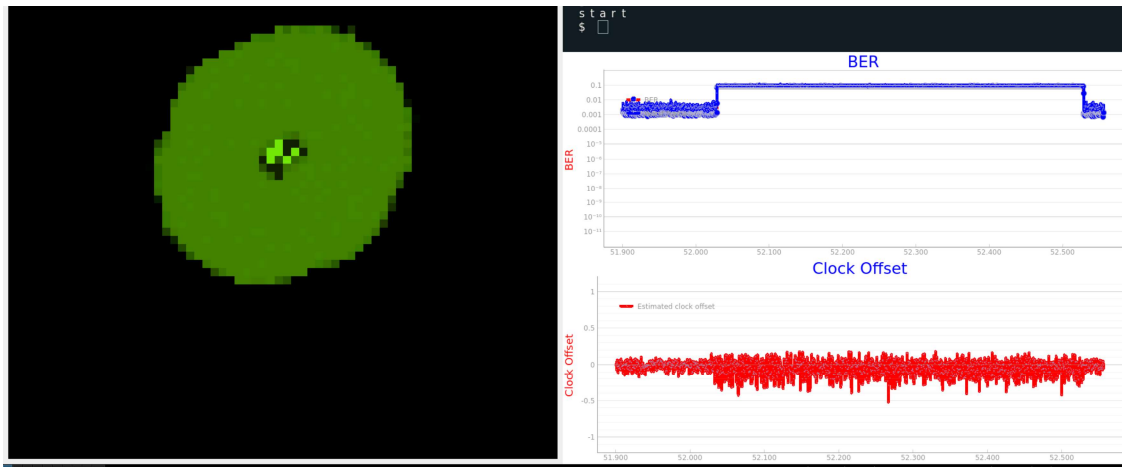


Figure 6.7: System performance during fading, observation variance=1e5, corresponds to the Kalman filter using its internal linear model to predict the phase while largely ignoring its input.

7 Conclusion and future outlook

7.1 Future outlook

This thesis shows that it is viable to use Kalman filters running on an FPGA as part of a digital receiver used for systems with high data rates. They improve estimation performance while not constraining the timing performance or resource usage compared to the rest of the system.
As the VHDL implementation of the Kalman filter was only a part of the thesis and not the full focus, the implementation quality of the Kalman filter can likely be improved. First of all, the convergence speed of the Kalman filter can be improved if the observation variance is optimally chosen. Second of all, the accuracy to which the filter converges can potentially be improved as well. Furthermore, by using a more sophisticated division algorithm, higher rates of change in the timing error can be achieved as the filter can be updated more frequently.
In terms of system level verification, the experimental setup so far only tests static offsets. It is suggested to experimentally confirm the timing recovery performance also when having the timing error changing over time. Furthermore, it is recommended that to more precisely quantify the timing recovery performance, optical SNR measurements before the optical receiver as well as electrical SNR measurements before the ADCs are conducted.
Finally, a way to improve future system integration and testing is to ensure that the system components, be it lasers, modulators or VHDL components on the FPGA are rigorously tested under controlled and reproducible environments before integration to prevent unexpected sources of errors during test campaigns.

7.2 Conclusion

This thesis provides a closer look at using a Kalman filter in combination with the Lee timing error detector as part of all-digital online timing recovery in context of FSO, especially DTE links.
One of the main motivations for FSO DTE is providing global broadband connectivity,

for which FSO DTE links are a promising technology offering higher throughput and security compared to classical RF links with demonstrated data rates up to 200 GBit/s. A central problem of FSO links DTE that effects the timing recovery algorithms, is fading of the optical signal due to atmospheric turbulence leading to signal level variation of several orders of magnitude with fade duration up to milliseconds. Fading must therefore be considered in the receiver design and the Kalman filter promises improvement during fading, utilizing information about the received power to weight the observations coming from a TED.

After introducing the concept timing error in receivers, this thesis provides an overview of digital timing recovery, showing that factors such as samples per symbol as well as computational complexity play an important role when choosing a TED algorithm. The theoretical limits of timing error estimation are briefly discussed as well. The final aspect of timing recovery is introducing interpolation as a way to digitally achieve the same effect as adjusting the sampling speed of an ADC in the analog domain.

The thesis continues by introducing the generic formulation of the Kalman filter and specializing it for linear timing error estimation. The filter is simulated in Python and the results show that the Kalman filter provides an improvement to timing error observations that are disturbed by Gaussian noise during and after fading as well as when when the system generally operates under low SNR conditions.

The Kalman filter is then put into context with its interfacing components, the Lee TED before the filter and the interpolator after the filter. Furthermore, the system level constraints are stated under which the timing recovery algorithm must operate when operating as part of an optical FSO receiver.

The system level considerations are followed by implementing the Kalman filter on an FPGA. First of all, it is observed that the FPGA is able to run fast enough to process IQ samples in real-time coming from ADCs running at 4 GSamples/s, while using the Kalman filter as part of the whole system. The Kalman filter does not provide a bottleneck for the speed. In addition, it is shown that the relative usage of DSPs blocks when compared to the total amount of DSPs blocks used for the timing recovery is 4 %, while the relative usage of LUTs is 2 % when compared against the full timing recovery. Furthermore, using an optical fading testbench, it shown that using the Lee algorithm with the Kalman filter improves the BER when compared against just using the Lee implementation without the Kalman filter.

Summarizing the key findings, the results show that a Kalman filter, integrated into a Lee algorithm-based all-digital online timing recovery chain running on a FPGA, does not constrain FPGA clock speed, uses only 4 % of the resources needed for timing recovery and provides an improvement in BER during fading when used to filter the Lee TED

output.
Therefore, this thesis concludes that it is beneficial to use a Kalman filter within the context of FSO DTE links for all-digital timing recovery when using the Lee algorithm for timing error detection. By adopting suggestions outlined in Section 7.1, the performance of the filter may be further improved. Using Kalman filters in FSO receivers is thus a promising step towards realising FSO based communication networks that also include DTE links.

Appendix A

7.1 MCRB constant derivation

7.1.1 Evaluation of denominator

Case 1:

$$|G(f)|^2 = 1, \quad |f| \leq \frac{1-\beta}{2T}$$

thus

$$\int_{-\infty}^{\infty} |G(f)|^2 \, df = \int_{-\frac{1-\beta}{2T}}^{\frac{1-\beta}{2T}} 1 \, df$$

as the integrand is symmetric, this can be written as

$$= 2 \cdot \int_{0}^{\frac{1-\beta}{2T}} 1 \, df = 2 \cdot \frac{1-\beta}{2T}$$

and finally

$$\int_{-\infty}^{\infty} |G(f)|^2 \, df = \frac{1-\beta}{T}, \quad |f| \leq \frac{1-\beta}{2T} \tag{7.1}$$

Case 2:

$$|G(f)|^2 = \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right]$$

$$\int_{-\infty}^{\infty} |G(f)|^2 \, df, \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T}$$

as symmetric

$$= 2 \cdot \int_{\frac{1-\beta}{2T}}^{\frac{1+\beta}{2T}} \frac{1}{2}\left[1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right)\right] \, df$$

$$= \int_{\frac{1-\beta}{2T}}^{\frac{1+\beta}{2T}} 1 + \cos\left(\frac{\pi T}{\beta}\left[|f| - \frac{1-\beta}{2T}\right]\right) \, df$$

with

$$a := \frac{1-\beta}{2T} \in \left[0, \frac{1}{2T}\right], \; b := \frac{1+\beta}{2T} \in \left[\frac{1}{2T}, \frac{2}{2T}\right], \quad c := \frac{\pi T}{\beta}$$

$$= \int_a^b 1 + \cos(c \cdot (|f| - a))\ df$$

due to

$$f \geq 0, \quad a \leq f \leq b \rightarrow |f| = f$$

together with splitting the integral yields

$$= (b-a) + \int_{f=a}^{f=b} \cos(c \cdot (f-a))\ df$$

substituting with

$$f' := f - a, \quad df' = df$$

$$= (b-a) + \int_{f'=0}^{f'=b-a} \cos\big(c \cdot f'\big)\ df'$$

$$= (b-a) + \frac{1}{c} \sin\big(c \cdot f'\big)\Big|_{f'=0}^{f'=b-a}$$

$$= (b-a) + \frac{1}{c} \sin(c \cdot (b-a))$$

inserting values for a, b, c

$$= \left(\frac{1+\beta}{2T} - \frac{1-\beta}{2T}\right) + \frac{1}{\frac{\pi T}{\beta}} \sin\left(\frac{\pi T}{\beta} \cdot \left(\frac{1+\beta}{2T} - \frac{1-\beta}{2T}\right)\right)$$

$$\frac{\beta}{T} + \frac{\beta}{\pi T} \sin\left(\frac{\pi T}{\beta} \frac{\beta}{T}\right) = \frac{\beta}{T} + \frac{\beta}{\pi T} \sin(\pi) = \frac{\beta}{T}$$

finally yields

$$\int_{-\infty}^{\infty} |G(f)|^2\ df = \frac{\beta}{T}, \quad \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \tag{7.2}$$

Case 3:

$$|G(f)|^2 = 0 \rightarrow \int_{-\infty}^{\infty} |G(f)|^2\ df = 0 \rightarrow \text{No contribution} \tag{7.3}$$

Combining cases in Eq. 7.1, Eq. 7.2 and Eq. 7.3 yields

$$\int_{-\infty}^{\infty} |G(f)|^2\ df = \frac{\beta}{T} + \frac{1-\beta}{T} = \frac{1}{T} \tag{7.4}$$

7.1.2 Evaluation of numerator

Case 1:

$$|G(f)|^2 = 1, \quad |f| \leq \frac{1-\beta}{2T}$$

$$\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2 \, df = T^2 \int_{-\frac{1-\beta}{2T}}^{\frac{1-\beta}{2T}} f^2 \, df$$

as the integrand is symmetric

$$= T^2 \cdot 2 \cdot \int_{0}^{\frac{1-\beta}{2T}} f^2 \, df = T^2 \cdot 2 \cdot \frac{1}{3} \left(\frac{1-\beta}{2T} \right)^3$$

$$= T^2 \frac{2}{3} \left(\frac{1-\beta}{2T} \right)^3$$

$$= T^2 \frac{2 \cdot (1-\beta)^3}{24 \cdot T^3}$$

finally this yields

$$\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2 \, df = T^2 \frac{2 \cdot (1-\beta)^3}{24 \cdot T^3}, \quad |f| \leq \frac{1-\beta}{2T} \tag{7.5}$$

Case 2:

$$|G(f)|^2 = \frac{1}{2} \left[1 + \cos\left(\frac{\pi T}{\beta} \left[|f| - \frac{1-\beta}{2T} \right] \right) \right], \quad \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T}$$

$$\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2 \, df$$

as integrand is symmetric

$$= T^2 \cdot 2 \cdot \int_{\frac{1-\beta}{2T}}^{\frac{1+\beta}{2T}} \frac{1}{2} f^2 \left[1 + \cos\left(\frac{\pi T}{\beta} \left[|f| - \frac{1-\beta}{2T} \right] \right) \right] \, df$$

$$= T^2 \int_{\frac{1-\beta}{2T}}^{\frac{1+\beta}{2T}} f^2 + f^2 \cos\left(\frac{\pi T}{\beta} \left[|f| - \frac{1-\beta}{2T} \right] \right) \, df$$

with

$$a := \frac{1-\beta}{2T} \in \left[0, \frac{1}{2T} \right], \; b := \frac{1+\beta}{2T} \in \left[\frac{1}{2T}, \frac{2}{2T} \right], \quad c := \frac{\pi T}{\beta}$$

$$= T^2 \int_{a}^{b} f^2 + f^2 \cos(c \cdot (|f| - a)) \, df$$

due to

$$f \geq 0, \quad a \leq f \leq b \rightarrow |f| = f$$

$$= T^2 \int_{a}^{b} f^2 + f^2 \cos(c \cdot (f - a)) \, df$$

$$= T^2 \frac{1}{3}\left(b^3 - a^3\right) + T^2 \int_a^b f^2 \cos(c \cdot (f - a))\, df$$

$$= T^2 \frac{1}{3}\left(b^3 - a^3\right) + T^2 \frac{-(2 - b^2c^2)\sin(c(b-a)) + 2bc\cos(c(b-a)) - 2ac}{c^3}$$

and as

$$c(b-a) = \frac{\pi T}{\beta}\frac{\beta}{T} = \pi \rightarrow \sin(c(b-a)) = \sin(\pi) = 0,\ \cos(c(b-a)) = \cos(\pi) = -1$$

this leads to simplification of the sine and cosine terms:

$$= T^2 \frac{1}{3}\left(b^3 - a^3\right) + T^2 \frac{2bc(-1) - 2ac}{c^3}$$

$$= T^2 \frac{1}{3}\left(b^3 - a^3\right) - T^2 2\frac{b+a}{c^2}$$

Finally plugging in a, b, c yields

$$= T^2 \left[\frac{1}{3}\frac{1}{(2T)^3}\left((1+\beta)^3 - (1-\beta)^3\right) - 2\frac{\frac{1}{T}}{(\frac{\pi T}{\beta})^2}\right]$$

$$= T^2 \left[\frac{1}{3}\frac{1}{(2T)^3}\left((1+\beta)^3 - (1-\beta)^3\right) - 2\frac{\beta^2}{T(\pi T)^2}\right]$$

$$= T^2 \left[\frac{(1+\beta)^3 - (1-\beta)^3 - 2 \cdot 3 \cdot 8 \cdot 1/\pi^2\beta^2}{3 \cdot 8 \cdot T^3}\right]$$

$$\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2\, df = T^2 \frac{(1+\beta)^3 - (1-\beta)^3 - 48\beta^2/\pi^2}{24 \cdot T^3},\ \frac{1-\beta}{2T} < |f| \leq \frac{1+\beta}{2T} \quad (7.6)$$

Case 3:

$$|G(f)|^2 = 0 \rightarrow \int_{-\infty}^{\infty} T^2 f^2\, |G(f)|^2\, df = 0 \rightarrow \text{No contribution} \quad (7.7)$$

Using Eq. 7.5, 7.6 and 7.7 yields

$$\int_{-\infty}^{\infty} T^2 f^2\, |G(f)|^2\, df = T^2 \frac{2 \cdot (1-\beta)^3}{24 \cdot T^3} + T^2 \frac{(1+\beta)^3 - (1-\beta)^3 - 48\beta^2/\pi^2}{24 \cdot T^3}$$

$$= T^2 \frac{(1+\beta)^3 + (1-\beta)^3 - 48\beta^2/\pi^2}{24 \cdot T^3} = T^2 \frac{2(3\beta^2+1) - 48\beta^2/\pi^2}{24 \cdot T^3}$$

$$= T^2 \frac{3\beta^2 + 1 - 24\beta^2/\pi^2}{12 \cdot T^3} = \frac{(3 - \frac{24}{\pi^2})\beta^2 + 1}{12 \cdot T}$$

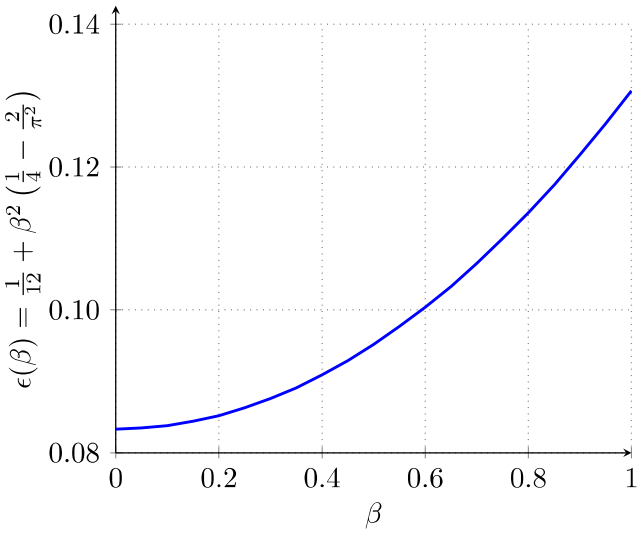


Figure 7.1: MCRB constant $\epsilon(\beta)$ for $\beta \in [0, 1]$

and finally

$$\int_{-\infty}^{\infty} T^2 f^2 \, |G(f)|^2 \, df = \frac{(3 - \frac{24}{\pi^2})\beta^2 + 1}{12 \cdot T} \tag{7.8}$$

7.1.3 Equation for constant

Combining numerator from Eq. 7.8 and denominator from Eq. 7.4 and plugging them into Eq. 2.8 gives

$$\begin{aligned}\epsilon(\beta) &= \frac{\frac{(3-\frac{24}{\pi^2})\beta^2+1}{12 \cdot T}}{\frac{1}{T}} \\ &= \frac{(3 - \frac{24}{\pi^2})\beta^2 + 1}{12} \\ &= \left(\frac{3}{12} - \frac{24}{12\pi^2}\right)\beta^2 + \frac{1}{12}\end{aligned}$$

from which consequently follows

$$\epsilon(\beta) = \frac{\int_{-\infty}^{\infty} T^2 f^2 |G(f)|^2 \, df}{\int_{-\infty}^{\infty} |G(f)|^2 \, df} = \frac{1}{12} + \beta^2\left(\frac{1}{4} - \frac{2}{\pi^2}\right) \tag{7.9}$$

as illustrated in Fig. 7.1 for different β.

7.2 FPGA Resource utilization

Table 7.1: FPGA resource utilization, part 1. Values in parenthesis are relative to the total amount of available resources.

	Total LUTs	Logic LUTs	LUTRAMs	SRLs
System	110933 (26.08 %)	107186 (25.20 %)	248 (0.12 %)	3499 (1.64 %)
Receiver	81367 (19.13 %)	77939 (18.33 %)	0 (0.00 %)	3428 (1.60 %)
Timing recovery	32007 (7.53 %)	31044 (7.30 %)	0 (0.00 %)	963 (0.45 %)
Kalman filter	658 (0.15 %)	658 (0.15 %)	0 (0.00 %)	0 (0.00 %)

Table 7.2: FPGA resource utilization, part 2. Values in parenthesis are relative to the total amount of available resources.

	FFs	RAMB36	RAMB18	URAM	DSP Blocks
System	153783 (18.08 %)	100 (9.26 %)	0 (0.00 %)	0 (0.00 %)	708 (16.57 %)
Receiver	112796 (13.26 %)	35 (3.24 %)	0 (0.00 %)	0 (0.00 %)	708 (16.57 %)
Timing recovery	16276 (1.91 %)	7 (0.65 %)	0 (0.00 %)	0 (0.00 %)	200 (4.68 %)
Kalman filter	443 (0.05 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	8 (0.19 %)

Bibliography

[AF09] F. Adamec and T. Fryza, "Binary division algorithm and implementation in vhdl," in *2009 19th International Conference Radioelektronika*, 2009, pp. 87–90.

[BK23] O. I. Bureneva and O. U. Kaidanovich, "Fpga-based hardware implementation of fixed-point division using newton-raphson method," in *2023 IV International Conference on Neural Networks and Neurotechnologies (NeuroNT)*, 2023, pp. 45–47.

[Cha08] K. Chapman, "Expanding dedicated multipliers," *Xilinx white paper*, 2008.

[Con17] P. Conroy, "Intradyne signal recovery and equalization for optical satellite links," Master's thesis, KTH Royal Institute of Technology, 2017.

[D'A15] A. A. D'Amico, "Efficient maximum-likelihood based clock and phase estimators for oqpsk signals," *IEEE Transactions on Communications*, vol. 63, no. 7, pp. 2647–2657, 2015. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7105396

[DLRa] DLR. Global connectivity. [Online]. Available: https://www.dlr.de/en/kn/research-transfer/research-topics/global-connectivity-for-people-and-machines-2

[DLRb] ——. Optical groundstation ogsop. [Online]. Available: https://www.dlr.de/en/kn/research-transfer/research-infrastructure/optical-groundstation-ogsopf

[DLRc] ——. Osirisv3. [Online]. Available: https://www.dlr.de/en/kn/research-transfer/projects/osiris-optical-communication-in-space/osirisv3

[DMR94] A. D'Andrea, U. Mengali, and R. Reggiannini, "The modified cramer-rao bound and its application to synchronization problems," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 1391–1399, 1994. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=580247

[Gar93] F. Gardner, "Interpolation in digital modems. i. fundamentals," *IEEE Transactions on Communications*, vol. 41, no. 3, pp. 501–507, 1993.

[GPS+17] D. Giggenbach, S. Parthasarathy, A. Shrestha, F. Moll, and R. Mata-Calvo, "Power vector generation tool for free-space optical links — pvget," in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, 2017, pp. 160–165.

[KT08] K. Kikuchi and S. Tsukamoto, "Evaluation of sensitivity of the digital coherent receiver," *Journal of Lightwave Technology*, vol. 26, no. 13, pp. 1817–1822, 2008.

[Lee02] S. J. Lee, "A new non-data-aided feedforward symbol timing estimator using two samples per symbol," *IEEE Communications Letters*, vol. 6, no. 5, pp. 205–207, 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1001665

[MYXF19] H. Ma, L. Yan, Y. Xia, and M. Fu, *Kalman filtering and information fusion.* Cham, Switzerland: Springer Nature, Nov. 2019.

[PSC21] P. K. Pandey, D. Singh, and R. Chandel, "Fixed-point divider using newton raphson division algorithm," in *Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems*, V. Nath and J. K. Mandal, Eds. Singapore: Springer Singapore, 2021, pp. 225–234.

[SRB+23] C. M. Schieler, K. M. Riesing, B. C. Bilyeu, J. S. Chang, A. S. Garg, N. J. Gilbert, A. J. Horvath, R. S. Reeve, B. S. Robinson, J. P. Wang, S. Piazzolla, W. T. Roberts, J. M. Kovalik, and B. Keer, "On-orbit demonstration of 200-Gbps laser communication downlink from the TBIRD CubeSat," in *Free-Space Laser Communications XXXV*, H. Hemmati and B. S. Robinson, Eds., vol. 12413, International Society for Optics and Photonics. SPIE, 2023, p. 1241302. [Online]. Available: https://doi.org/10.1117/12.2651297

[ZC11] X. Zhou and X. Chen, "Parallel implementation of all-digital timing recovery for high-speed and real-time optical coherent receivers," *Opt. Express*, vol. 19, no. 10, pp. 9282–9295, May 2011. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-19-10-9282