MASTER

Implementation of a data acquisition system for phase modulation ellipsometers

Messelaar, A.P.J.M.

Award date:
2002

Link to publication
Implementation of a Data Acquisition System for Phase Modulation Ellipsometers

Arthur Messelaar
December 2000

NF/FIK 2000-14

Eindhoven University of Technology

supervisors:
Jean-Charles Cigal, M.Sc.
Prof. Dr. Ir. Klaas Kopinga
Abstract

The group Elementary Processes in Gas discharges performs research on plasma-induced surface modification. To gain insight in the fast processes involved in this surface modification, a Photo-elastic Modulator Ellipsometer (PME) setup had been used for some time to monitor these processes. A new implementation of the data acquisition part of this setup was required. The new system that has been developed comprises changes in the hardware and the software. As was the case with the existing acquisition system, the hardware part has been implemented using the PhyDAS system. However, obsolete standard PhyDAS components have been replaced by new models, and some components are now used more efficiently. Also some non-standard components have been developed specifically for the new hardware setup.

The software has been completely rewritten. In particular, attention was devoted to the maintainability and user-friendliness of the software and the upgrading of real-time monitoring possibilities of measurement results. All mentioned items have been considerably improved.

The total setup, consisting of the acquisition system and the PME setup, has been tested for its integrity by performing some measurements, of which the results were in agreement with what was expected.

Some attention has been devoted to an analytical and numerical investigation of the relation between an important acquisition parameter, the sample frequency, and the accuracy of the results. It was found that the model generally used for this relation is too optimistic, and that the useful effect of increasing the sample frequency is less than predicted by this model.
## Contents

1 Introduction 3

2 Design Specifications 5

3 Ellipsometry Theory 7
   3.1 General principles of ellipsometry 7
   3.1.1 Representations of polarized light 8
   3.1.2 Representation of polarization state changes 9
   3.1.3 The ellipsometric angles 9
   3.1.4 Ellipsometer components 10
   3.1.5 The Jones-matrix formulation 11
   3.1.6 Measurement Principles 12
   3.2 Photo-elastic Modulator Ellipsometry 13
   3.2.1 Characteristics of the detected intensity 14
   3.2.2 Configurations 16

4 Fourier Theory 19
   4.1 Fourier Series 19
   4.1.1 Signal components and power 20
   4.2 Discrete Fourier Transformation 21
   4.2.1 Sampling of continuous signals 21
   4.2.2 Discretisation 21
   4.2.3 The influence of errors 24

5 Experimental and Hardware Setup 29
   5.1 Experimental Setup 29
   5.1.1 Measurements with and without chopper 31
   5.2 Hardware Setup 32
   5.2.1 The interfaces 32
   5.2.2 Acquisition configuration 35
   5.2.3 Data reduction by the ASR 36
   5.2.4 Timing aspects of chopper related signals 37
   5.2.5 Bottlenecks 38
Chapter 1

Introduction

The group Elementary Processes in Gas discharges (EPG) is a subdivision of the Department of Applied Physics of the Eindhoven University of Technology (TUE). In this group research is performed on gas discharges, especially radio frequency plasmas. One of the points of particular interest is the processing of surfaces with the aid of plasmas.

In industry, the modification of surfaces with the aid of plasmas is a technique which is used for a myriad of manufacturing processes. Examples are the manufacturing of integrated circuits (IC's) and of solar cells. To optimize these manufacturing processes a better understanding of plasma-induced surface modification is required.

In order to gain more insight in the processes involved in plasma-induced surface modification it is necessary to analyze the surface. Several surface analysis techniques are available. A number of aspects with regard to a technique are of importance.

- The conditions under which the surface can be analyzed. It is preferred that the surface can be analyzed in situ, that is inside the plasma reactor (vacuum conditions), as opposed to outside the reactor (ex-situ analysis).
- The moment at which the surface can be analyzed. It is preferred that the surface can be monitored real-time, that is while a plasma is active, as opposed to after the plasma has been deactivated (post-processing analysis).

Furthermore general aspects such as sensitivity, accuracy, and (for real-time measurements) temporal resolution are of importance.

Ellipsometry is a technique that is employed in the group EPG for the analysis of surfaces. It offers high sensitivity and accuracy, in combination with in situ and real-time possibilities. The basic principle of ellipsometry is to let polarized light impinge on a surface, and measure the polarization state of the reflected light. The surface is thus characterized by the change in polarization state it induces.

There are several types of ellipsometry setups. An important distinction between different types of setups is that between spectroscopic setups (which use a multiple wavelength light source and provide spectral information) and non-spectroscopic setups. A disadvantage of spectroscopic setups is the fact that they can limit the feasible temporal resolution.
In the group EPG there is a desire to have both spectral and real-time information about processes. At the onset of this project an infrared spectroscopic ellipsometry setup was already under construction. The aim with this setup is to obtain detailed spectral information rather than to perform real-time measurements. For real-time measurements, a Photo-elastic Modulator Ellipsometry (PME) setup had been in use for some time. One of the aims with respect to this setup is to monitor fast processes during the initial stages after the activation of a plasma, which requires a temporal resolution of the order of milliseconds or less. Because the software and hardware controlling the data acquisition of this setup had become obsolete, there was an interest in a renewed implementation of the data acquisition part of this setup.

This report documents the renewed implementation of the acquisition part of the PME setup. Chapter 2 describes the specifications that were provided with respect to the new implementation. Chapters 3 and 4 deal with ellipsometry theory and Fourier theory respectively. Besides serving as a general introduction to the field of ellipsometry these chapters give some insight in the requirements a typical ellipsometrist may have and the resulting requirements for the acquisition part. Chapters 5 and 6 describe the hardware and software of the acquisition part respectively. Chapter 7 describes how the developed implementation has been tested. Chapter 8 lists some conclusions and recommendations.
Chapter 2

Design Specifications

This chapter describes briefly the wishes and requirements that were formulated with respect to the renewed implementation of the data acquisition part of the Phase Modulation Ellipsometry (PME) setup. The following requirements were explicitly given:

- **Graphical user interface.** The visual representation of the acquired results and the user interface of the existing software were judged as poor. This was for a large part due to the fact the software was written in Pascal. It was suggested to rewrite the software in C, using the Labwindows/CVI [1] environment. This is a C-programming environment which facilitates the design of graphical user interfaces. Furthermore Labwindows/CVI was expected to become one of the new standards for developing acquisition software within the Department of Physics.

- **Upgrading of real-time monitoring possibilities.** The old acquisition part of the PME setup only offered real-time monitoring of the results for very slow measurements. New hardware that had become available was expected to be able to offer more possibilities. The new software was required to support the 'upgrading' of hardware elements, and to utilize the improved features of new hardware elements to enhance the real-time monitoring performance.

- **Support wavelength scans.** In principle, the PME setup is a non-spectroscopic setup. However, the old setup offered the possibility to perform a *wavelength scan*, which is a repetition of the same measurement, each time using light of a different wavelength. So spectral information is gathered *successively* rather than simultaneously, which can be a good option, provided that the measurement conditions can be reproduced well. The new software was required to support an easy integration of a wavelength scan module.

In addition to the above items, which were explicitly provided, an analysis of the existing software yielded some other items that should be improved.

- **User-friendliness.** The user interface should not only be more visually appealing, but also be more user-oriented. Apart from calibration parameters, the user should only have to input parameters directly related to a measurement, such as for instance the measurement time and the temporal resolution. The user interface that was present was more oriented towards a developer, insofar as that some knowledge of the internal operation of the software and acquisition hardware was required in order to successfully perform a measurement.
- **Developer-friendliness.** The actual use of the setup 'in the field' will undoubtedly give rise to a desire for additional features/requirements. To minimize the effort needed to implement additional functionality in the software, the software should be written in a developer-friendly fashion. To this end two categories of developers should be kept in mind. Firstly, the user of the setup, who in principle has no knowledge of the internal operation of the software and only seeks to add or modify models/calculations related to ellipsometry. This type of 'developer' should be able to do this without knowing anything about the rest of the software. On the other hand a developer seeking to enhance the acquisition possibilities of the setup should be able to do this without knowing all the details about ellipsometric models. Hence a distinct separation between acquisition related and ellipsometry related modules should be present. In the software that was present these two domains were inextricably intertwined. Also on other levels a more modular programming style is desired.
Chapter 3

Ellipsometry Theory

This chapter starts with introducing the general principles of ellipsometry. Subsequently it focuses on Phase Modulation Ellipsometry (PME). Besides serving as a general introduction to ellipsometry, the main goal of this chapter is to create an awareness of the characteristics of the signal emerging from a PME ellipsometer, and of the form and nature of the procedures necessary to derive from this signal the desired information.

3.1 General principles of ellipsometry

If polarized light is made to interact with a sample, the emergent (reflected, transmitted) light will in general have a polarization state different from that of the incident light. Ellipsometry is a measurement technique based on the analysis of this change in polarization state. Figure 3.1 shows a schematic representation of an ellipsometer. An unpolarized or non-specifically polarized light beam enters the ellipsometer and interacts with one or more optical components that establish a well-defined polarization state. Subsequently the beam interacts with a sample that induces a change in polarization state, one or more optical components needed for the determination of the resultant polarization state, and finally a detector which detects the intensity of the light.

We will elucidate the concept of polarization and introduce some mathematical representations of polarized light and of polarization changes. These representations will then be used to investigate the interaction of polarized light with optical components and with the sample quantitatively. Important assumptions made are that the light travelling through the ellipsometer can be approximated by a quasi-monochromatic uniform Transverse Electric (TE) infinite plane wave, and that all interactions that occur between light and ellipsometer components are linear and frequency-conserving.

![Figure 3.1: Schematic representation of an ellipsometer](image)
3.1.1 Representations of polarized light

In the following we will consider plane monochromatic TE light waves, with angular frequency $\omega$ and wavenumber $k$. We will always choose the propagation direction of the wave to be the $z$-direction. Such a wave is determined completely by the components of its electrical field vector $\vec{E}$:

$$
E_x = \vec{E}_x \cos(\omega t - k z + \delta_x) = \text{Re} \left( \vec{E}_x e^{i \delta_x} e^{i(\omega t - k z)} \right)
$$

$$
E_y = \vec{E}_y \cos(\omega t - k z + \delta_y) = \text{Re} \left( \vec{E}_y e^{i \delta_y} e^{i(\omega t - k z)} \right)
$$

$$
E_z = 0
$$

The amplitudes $\vec{E}_x$, $\vec{E}_y$ and the phases $\delta_x$, $\delta_y$ are the only parameters, apart from the propagation direction, that are not conserved when this wave interacts with a medium. It proves convenient to combine these parameters into a column vector $J$ called a Jones vector [2]:

$$
J = \left[ \begin{array}{c} \vec{E}_x e^{i \delta_x} \\ \vec{E}_y e^{i \delta_y} \end{array} \right] = \left[ \begin{array}{c} E_{xc} \\ E_{yc} \end{array} \right]
$$

The parameters $E_{xc}$ and $E_{yc}$, which combine amplitude and phase information for the $x$-component and $y$-component respectively into one complex number, are called complex amplitudes. Note that the complete description of a wave can be unambiguously reconstructed from its Jones vector:

$$
J = \left[ \begin{array}{c} E_{xc} \\ E_{yc} \end{array} \right] \Rightarrow E_x(z, t) = |E_{xc}| \cos(\omega t - k z + \text{arg}(E_{xc}))
$$

$$
E_y(z, t) = |E_{yc}| \cos(\omega t - k z + \text{arg}(E_{yc}))
$$

$$
E_z = 0
$$

If a monochromatic wave is considered at a fixed point in space, one finds that the tip of electrical field vector at this point follows a well-defined trajectory, in a repetitive fashion. This fact is referred to as polarization, and the ensemble of the shape, orientation in space and sense of rotation of the trajectory is referred to as the polarization state. In the general case this trajectory is an ellipse, and the light is then called elliptically polarized. Special cases are a circular trajectory (circularly polarized light), and a linear trajectory (linearly polarized light). For elliptical and circularly polarized light one distinguishes between right handed and left handed states, according to the sense of rotation.

A Jones vector is a mathematical representation of a polarization state, since it contains all information about the trajectory. For instance the Jones vector

$$
A e^{i\delta} \left[ \begin{array}{c} \cos \varepsilon \\ i \sin \varepsilon \end{array} \right]
$$

is associated with an elliptical trajectory. This can be seen by reconstructing the electrical field vector components and evaluating them at a fixed point in space (for instance $z = 0$):

$$
J = A e^{i \delta} \left[ \begin{array}{c} \cos \varepsilon \\ i \sin \varepsilon \end{array} \right] \Rightarrow E_x(0, t) = A \cos \varepsilon \cos(\omega t + \delta)
$$

$$
E_y(0, t) = A \sin \varepsilon \cos(\omega t + \delta + \frac{\pi}{2})
$$

(3.1)

What remains is a parametric representation of an ellipse with ellipticity $\tan \varepsilon$ and amplitude $A$. From eq. (3.1) it follows that a polarization state is determined by the phase difference between the $x$-component and $y$-component and by the ratio of amplitudes of the $x$-component and $y$-component. Absolute amplitudes and phases only determine the size of the trajectory.
and the initial starting point of the electrical field vector on this trajectory. A mathematical representation of polarized light that does not contain this additional information is the ratio of complex amplitudes \( \chi \):
\[
\chi = \frac{E_{pc}}{E_{sc}} = \frac{\tilde{E}_p}{\tilde{E}_s} e^{i(\delta_p - \delta_s)}
\]

### 3.1.2 Representation of polarization state changes

When light reflects at a medium it is expedient to consider the components of the electrical field vector with respect to the plane of incidence, that is the plane containing the incident and reflected beam. It is customary to use the symbols \( p \) and \( s \) as opposed to \( x \) and \( y \) to denote directions, \( p \) being the direction parallel to the plane of incidence, \( s \) the direction perpendicular to it. The polarization states of the incident and emergent light are both determined by a relative phase \( (\delta_p - \delta_s) \) and by a relative amplitude \( (E_p/E_s) \), which can be mathematically represented by two ratios of complex amplitudes:

\[
\text{incident: } \chi = \frac{E_{pc}}{E_{sc}} = \frac{\tilde{E}_p}{\tilde{E}_s} e^{i(\delta_p - \delta_s)} \quad \text{emergent: } \chi' = \frac{E_{pc}'}{E_{sc}'} = \frac{\tilde{E}_p'}{\tilde{E}_s'} e^{i(\delta_p' - \delta_s')}
\]

The change in polarization state is defined as the change in relative phase in conjunction with the change in relative amplitude. To represent these two parameters into one complex number the ratio \( \rho \) is defined:

\[
\rho = \frac{\chi'}{\chi} = \frac{\frac{\tilde{E}_p'}{\tilde{E}_s'} e^{i(\delta_p' - \delta_s')}}{\frac{\tilde{E}_p}{\tilde{E}_s} e^{i(\delta_p - \delta_s)}} = \frac{E_p'/E_s'}{E_p/E_s} e^{i[(\delta_p' - \delta_s')-(\delta_p - \delta_s)]}
\]

Note that the absolute value of \( \rho \) equals the change (expressed as a factor) in relative amplitude whereas the argument of \( \rho \) equals the change in relative phase.

### 3.1.3 The ellipsometric angles

In ellipsometry it is customary to characterize a change in polarization state by the so called ellipsometric angles \( \Delta \) and \( \psi \). These angles are directly related to the ratio \( \rho \):

\[
\rho = \tan \psi e^{i\Delta}
\]

So \( \Delta \) and the tangent of \( \psi \) represent the change in relative phase and the change in relative amplitude respectively. (see also fig. 3.2). All conceivable changes in polarization states can be represented by choosing \( 0 \leq \Delta < 2\pi \) and \( 0 \leq \psi \leq \frac{\pi}{2} \). For this reason the convention has been adopted that \( \Delta \) and \( \psi \) should be restricted to these intervals.\(^1\) The collection of all such \((\Delta, \psi)\)-pairs

\[
\{(\Delta, \psi) \mid 0 \leq \Delta < 2\pi, \ 0 \leq \psi \leq \frac{\pi}{2}\}
\]

is referred to as the \( \Delta-\psi \) plane.

Determining the parameters \( \Delta \) and \( \psi \) of a sample is the goal of ellipsometry. The parameters \( \Delta \) and \( \psi \) are independent of ellipsometer settings, and are directly related to the sample.

\(^1\)Occasionally the interval \([-\pi, \pi]\) is used for \( \Delta \).
Figure 3.2: Illustration of a reflection, including the parameters $\Delta$ and $\psi$, the wavelength $\lambda$, and the $p$ and $s$ directions. The incident wave has a $p$ and $s$ component that are equal in magnitude. In this case the ratio of the amplitudes of the $p$ and $s$ component of the emergent wave is equal to the change in this ratio.

However they are in general not the sample properties that the experimenter is interested in. Generally some physical model of the surface of the sample is applied to obtain the desired parameters (for instance layer thickness, complex refractive index) from $\Delta$ and $\psi$ and the used wavelength and angle of incidence. In this respect the parameters $\Delta$ and $\psi$ can be regarded as intermediary parameters, independent of experimental issues (apart from the used wavelength and angle of incidence), and independent of any physical model about the sample.

### 3.1.4 Ellipsometer components

Optical components that typically appear in an ellipsometer are linear polarizers, retarders, and quarter wave plates. Below follows a brief discussion of each.

- A linear polarizer is an optical component which transforms non-specifically polarized light into linearly polarized light. Linear polarizers can have two functions in ellipsometers. Firstly they can be used to produce linearly polarized light, from non-specifically or randomly polarized light. Secondly they can be used for the determination of the polarization state of light which does have a specific polarization state. A polarizer used for this purpose is also referred to as an analyzer.

- A retarder or compensator is an optical component that alters the phase difference between two orthogonal components of the electrical field vector of a wave, leaving the amplitudes unaltered. The change in relative phase that is induced is called the retardation. A typical example of a retarder is a plate of birefringent material. Birefringent material is an optically anisotropic material that can be characterized by two differing refractive indices. These two refractive indices correspond with two perpendicular directions called the fast axis (corresponding with the smallest of the two refractive indices) and the slow axis (corresponding with the largest of the two refractive indices). When a linearly polarized light wave enters the material with its propagation direction perpendicular to these two axes, the electrical field vector of the wave can be decomposed in a component along the fast axis, and a component along the slow axis. Due to the differing refractive indices, these two components of the wave travel with a different phase velocity through the material, which causes a change in the relative phase. The two components of the wave are called the ordinary...
Table 3.1: Jones matrices associated with optical components. The angular setting for the linear polarizer and the retarder is $\theta$.

<table>
<thead>
<tr>
<th>optical component</th>
<th>Jones matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear polarizer</td>
<td>$\begin{bmatrix} \cos^2 \theta &amp; \sin \theta \cos \theta \ \sin \theta \cos \theta &amp; \sin^2 \theta \end{bmatrix}$</td>
</tr>
<tr>
<td>retarder (retardation $\delta$)</td>
<td>$\begin{bmatrix} \cos \frac{1}{2} \delta + i \cos 2\theta \sin \frac{1}{2} \delta &amp; i \sin 2\theta \sin \frac{1}{2} \delta \ i \sin 2\theta \sin \frac{1}{2} \delta &amp; \cos \frac{1}{2} \delta - i \cos 2\theta \sin \frac{1}{2} \delta \end{bmatrix}$</td>
</tr>
<tr>
<td>sample (reflection)</td>
<td>$\begin{bmatrix} \tan \psi e^{i \Delta} &amp; 0 \ 0 &amp; 1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

and the extra-ordinary wave.

- A quarter wave plate is a retarder which induces a change in relative phase of $\pm \frac{\pi}{2}$ (which corresponds with a quarter of a wavelength).

When the above optical components are part of an ellipsometer setup, they have an angular setting associated with them. It is customary that all angles are defined with respect to the plane of incidence of the sample. For a linear polarizer the angular setting is simply the angle between the plane of incidence of the sample and the polarization direction of the emergent light. For a retarder the angular setting is the angle between the plane of incidence of the sample and the fast axis. Changing the angular settings of ellipsometer components (ellipsometer settings) will change the intensity at the detector. The next section will present a method to quantitatively investigate the relation between the detected intensity, the ellipsometer settings, and the sample parameters $\Delta$ and $\psi$.

### 3.1.5 The Jones-matrix formulation.

Conceptually an ellipsometer can be regarded as an array of optical components which alter some characteristics of the light. Since all interactions are assumed to be linear and frequency-conserving the wave can be characterized at each point in the ellipsometer by a Jones-vector. An optical component can thus be regarded as a device which maps the Jones vector of the incident light onto the Jones vector of the emergent light. The Jones-matrix formulation[2] represents these mappings as simple linear matrix multiplications. So with each optical component a Jones-matrix can be associated. If a light wave originally can be represented by a Jones-vector $J$ and it subsequently interacts with optical components $C_1 \ldots C_n$ (see fig. 3.3) which have corresponding Jones-matrices $T_1 \ldots T_n$ the resultant wave can be represented by Jones-vector

$$ T_n \cdot T_{n-1} \ldots \cdot T_1 \cdot J $$

Table 3.1 lists Jones-matrices for some ellipsometer components. The validity of these matrices can be verified by simply investigating the effect that they have on an arbitrary Jones vector.

We note that the intensity $I$ of a wave is proportional to the square of the amplitude of the electrical field. Therefore it follows that the square of the Frobenius norm of the Jones-vector

$$ |J|^2 = \sum |J_{ij}|^2 $$

where $J_{ij}$ are the elements of the Jones-vector.
of a wave

$$||J||^2 = J \cdot J^\dagger = \begin{bmatrix} E_x e^{i\beta_x} \\ E_y e^{i\beta_y} \end{bmatrix} \begin{bmatrix} E_x e^{-i\beta_x} \\ E_y e^{-i\beta_y} \end{bmatrix} = \tilde{E}_x^2 + \tilde{E}_y^2$$

is directly proportional to its intensity.

The Jones-matrix formulation thus supplies a method to relate the intensity at the detector to the ellipsometer settings and the parameters $\Delta$ and $\psi$. Another method that in practice is often used for this purpose is the Mueller-matrix formulation [3]. This method is more powerful as it also allows the presence of depolarizing optical components and partially polarized light. The Mueller-matrix formulation will not be discussed here.

3.1.6 Measurement Principles

There are two expedient ways to experimentally employ the relation between ellipsometer settings and the intensity at the detector, called the null-method and the photometric method. The null-method involves attempting to find ellipsometer settings for which the intensity at the detector is null. This is a slow procedure unsuitable for real-time ellipsometry, and will not be considered further. The photometric method involves modulating one of the ellipsometer settings in time, and deriving the ellipsometric angles from the shape of the time-dependant detected signal. A fundamental assumption of all photometric methods is that the timescale on which the sample parameters $\Delta$ and $\psi$ significantly change is much larger than the modulation period $T$, so that during one (or several) modulation periods $\Delta$ and $\psi$ can be considered (quasi) constant. Typical examples of ellipsometers based on the photometric method are

- Rotating Polarizer Ellipsometer (RPE), Rotating Compensator Ellipsometer (RCE) and Rotating Analyzer Ellipsometer (RAE). These types of ellipsometers have a rotating component, and the ellipsometer setting modulated in time is an angular setting. The modulation frequency is limited to approximately 200-300 Hz.

- Photo-elastic Modulator Ellipsometer (PME). This ellipsometer has a component called a modulator, which consists of a piezo electric component that applies periodic stress to an adjacent crystal causing the crystal to become birefringent. The ellipsometer setting modulated in time is the magnitude of the retardation caused by this stress-induced birefringence. The modulation frequency lies typically in the 20-50 kHz range.

The main advantage of the PME is the feasible temporal resolution, which is directly related to its high modulation frequency. The PME is the topic of the remaining part of this chapter.
3.2 Photo-elastic Modulator Ellipsometry

The unique element of a PME setup is a Photo Elastic Modulator (PEM). As indicated in the previous section this element causes a retardation to be modulated with a certain frequency $f_{\text{PEM}}$. This frequency is the resonance frequency of the crystal in which the birefringence is induced, and is therefore fixed. The amplitude of the retardation can be modified however by controlling the voltage supplied to the piezo-electric element.

Figure 3.4 shows a schematic representation of a PME setup, consisting of a linear polarizer (P), a modulator (M), a sample (S) and an analyzer (A). The ellipsometric settings associated with these components are the angular settings $P$ and $A$ of the polarizer and analyzer respectively, and the angular setting $M$ and (time-dependent) retardation $\delta$ of the modulator. The intensity $I_d$ at the detector is determined by the ellipsometric settings, the sample parameters $\Delta$ and $\psi$, and by the initial intensity $I_i$:

$$I_d = I_d(I_i, P, M, A, \delta, \Delta, \psi)$$

This relation can be determined more explicitly using either the Jones-matrix formulation or the Mueller-matrix formulation. Applying one of these formalisms shows that the detected intensity can be written in the form [4]

$$I_d = \alpha I_0(1 + I_c \cos \delta + I_s \sin \delta)$$

where $\alpha$ is a proportionality constant, and $I_0$, $I_c$ and $I_s$ are dependant on the ellipsometer settings $P$, $M$, $A$ and $\Delta$ and $\psi$:

$$I_0 = 1 - \cos 2\psi \cos 2A + \cos 2(P - M) \cos 2M(\cos 2A - \cos 2\psi)$$
$$+ \sin 2A \cos \Delta \cos 2(P - M) \sin 2\psi \sin 2M$$

$$I_c = \sin 2(P - M)[\sin 2M(\cos 2\psi - \cos 2A) + \sin 2A \cos 2M \sin 2\psi \cos \Delta]/I_0$$

$$I_s = \sin 2(P - M) \sin 2A \sin 2\psi \sin \Delta/I_0$$

Once a choice has been made for the ellipsometric angles $P$, $M$ and $A$, there is a direct relation between $\Delta$ and $\psi$ and $I_c$ and $I_s$. A choice for the ellipsometric angles is called a configuration.

When $\Delta$ and $\psi$ can be considered constant during some time-interval (larger than the modulation period) the time-dependance of $\delta$ is the only source of the time-dependance of the detected intensity, and thus of its shape. Eq. (3.2) suggests that this source can be decomposed into two 'basic contributions', $\cos \delta$ and $\sin \delta$, and that $I_c$ and $I_s$ denote how 'strong' the corresponding basic contribution is present in the source. In the next section it will be shown that for the case of a purely harmonic retardation the two basic contributions leave their own characteristic traces in the shape of the detected intensity.
3.2.1 Characteristics of the detected intensity

The stress which causes the birefringence in a modulator is periodic. It is therefore a safe assumption that the resulting retardation is periodic too. In this case the detected intensity is periodic as well, and can therefore be expressed as a Fourier series:

\[ I_d(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t) \quad (\omega = 2\pi/T) \] (3.6)

The harmonic model for the modulation

The form of the modulation is often assumed to be purely harmonic [4]:

\[ \delta(t) = B \sin \omega t \] (3.7)

The amplitude \( B \) is referred to as the retardation or modulation amplitude as well as the retardation or modulation depth. With this assumption the detected intensity (3.2) becomes:

\[ I_d(t) = I_0 + I_0 I_c \cos(B \sin \omega t) + I_0 I_s \sin(B \sin \omega t) \] (3.8)

Note that we have omitted the proportionality constant \( a \), which is permitted since we're only interested in the shape and not the magnitude of the detected intensity. According to Abramowitz [5] the Fourier series for the functions \( \cos(B \sin \omega t) \) and \( \sin(B \sin \omega t) \) are:

\[ \cos(B \sin \omega t) = J_0(B) + 2J_2(B) \cos 2\omega t + 2J_4(B) \cos 4\omega t + \cdots \]
\[ \sin(B \sin \omega t) = 2J_1(B) \sin \omega t + 2J_3(B) \sin 3\omega t + \cdots \] (3.9)

In these expressions the functions \( J_n \) are Bessel functions [5] of the first kind. Figure 3.5 plots some of these functions on the interval \([0, 2\pi]\). Inserting expressions (3.9) into (3.8) we obtain

\[ I_d(t) = I_0 + I_0 I_c (J_0(B) + 2J_2(B) \cos 2\omega t + 2J_4(B) \cos 4\omega t + \cdots) + I_0 I_s (2J_1(B) \sin \omega t + 2J_3(B) \sin 3\omega t + \cdots) \]

The Fourier coefficients in (3.6) are thus given by

\[ a_0 = I_0 + I_0 I_c J_0(B) \]
\[ a_1 = 0 \]
\[ a_2 = 2I_0 I_c J_2(B) \]
\[ a_3 = 0 \]
\[ a_4 = 2I_0 I_c J_4(B) \]
\[ \vdots \]
\[ b_1 = 2I_0 I_s J_1(B) \]
\[ b_2 = 0 \]
\[ b_3 = 2I_0 I_s J_3(B) \]
\[ b_4 = 0 \]
\[ \vdots \] (3.10)
From these results a number of conclusions can be drawn:

- A nonzero value for $I_e$ gives rise to the presence of even Fourier components, whereas a nonzero value of $I_s$ gives rise to the presence of uneven Fourier components. The effects of the two 'basic contributions' mentioned in the previous section are thus very easily identifiable.

- The magnitude of the $n$-th Fourier component is proportional to $J_n(B)$. The retardation depth can be assumed to be smaller than $2\pi$. Figure 3.5 suggests and calculations show that as $n$ becomes larger, $J_n(B)$ becomes negligibly small (yet remains nonzero). Therefore, from a practical point of view, the number of nonzero Fourier components may be regarded as finite, and the detected intensity may regarded as a bandwidth-limited signal.

- $I_e$ and $I_s$ can be determined from $a_0$, and an even and uneven component. Since the lower order components are expected to be largest it is advisable to use $a_0$, $b_1$ and $a_2$. This yields

$$I_0 = a_0 - \frac{1}{2}J_0(B)/J_2(B)$$

$$I_e = \frac{1}{2}a_2/(J_0J_2(B))$$

$$I_s = \frac{1}{2}b_1/(J_0J_1(B))$$

- Dividing two even components or two odd components provides a relation between the retardation amplitude and a ratio of Fourier components. For instance

$$b_1/b_3 = J_1(B)/J_3(B) \quad \text{and} \quad a_2/a_4 = J_2(B)/J_4(B) \quad (3.11)$$

These relations can be used to determine the retardation amplitude from the detected intensity.

A general model

Several other models for the form of the modulation have been proposed, such as the residual strain model[4], which assumes a retardation of the form

$$\delta(t) = \delta_0 + B \sin \omega t,$$

and the multiple-harmonics model[4], which assumes the retardation to contain some higher harmonics. What follows below is not a model of the form of the retardation, but a description of the characteristics of the detected intensity as a result of an arbitrary periodic retardation. This description is based on the following ideas:

- Since only the shape of the signal is of importance, there is no need to consider $a_0$. The normalized Fourier coefficients

$$\hat{a}_n = a_n/a_0$$

$$\hat{b}_n = b_n/a_0$$

suffice to describe the shape.

- Equation (3.2) applies irrespective of the form of the retardation. Since the ellipsometer is a linear apparatus each of the basic contributions $\cos \delta$ and $\sin \delta$ 'add' their Fourier components to the detected intensity independently of one another. The sum effect can thus be expressed as a linear combination of individual effects of the basic contributions.
The normalized Fourier coefficients determining the shape of the detected intensity can thus be written as

$$\hat{a}_n = \alpha_{cn} I_c + \alpha_{sn} I_s$$
$$\hat{b}_n = \beta_{cn} I_c + \beta_{sn} I_s$$

(3.12)

The coefficients $\alpha_{cn}$, $\alpha_{sn}$, $\beta_{cn}$, $\beta_{sn}$ must be determined experimentally, by measuring the detected intensity for at least two well-known $(I_c, I_s)$ combinations. The next section will show that it is possible to employ the convenient $(1,0)$ and $(0,1)$ combinations, for which the normalized Fourier coefficients $(\hat{a}_n, \hat{b}_n)$ equal the coefficients $\alpha_{cn}$, $\beta_{cn}$ and $\alpha_{sn}$, $\beta_{sn}$ respectively. Once the coefficients have been determined the linear sets of equations (3.12) can be used to derive $I_c$ and $I_s$ from the detected intensity.

### 3.2.2 Configurations

The parameters $I_c$ and $I_s$ can be considered intermediary parameters. Their relation with $\Delta$ and $\psi$ is dependant on the ellipsometer settings $P$, $M$ and $A$ via equations (3.3-3.5). A particular choice for these settings is called a configuration. There are two frequently used configurations, generally referred to as configuration II and configuration III [6], for which equations (3.3-3.5) considerably simplify.

#### Configuration II

For the angular settings $P = \frac{\pi}{4}$, $M = 0$, $A = \frac{\pi}{4}$ equations (3.3-3.5) reduce to

$$I_0 = 1$$
$$I_c = \sin 2\psi \cos \Delta$$
$$I_s = \sin 2\psi \sin \Delta$$

(3.13)

The collection of angular settings

$$\{(P, M, A) \mid P - M = \pm \frac{\pi}{4}, M = 0 \lor M = \frac{\pi}{2}, A = \pm \frac{\pi}{4}\}$$

(3.14)

all result in either equations (3.13) or in equations

$$I_0 = 1$$
$$I_c = -\sin 2\psi \cos \Delta$$
$$I_s = -\sin 2\psi \sin \Delta$$

(3.15)

Each of the angular settings of (3.14) is referred to as configuration II. Solving equations (3.13) or (3.15) shows that $\Delta$ and $\psi$ are not uniquely determined (see fig. 3.6). In the previous section the special cases that $(I_c, I_s) = (1,0)$ and $(I_c, I_s) = (0,1)$ were mentioned with respect to the calibration method for the general model. We note that configuration II offers the possibility to generate these $(I_c, I_s)$-combinations by choosing $(\Delta, \psi) = (0, \frac{\pi}{4})$ and $(\Delta, \psi) = (\frac{\pi}{2}, \frac{\pi}{4})$ respectively. As can be seen by considering the Jones-matrix for a reflection at a surface

$$\begin{bmatrix}
\tan \psi \exp i \Delta & 0 \\
0 & 1
\end{bmatrix}$$

these two $(\Delta, \psi)$-pairs correspond with no sample and with transmission through a quarter wave plate respectively.
Configuration III

For the angular settings \( P = 0, M = \frac{\pi}{4}, A = -\frac{\pi}{4} \) equations (3.3-3.5) reduce to

\[
\begin{align*}
I_0 &= 1 \\
I_c &= -\cos 2\psi \\
I_s &= \sin 2\psi \sin \Delta
\end{align*}
\] (3.16)

The collection of angular settings

\[
\{(P, M, A) \mid P - M = \pm \frac{\pi}{4}, M = \pm \frac{\pi}{4}, A = \pm \frac{\pi}{4}\}
\] (3.17)

all result in either equations (3.16) or in the equations

\[
\begin{align*}
I_0 &= 1 \\
I_c &= -\cos 2\psi \\
I_s &= -\sin 2\psi \sin \Delta
\end{align*}
\] (3.18)

Each of the angular settings of (3.17) is referred to as configuration III. Solving equations (3.16) or (3.18) shows that \( \Delta \) and \( \psi \) are not uniquely determined (see fig. 3.6).

Other configurations

Besides the fact that configurations II and III do not uniquely determine \( \Delta \) and \( \psi \), they have the drawback that the accuracy of the results is non-uniform. This means that the distorting effect that errors in \( I_c \) and \( I_s \) have on the determination of \( \Delta \) and \( \psi \) depends on the position in the \( \Delta-\psi \) plane. Choosing the angular settings different from those of configuration II or III will effect the characteristics of the accuracy and the ambiguity. It is possible to choose the angular settings in such a way that a large part of the \( \Delta-\psi \) plane is free of ambiguity. However, for these angular settings equations (3.3-3.5) do not simplify, and one may have to resort to numerical methods for solving of the equations. Since those methods can be relatively very time-consuming, this may pose a problem when real-time monitoring of the results is desired. A solution to this problem is to create a lookup-table prior to a measurement. Combined with linear interpolation a lookup-table can produce results fast and accurately.
Chapter 4

Fourier Theory

The first part of this chapter is common-knowledge, basic Fourier series theory, and is included mainly to introduce the notation and terminology used. Next, the expressions for the Discrete Fourier Transformation are derived and presented as a means to extract the Fourier coefficients from a sampled signal.

Finally, the influence of errors in the sample values on the calculated Fourier coefficients is examined, and it is investigated whether increasing the sample frequency can reduce the effects of these errors. Whether increasing the sample frequency indeed improves the accuracy of the results is of importance with respect to the data acquisition part, since increasing the sample frequency will increase the data flow.

4.1 Fourier Series

According to Fourier, every periodic function \( g \) with period \( T \) can be written as a Fourier series:

\[
g(t) = a_0 + \sum_{n=1}^{\infty} \left( a_n \cos \frac{2\pi nt}{T} + b_n \sin \frac{2\pi nt}{T} \right) \tag{4.1}
\]

which is sometimes written more compactly as

\[
g(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos n\omega t + b_n \sin n\omega t)
\]

by using the angular frequency \( \omega = 2\pi/T \). The sets of numbers \( a_n \) and \( b_n \) are called the Fourier coefficients and are given by the so-called Euler formulas:

\[
a_0 = \frac{1}{T} \int_{<T>} g(t) \, dt, \quad a_n = \frac{2}{T} \int_{<T>} g(t) \cos n\omega t \, dt, \quad b_n = \frac{2}{T} \int_{<T>} g(t) \sin n\omega t \, dt,
\]

where the notation \( \langle T \rangle \) is used to denote that the interval of integration may be \textit{any} interval of length \( T \). The Fourier series may be thought of as a decomposition of a periodic function into several components, which are functions of the form

\[
a_n \cos n\omega t + b_n \sin n\omega t \tag{4.2}
\]
Since (4.2) is obviously a periodic function with angular frequency \( n\omega \), this component is called the \( n\)-th frequency component. The frequency of the \( n\)-th frequency component is denoted by \( f_n \), and has the value:

\[
f_n = 2\pi n\omega = n/T
\]  

Equation (4.2) can also be written in the form

\[A \cos(n\omega t + \varphi)\]

The parameter \( A \) is called the magnitude, amplitude or strength of the component, and \( \varphi \) is called the phase.

Another representation of (4.1) is

\[g(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega t}\]  

which can be obtained by writing the cosines and sines in (4.1) as sums of complex exponentials \( e^{in\omega t} \) and \( e^{-in\omega t} \), and by defining

\[c_n = \begin{cases} a_0 & n = 0 \\ \frac{1}{2}a_n - \frac{1}{2}ib_n & n > 0 \\ \frac{1}{2}a_{-n} + \frac{1}{2}ib_{-n} & n < 0. \end{cases}\]  

Inserting these expressions for \( c_n \) in the Euler formulas yields (for all \( n \)):

\[c_n = \frac{1}{<T>} \int g(t) e^{-in\omega t} dt\]  

Which of the representations (4.1), (4.4) is used is mostly a matter of convenience or personal taste. Either one of the pairs of coefficients \((a_n, b_n)\) and \((c_n, c_{-n})\) provides all information about the \( n\)-th frequency component of the signal \( g \). Furthermore, when a pair of coefficients for one representation is known, the pair for the other representation follows immediately from equation (4.5).

### 4.1.1 Signal components and power

Often a physical signal has a power associated with it, which is proportional to the square of the signal. Hence the mathematical definition of the average power \( P \) of a periodic signal \( g \) reads

\[P(g) = \overline{g^2} = \frac{1}{<T>} \int g^2(t) dt\]  

If we insert the Fourier series representation (4.1) of \( g \) we obtain, considering the orthogonality of the cosines and sines

\[P(g) = a_0^2 + \sum_{n=1}^{\infty} \frac{1}{2}(a_n^2 + b_n^2)\]
which shows that the power of a periodic signal is the sum of the power of the dc-component and the individual powers of the frequency components. Inspired by this fact we write (4.8) as

\[ P(g) = P(0) + \sum_{n=1}^{\infty} P(f_n) \]  

(4.9)

with

\[ P(0) = a_0^2 \]

the power of the dc-component, and

\[ P(f_n) = \frac{1}{2}(a_n^2 + b_n^2) \]

the power of the n-th frequency component.

4.2 Discrete Fourier Transformation

4.2.1 Sampling of continuous signals

When digital processing of a continuous signal is desired, the signal must first be sampled, typically by an analog to digital converter (ADC). Conceptually this sampling of the signal may be considered as a combination of two processes: discretisation and quantisation. Discretisation refers to the sample moment, and more specifically to the fact that the signal value is only recorded at discrete, evenly spaced points in time, as opposed to being recorded continuously at all times. Quantisation refers to the sample value, and more specifically to the fact that a sample value is represented by a finite number of bits, and thus can only assume a finite number of (evenly spaced) values. The signal value is rounded off to the nearest number that can be represented by a bit combination.\(^1\) Note that throughout this report the terms 'sample' and 'sample value' are used in an ambiguous fashion. They can either refer to a bit combination representing an integer value, or to the real (in the mathematical sense) value of a signal that corresponds with this bit combination.

When one wants to extract the Fourier coefficients from a sampled signal both quantisation and discretisation pose a problem. For now we will concentrate on the issue of discretisation, and defer the issue of quantisation to another section.

4.2.2 Discretisation

This section deals with the discretisation of a periodic and bandwidth limited signal. This is a periodic signal which consists of only a finite number \( m \) of frequency components:

\[ g(t) = a_0 + \sum_{n=1}^{m} \left( a_n \cos \frac{2\pi nt}{T} + b_n \sin \frac{2\pi nt}{T} \right) \]

(4.10)

(From section 3.2.1 it is clear that the detector signal is such a signal). When sampling such a signal, a number of parameters are of importance (see also figure 4.1):

\(^1\) This applies to an ideal ADC. In practice each sample moment is subject to jitter.

\(^2\) This applies to an ideal ADC. In practice an ADC exhibits differential nonlinearity, which means that the boundary separating a 'rounding-off-upward-action' from a 'rounding-off-downward-action' doesn't lie symmetrically in the middle between two consecutive bit combinations, but may lie anywhere between these two bit combinations, the position depending on the pair of bit combinations concerned.
- The number of samples $N$ that is taken per period $T$ of the signal.
- The sample period $T_s = T/N$, which is the time between two consecutive sample moments.
- The time $t_k$ at which the $k$-th sample is taken. The time at which the first sample is taken is defined as the origin of the time axis, so that $t_k = kT_s$, $k = 0 \ldots N - 1$.
- The value $g_k$ of the $k$-th sample: $g_k = g(t_k)$.

We note that $(4.10)$ contains $2m + 1$ unknown coefficients. When the signal value is known at $2m + 1$ or more times, that is when $N > 2m$, it should be possible to determine these coefficients from the resulting $2m + 1$ or more equations.

When observing one of the Euler formulas, for instance

$$a_n = \frac{2}{T} \int_{<T>} g(t) \cos(n2\pi t/T) \, dt \quad (4.11)$$

it is clear that this formula can't be used to determine $a_n$, because the function $g$, and therefore the integrand, is only known at times $t_k$. However, a natural way to approximate $(4.11)$, is to replace the integral with a summation, using only the values of the integrand that are known. This boils down to approximating a smooth surface by a sum of rectangular surfaces. (See fig. 4.2).

Applying this approximation principle to all Euler formulas yields the expressions:

$$\tilde{a}_0 = \frac{1}{N} \sum_{k=0}^{N-1} g_k, \quad \tilde{a}_n = \frac{2}{N} \sum_{k=0}^{N-1} g_k \cos(n2\pi k/N), \quad \tilde{b}_n = \frac{2}{N} \sum_{k=0}^{N-1} g_k \sin(n2\pi k/N), \quad (4.12)$$

where tildes have been added to denote that the symbols refer to values which were obtained via an approximation. When the same approximation principle is applied to $(4.6)$ the result is

$$\tilde{c}_n = \frac{1}{N} \sum_{k=0}^{N-1} g_k e^{-in2\pi k/N} \quad (4.13)$$

Expressions $(4.12)$ and $(4.13)$ are both referred to as the Discrete Fourier Tranformation (DFT). They map a discrete set of numbers $g_k$ onto another discrete set of numbers $(\tilde{a}_n, \tilde{b}_n)$ c.q. $\tilde{c}_n$. It is possible to investigate the 'goodness' of the obtained approximations by relating the approximated coefficients $(\tilde{a}_n, \tilde{b}_n, \tilde{c}_n)$ to the original ones $(a_n, b_n, c_n)$. We can achieve this through inserting the Fourier representation $(4.4)$ evaluated at time $t = t_k$ into $(4.13)$,
In Figure 4.2: Approximation of an integral by a sum of rectangular surfaces

which produces the relation between the coefficients \( \hat{c}_n \) and \( c_n \). The relation between the coefficients \((\hat{a}_n, \hat{b}_n)\) and \((a_n, b_n)\) will then immediately follow from equation (4.5). We start with evaluating (4.4) at \( t = kT_s \). This yields

\[
g(t) = \sum_{l=-\infty}^{\infty} c_l e^{i2\pi lt/T} \quad \Rightarrow \quad g_k = \sum_{l=-\infty}^{\infty} c_l e^{i2\pi lk/N}
\]

Inserting this into (4.13) we obtain

\[
\tilde{c}_n = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=-\infty}^{\infty} c_l e^{i2\pi lk/N} e^{-i2\pi kn/N}
\]

\[
= \frac{1}{N} \sum_{l=-\infty}^{\infty} c_l \sum_{k=0}^{N-1} e^{i2\pi k(l-n)/N}
\]

\[
= \frac{1}{N} \sum_{l=-\infty}^{\infty} c_l \sum_{k=0}^{N-1} \delta(l - n - k) \quad (1)
\]

\[
= \sum_{l=-\infty}^{\infty} c_l \sum_{r=-\infty}^{\infty} \delta(l - n - rN)
\]

\[
= \sum_{r=-\infty}^{\infty} c_{n+rN} \quad (4.14)
\]

where \( \delta \) is the Krönecker delta function. In step (1) the identity

\[
\sum_{k=0}^{N-1} e^{i2\pi k(l-n)/N} = N \sum_{r=-\infty}^{\infty} \delta(l - n - rN) \quad (4.15)
\]

has been used, which is derived in appendix A.1.

This result shows that the discrete row of coefficients \( \tilde{c}_n \) is a superposition of the original row \( c_n \) and infinitely many 'shifted copies' of this row. For a bandwidth-limited signal of the form (4.10), only the coefficients \( c_{-m} \ldots c_m \) are nonzero. This means that if the 'width' \( N \) of the shifting interval is larger than \( 2m \), there is no overlap between the nonzero coefficients of the row \( c_n \), and the nonzero coefficients of an arbitrary shifted copy of that row. See also figure (4.3). The condition \( N > 2m \) can also be written as

\[
\frac{N}{T} > \frac{2m}{T} \quad \Rightarrow \quad f_s > 2f_m
\]

where \( f_s = 1/T_s \) is the sample frequency.
Figure 4.3: The row $c_n$ and two shifted copies, when $m = 3$, and $N = 10$. Since the coefficients $\tilde{c}_n$ and $c_n$ are complex numbers the absolute value is displayed.

**Conclusion**

When a periodic signal is bandwidth-limited, that is when it is of the form (4.10), the expressions (4.12-4.13) yield exact results for the Fourier coefficients, provided that the sample frequency is higher than twice the maximum frequency present in the signal. This fact is known as the Nyquist or Shannon theorem.

This conclusion provides a minimal requirement for the sample frequency. The next section will show that it can be advantageous to choose the sample frequency higher than this minimal requirement.

**Notation**

Occasionally it can be convenient to write expressions (4.12) more compactly. To this end we introduce the abbreviations

$$c_{kn}^N = \cos(n2\pi k/N) \quad s_{kn}^N = \sin(n2\pi k/N)$$

Note that $c_{kn}^N$ is the $k$-th element of an $n$-cycle discrete cosine with period $N$, and $s_{kn}^N$ is the $k$-th element of an $n$-cycle discrete sine with period $N$. With these abbreviations (4.12) becomes:

$$a_0 = \frac{1}{N} \sum_{k=0}^{N-1} g_k, \quad a_n = \frac{2}{N} \sum_{k=0}^{N-1} g_k c_{kn}^N, \quad b_n = \frac{2}{N} \sum_{k=0}^{N-1} g_k s_{kn}^N$$

(4.16)

where we have omitted the tildes.

**4.2.3 The influence of errors**

A row of numbers obtained through sampling a continuous signal will in general only approximate the values of this signal at the sample times $t_k$. There can be several sources of error:

- Quantization noise, as a consequence of the fact that only a finite number of bits are used to represent a sample value.
- Phase noise, which refers to jitter in the sample moment. The error in the time at which a sample is taken can be taken into account by proposing an error in the sample value, while considering the sampling to be perfectly equidistant in time.
- Natural noise which is always present on a signal, unrelated to the sampling process.
These sources of error can be taken into account by considering a noise-corrupted signal
\[ \tilde{g} = g + e \]
which is the sum of the original signal \( g \) and an error or noise signal \( e \). We explicitly make the assumption that the noise signal is so rapidly varying that it can be treated as white noise. This implies that the noise values at two different sample moments \( t_k \) and \( t_l \) are uncorrelated:
\[ \epsilon_k \text{ and } \epsilon_l \text{ uncorrelated } \quad k \neq l \quad (4.17) \]
Furthermore, the signals \( g \) and \( e \) are considered to be uncorrelated and the power of the noise is expected to be constant. We will refer to these conditions as the white noise conditions.
Sampling of the noise-corrupted signal yields noise-corrupted sample values
\[ \tilde{g}_k = g_k + \epsilon_k \]
Applying the DFT to noise-corrupted sample values yields noise-corrupted Fourier coefficients \((\tilde{a}_n, \tilde{b}_n)\). When investigating the relation between the errors \( \epsilon_k \) in the sample values, the noise-corrupted Fourier coefficients \((\tilde{a}_n, \tilde{b}_n)\), and the uncorrupted coefficients it proves advantageous to consider the error in the power
\[ \tilde{P}(f_n) = \frac{1}{2}(\tilde{a}_n^2 + \tilde{b}_n^2) \quad (4.18) \]
of a frequency component, rather than the individual errors in \( a_n \) and \( b_n \). We denote the 'power' that one finds by applying (4.18) to the noise-corrupted coefficients by \( \tilde{P}(f_n) \), and the difference with the proper value for the power by \( E(f_n) \):
\[ \tilde{P}(f_n) = \frac{1}{2}(\tilde{a}_n^2 + \tilde{b}_n^2) = P(f_n) + E(f_n) \quad (4.19) \]
Inserting the expressions for the DFT we find for \( \tilde{P}(f_n) \):
\[ \tilde{P}(f_n) = \frac{2}{N^2} \left( \sum_{k=0}^{N-1} (g_k + \epsilon_k)\hat{e}_N^k \right)^2 + \frac{2}{N^2} \left( \sum_{k=0}^{N-1} (g_k + \epsilon_k)\hat{s}_N^k \right)^2 \quad (4.20) \]

Some additional algebra is required to write this in the form (4.19), that is as a sum of the proper power and an error.

**Additive power model**

The power \( P(\tilde{g}) \) of the noise-corrupted signal is
\[ P(\tilde{g}) = \frac{1}{T} \int_{<T>} \tilde{g}^2(t) \, dt = \frac{1}{T} \int_{<T>} g^2(t) \, dt + \frac{1}{T} \int_{<T>} e^2(t) \, dt + \frac{1}{T} \int_{<T>} 2g(t)e(t) \, dt \]
If the white noise conditions are met the cross-term approximates to zero, and the power of the noise-corrupted signal can then be written as the sum of the powers of the two signals of which it is composed
\[ P(\tilde{g}) = P(g) + P(e) \]
The calculation of the 'power' \( \tilde{P}(f_n) \) simplifies if similar approximations are applied to discrete cross-terms while writing out (4.20):
- The signals $e$ and $g$ are uncorrelated. This implies that expressions of the form
  \[ \sum_{k,l} g_k e_l \]
  approximate to zero.
- Values of the noise at different times are uncorrelated. This implies that expressions of the form
  \[ \sum_{k \neq l} e_k e_l \]
  approximate to zero.

With these approximations (4.20) can be written as

\[
\hat{P}(f_n) = \frac{1}{2}(a_n^2 + b_n^2) + \frac{2}{N^2} \sum_{k=0}^{N-1} e_k^2 = P(f_n) + \frac{2}{N} P(e)
\]

(4.21)

where $P(e) = \sum_k e_k^2 / N$ is an estimate of the power of the noise signal. The power $\hat{P}(f_n)$ found by using noise corrupted sample values is thus expressed as the sum of the power of the uncorrupted signal and part of the power of the noise signal. For this reason we will refer to this model as the additive power model. According to this model the error $E(fn)$ in the power of a frequency component is inversely proportional to the number of samples per period. Increasing the sample frequency would therefore increase the accuracy. The model also suggests that the error is always positive and (approximately) the same for all frequency components. If this were indeed the case, one could correct the experimentally found $\hat{P}(f_n)$ by subtracting the error. The error could be found by determining the power of a component which is known not to be present in the signal. The additive power model is a widely used model. However, the next section will show that it is not very realistic. As a consequence the mentioned correction is not possible.

**Simulation of errors**

If no a priori assumptions are made about the contribution of the various terms putting (4.20) into the form (4.19) becomes more laborious. The result is (see appendix A.2):

\[
\hat{P}(f_n) = P(f_n) + \frac{2}{N^2} \sum_k e_k^2 + \frac{2}{N^2} \sum_{k \neq l} e_k e_l (c_N^{nk} c_N^{nl} + s_N^{nk} s_N^{nl}) + \frac{2}{N} (a_n \sum_k e_k c_N^{nk} + b_n \sum_k e_k s_N^{nk})
\]

(4.22)

The error $E(fn)$ has been split up into three contributions:

- Error 1 is a sum of positive definite terms and therefore always increases the obtained value for the power. This error only depends on the noise signal.
Figure 4.4: (a) Errors 1 and 2 for 100 different noise patterns, (b) Errors 2 and 3 for 100 different noise patterns, (c) The mean of error 1 and the standard deviations of error 2 and 3 plotted as a function of \( N \) on a double-logarithmic scale.

All obtained results apply to a frequency component of magnitude 10, and a noise amplitude of 0.1, which corresponds with a signal to noise ratio of 42 dB.

- Error 2 is a sum of terms that can be positive as well as negative, and can therefore both increase or decrease the obtained value for the power. This error only depends on the noise signal.
- Error 3 is a sum of terms that can be positive as well as negative, and can therefore both increase or decrease the obtained value for the power. This error depends both on the noise signal, and on the magnitude of the frequency component for which the power is calculated.

To get an impression of the behavior of these errors some simulations have been carried out. Choosing a particular value for \( N \), the numbers \((e_i)_{i=0}^{N-1}\) have been generated several times with a random generator. The random generator had a uniform probability distribution within an interval positioned symmetrically around null. Also the numbers \(g_k\) and \(\tilde{g}_k\) have been calculated by making a particular choice for the signal \(g\). The errors according to eq. 4.22 have been calculated. As a test for the validity of eq. 4.22 the quantities \(\tilde{P}(J_n)\) and \(P(f_n)\) have been calculated by using the numbers \(\tilde{g}_k\) and \(g_k\) respectively. The difference of these two quantities should be equal to the sum of errors 1, 2 and 3, calculated via eq. 4.22. This was found to be the case.

Figure 4.4ab shows the typical behavior of the errors. As expected error 1 is approximately constant. Errors 2 and 3 do appear to have an expectation value of null, but the variation of particularly error 3 is enormous compared to the value of error 1.

The simulations have been repeated for different values of \( N \). For each \( N \) 30 noise patterns have been generated, resulting in 30 different values for the three errors. To characterize errors 2 and 3 their standard deviations (which are equal to the root mean square (RMS) values of these errors if one assumes that their expectation values are null) have been calculated using these 30 values. Figure 4.4c shows error 1 and the standard deviations of errors 2 and 3 as a function of \( N \). It can be seen that increasing \( N \) tends to decrease all errors, and that error 3 remains the dominant error.

Conclusions

Errors 2 and 3 are neglected by the additive powers model, since they are expected to average out. However it can be shown (see appendix A.3) that the RMS of error 3 is approximately a factor \(\sqrt{\tilde{P}(f_n)/P(e)}\sqrt{N}\) larger than the RMS of error 1. Thus any experiment with a realistic signal to noise ratio will have error 3 as the dominant error, as is reflected in figure 4.4. Since
error 3 can be positive as well as negative it is not possible to apply a correction, as was the case for the additive power model. Furthermore, the error in a frequency component depends on the magnitude of that component, and is therefore not the same for all components, as suggested by the additive power model.

It can be shown that (see appendix A.3) the standard deviation of error 3 is approximately proportional to $1/\sqrt{N}$. Figure 4.4c indeed reflects this behavior. Increasing the sample frequency (that is increasing $N$) thus decreases the error in the calculated power of the frequency components. However, it should be noted that the above results reflect an ideal-case scenario, and only hold if the white noise conditions are not violated. In practice noise is always bandwidth-limited and increasing the sample frequency (which implies decreasing the time interval between two sample moments) will cause condition (4.17) to be increasingly violated. At a certain point, increasing the sample frequency will no longer improve the accuracy of the results. This point is determined by the characteristics of the three sources of the noise signal mentioned at the beginning of this section, and can best be determined experimentally for a particular measurement.
Chapter 5

Experimental and Hardware Setup

The data acquisition part is intended to be independent of the specific characteristics of the experimental setup. However, there are some requirements that should be met. Therefore the first part of this chapter focuses on the experimental setup. The second part of this chapter describes the structure and operation of the hardware setup.

5.1 Experimental Setup

Figure 5.1 shows a schematic representation of the experimental setup. A discussion of some of the elements and the corresponding requirements follows.

Source

The acquisition part imposes no restrictions on the light source. However, the choice for a particular light source will determine the type of detector required. At the moment a helium neon laser is used, but for the future it is intended that an infrared tunable diode laser (TDL) is used as light source. This laser has the possibility of setting the wavelength by controlling a temperature and current setting, and thus offers the possibility of wavelength scans.

Polarizer and Analyzer

The acquisition part is not influenced by the characteristics of the polarizer and analyzer. (The validity of the obtained $\Delta$ and $\psi$ are, however.) The choice for a particular type of polarizer will mostly depend on the wavelength used.

PEM

The PEM is required to supply a logic TTL-level compatible reference signal that has the same frequency as the retardation. Furthermore the phase difference between the retardation and this TTL signal must be constant. The reference signal supplied by the PEM is the master signal from which all other signals controlling the acquisition are derived.

Detector

The detector (in conjunction with subsequent amplifying electronics) is required to supply a voltage which depends linearly on the intensity of the incident light. Furthermore this supplied
voltage must be confined to the interval \((-1V, 1V)\), this being the analog input range of the ADC of the acquisition part. If the infrared TDL is used as a light source, the detector will be a Mercury-Cadmium-Telluride (MCT) detector. This type of detector is linear during a restricted time interval (< 1s), but exhibits a drift in its offset on a larger time scale. This drift can be considered quasi-static with respect to the time scale of the PEM modulation, but has an amplitude that can be up to a factor 30 larger than the signal amplitude [7]. If the detector signal would be directly fed to the ADC the largest part of the ADC input range would be used to accommodate the uninteresting variations of the drift, and only a fraction of the input range would remain to accommodate the interesting part of the signal. A solution to this problem is to remove the quasi-statically varying offset of the detector signal (for instance by means of a highpass filter) before feeding it to the ADC. In this case the use of a chopper is required in order to reconstruct the dc-component \(a_0\) of the signal.

Chopper

The chopper is an optional part of the setup. It is needed when the intensity of the background light is not constant, or when the detector exhibits a drift in its offset. Figure 5.2 illustrates how a chopper in combination with some offset-removing circuit can be used to map the interesting part of the signal onto a more desirable interval, while not discarding any information about \(a_0\).

The hardware part supplies a TTL signal to the chopper (which we shall refer to as the chopper control signal), and the chopper is required to chop the light with the frequency of this signal. At present a mechanical chopper is used, consisting of a rotating blade with slits, the speed of rotation being determined by a Phase Locked Loop (PLL) circuit. The TDL which will be used in the future has a chopper function built in, which can be controlled by an external TTL signal.

Note that if a chopped laser beam is being measured by a detector, the output signal will have a high part (corresponding with an unblocked beam), a low part (corresponding with a blocked beam), and a transient part. The transient is not infinitely steep because of the finite bandwidth of the detector, and (for a mechanical chopper) because the laser beam has a nonzero width, so that the beam can also be partially blocked.
Figure 5.2: (a) Example of the intensity at the detector resulting from modulation of the laser beam by the PEM. (b) Intensity at the detector resulting from modulation by a chopper. The transients are not infinitely steep for a number of reasons (see text). (c) Intensity at the detector resulting from modulation by the PEM and a chopper. Note that at the chopper transitions the signal is distorted due to the non-infinite steepness of the chopper transient. (d) The output voltage of the detector, (e) The output voltage after some offset-removing circuit, which effectively subtracts the momentary average value from the signal. The parameter \( a_0 \) can be recovered by subtracting the signal value of the 'chopper closed' section from the average signal value of an integral number of (undistorted) PEM periods stemming from the 'chopper open' section.

### 5.1.1 Measurements with and without chopper.

For measurements without a chopper it is possible to calculate a \((\Delta, \psi)\)-pair for every PEM period, corresponding with a temporal resolution of \(1/f_{PEM}\). If this best performance with respect to the temporal resolution is not required, averaging of subsequent PEM periods can be used to improve the signal to noise ratio.

For measurements including a chopper it makes sense to calculate a \((\Delta, \psi)\)-pair for every chopper period, thereby averaging all undistorted PEM periods stemming from the chopper open zone (see fig. 5.3) and using the signal values in the chopper closed zone to correct for the detector offset and background light. For measurements including a chopper the minimum temporal resolution is then \(1/f_{chop}\). Several chopper periods can be averaged to increase the signal to noise ratio, at the cost of the temporal resolution.

The chopper frequency \(f_{chop}\) is an integral factor \(Q_{chop}\) smaller than the PEM frequency, so that an integral number \(Q_{chop}\) of PEM periods fit into one chopper period.\(^1\) The number \(U_{chop}\) of undistorted PEM periods in the chopper open zone is equal to the number of PEM periods in the chopper closed zone that can be used for determining the detector offset. The signal in the transient zone is neglected. Figure 5.3 illustrates parameters and definitions pertaining to a chopper period.

\(^1\)The term 'PEM period' is used strictly as a unit of time in this sense, since in the chopper closed zone obviously no signal resulting from the modulation by the PEM is present.
Figure 5.3: Definitions and parameters pertaining to a chopper period. All zones have a width of an integral number of PEM periods. The actual transient of the chopper will in general not have a width of an integral number of PEM periods. For this reason the transient zone will, besides the transient, also contain some small regions that correspond with a completely unblocked or completely blocked beam.

5.2 Hardware Setup

The hardware of the acquisition part has been implemented using the PhyDAS system, which is a system for the acquisition of data and control of experiments, that has been developed at the Department of Physics of the TUE. The PhyDAS system consists of (see also fig. 5.4)

- Interfaces, which are hardware components that communicate with the experiment, such as for instance ADC’s and digital to analog converters (DAC’s).
- A PhyDAS crate, which is a self-powered unit that can accommodate several interfaces. The interfaces can easily be placed or removed without dismanteling the crate. Once an interface is placed in the crate, it is connected to the PhyBUS.
- A PhyBUS, which is a bus containing 32 data lines, 12 address lines and some control lines. Each interface in the crate occupies a certain range of addresses on the PhyBUS.
- A connection to a personal computer (PC). This connection consists of a PhyBUS-Highway converter which is inserted into the PhyDAS crate, a Highway-PCI converter inserted into a PCI-slot of the PC, and an interconnecting Highway cable.

Data Transport

Data can be transported in the PhyDAS system in several ways. Firstly, data that has been acquired by an interface can be transported from the interface via the PhyBUS and the Highway connection to the PC memory. Alternatively, data can be transported from the PC memory to an interface. This may concern data that is to be used for controlling the experiment (such as a data which is used as input for a DAC), or data that is used for controlling the operation of the interface.

Furthermore, some interfaces offer the possibility to transfer data directly to another interface via a fast 32-bit point to point connection, called the PhyPAD.

5.2.1 The interfaces

This section gives a brief description of the interfaces that have been used in the hardware setup. Most interfaces can be configured via software in several ways, corresponding with
different functionalities or modes of operation. Only the functionalities or modes that have been used in this particular hardware setup are described. For more complete information references are included.

**Analog Signal Recorder**

The analog signal recorder [8] (ASR) is a dual channel 12-bits bipolar ADC with onboard memory and averaging possibilities intended for the 'averaging' of subsequent periods of a periodic signal. This averaging of subsequent periods is often desired to improve the signal to noise ratio. The averaging function of the ASR is determined by two parameters called the sample count $N_{\text{asr}}$ and the average count $M_{\text{asr}}$. The ASR performs an A/D conversion of $M_{\text{asr}}$ series of $N_{\text{asr}}$ samples of an analog input signal. The corresponding samples of each series are internally added (see fig. 5.5a) and the results are stored in an onboard memory. The number of times that this total sequence is repeated is determined by the repeat count $R_{\text{asr}}$. Note that averaging in this way only produces sensible results if the sample frequency is such that $N_{\text{asr}}$ corresponds with the number of samples per period of the periodic signal to be recorded. Furthermore note that the 'averaging' consists only of either addition or subtraction, the division by the number of added or subtracted numbers is omitted. The parameters $N_{\text{asr}}$, $M_{\text{asr}}$, and $R_{\text{asr}}$ can be controlled by software, and must be set prior to a measurement (during the initialisation of the ASR). The onboard memory is used in an interleaved fashion (see fig. 5.5b). After a sequence of $N_{\text{asr}}$ times $M_{\text{asr}}$ samples has been averaged and stored into one memory block, this block must be read (either via the PhyBUS or via the PhyPAD) while acquisition continues in the other block. Each block is thus alternately being filled or being read. This process continues a number of times determined by the repeat count $R_{\text{asr}}$. The word size that is used for storing the added sample values can be 2 bytes or 4 bytes. If the word size is chosen to be 2 bytes $M_{\text{asr}}$ must be less than or equal to 16, since the addition of more than 16 12-bit sample values might create an overflow. Apart from analog inputs the ASR has several inputs for TTL signals:
Figure 5.6: Schematic representations of the interfaces, their mutual communication signals, and their communication signals with the experimental setup. The memory MEM is either a MPM or a combination of 2 dual ported memories.

- Clock input. Each rising edge of the signal on the clock input starts an A/D conversion. The frequency of the TTL signal on the clock input therefore determines the sample frequency.
- Trigger input. The ASR starts its operation after the first trigger (rising edge of the TTL signal) has been received.
- Gate input. If the signal on the gate input is at a low level, pulses on the clock input are neglected and no A/D conversions are performed.
- Add/Subtract input. This input determines whether addition or subtraction is used for the averaging function. This input is not evaluated constantly, but only after $N_{asr}$ samples have been taken, and after the first trigger has been received.

It is possible to control via software whether the gate and add/subtract inputs are used or disregarded.

Dual ported memory

The purpose of a dual ported memory [9] can be to temporarily store data acquired by other interfaces. This data is supplied via the PhyPAD. A combination of two dual ported memories can be configured as an interleaved memory. While one memory is being filled via the PhyPAD, the other one can be read via the PhyBUS. This interface is quite obsolete nowadays and the dual ported memories in the acquisition part have therefore been replaced by a multi purpose memory.

Multi Purpose Memory

The multi purpose memory [10] (MPM) combines an interleaved memory with advanced programmable data processing possibilities. In the current hardware setup the MPM is used only as an interleaved memory. The total amount of physical memory present can be up to 64 Mbyte. Currently 2 blocks of 2 Mbyte are present. It is possible to choose via software a block size for the interleaved operation of the memory. 'Switching' from one memory block to another occurs when the memory block is filled up to the block size.
Frequency Multiplier Unit

The frequency multiplier unit (FMU) produces a TTL signal with a frequency that is a factor 128 larger than the frequency of the supplied input signal. The reference signal supplied by the PEM is fed to this unit, and the produced signal is fed to the clock signal generator (see fig. 5.6). The FMU is not a standard PhyDAS interface, but has been specifically designed for this hardware setup. See appendix B for details.

Clock Signal Generator

The clock signal generator [11] (CSG) produces the signals required by the TTL inputs of the ASR. This comprises a signal for the trigger input, the clock input, the gate input and the add/subtract input. Furthermore, this unit creates the chopper control signal (see fig. 5.6). The frequency of this chopper control signal provided by the CSG is 12 times smaller than the PEM frequency \( Q_{chop} = 12 \). The CSG is a standard PhyDAS interface, but the particular characteristics of the generated output signals have been specifically designed for this hardware setup. The signals generated by this interface are discussed in more detail in the section 5.2.4.

5.2.2 Acquisition configuration

Figure 5.7 shows the general acquisition flow. This flow comprises a number of processes:

- A/D conversion of analog signal values.
- Some simple data processing by the ASR. This processing reduces the data throughput over the PhyPAD, while no vital information is discarded. The three data reduction configurations that are used are discussed in section 5.2.3.
- Transfer of the reduced data-flow via the PhyPAD to the interleaved PhyBUS memory. This PhyBUS memory can be a MPM or a combination of two dual ported memories.
- Transfer of the data from the interleaved phybus memory over the highway cable to the PC memory. While one memory block is being filled via the PhyPAD the contents of the other memory block are:
  - Transferred via the PhyBUS and the highway cable to the PC memory.
  - Processed. This processing can comprise calculating, plotting and storing on disk of \( \Delta \) and \( \psi \) values.

Conceptually the acquisition configuration can be considered a concatenation of interconnected (temporary) data buffers. From front-end to back-end the the connections become
increasingly slower. For the PhyPAD the time required for a 32-bits transfer is approximately 150 ns and for the highway this time is approximately 800 ns.\textsuperscript{2} The connection between the PC memory and hard disk is much slower.

In contrast the concentration of the desired information with respect to the data increases as the data progresses from front-end to back-end. First, some data reduction is applied by the ASR before the data is sent to the PhyBUS memory. Secondly, a data 'unit' consisting of a number of samples corresponding with a PEM period (or with several averaged PEM periods) is transformed into a data unit consisting of a single ($\Delta$, $\psi$)-pair before this data unit is sent from the PC memory to hard disk.

5.2.3 Data reduction by the ASR

Before data is transferred over the PhyPAD the ASR applies some form of data reduction. All methods of data reduction use averaging of several PEM periods, by choosing the parameter $N_{asr}$ equal to the number of samples that is taken per PEM period. For measurements without a chopper the data reduction consists only of this averaging of $M_{asr}$ PEM periods.

For measurements including a chopper two configurations are used, called the integral offset correction configuration and the differential offset correction configuration.

**Integral offset correction configuration**

The integral offset correction configuration involves the use of the gate input of the ASR to suppress the transient zone of a chopper period. The gate signal provided by the CSG is at a low level during the transient zone of a chopper period (see fig. 5.8). Therefore no samples are taken during the transient zone, resulting in a data reduction without loss of information. Furthermore the $U_{chop}$ PEM periods stemming from the open zone and the $U_{chop}$ PEM periods stemming from the closed zone are averaged separately, by choosing $M_{asr}$ equal to $U_{chop}$. The data throughput is thus reduced by a factor $Q_{chop}/2$ (see fig 5.9b). Afterwards the (added) sample values corresponding with the chopper closed zone can be averaged by a program on the PC to obtain a representative value for the entire chopper closed zone, which can be subtracted from the dc-component $a_0$ obtained from the chopper open zone.

\textsuperscript{2}By the 'highway transfer time' is meant the time required to transport one 32-bit longword from the PhyBUS memory to the PC memory. This time not only depends on the physical speed of the highway cable and PhyBUS, but also to a large extent depends on the used PC and its operating software.
Figure 5.9: (a) Schematic representation of one chopper period as it is sampled, (b) Schematic representation of the reduced data for the integral offset correction configuration, (c) Schematic representation of the reduced data for the differential offset correction configuration.

Differential offset correction configuration

The differential offset correction configuration combines the suppression of the chopper transient zone with the use of the add/subtract input of the ASR, in order to obtain a point by point subtraction of the chopper closed zone from the chopper open zone. The add/subtract signal provided by the CSG is at a low level (corresponding with addition) during the chopper open zone and at a high level (corresponding with subtraction) during the chopper closed zone.\(^3\) See also fig. 5.8. The parameter \(M_{asr}\) is chosen equal to \(2U_{chop}\). The first \(U_{chop}\) PEM periods correspond with the chopper open zone and coincide with a low level add/subtract signal and are thus added. Next, some PEM periods corresponding with the transient zone and coinciding with a low level gate signal are neglected. The next \(U_{chop}\) PEM periods correspond with the chopper closed zone and coincide with a high level add/subtract signal and are subtracted from the \(U_{chop}\) already acquired and added PEM periods stemming from the open zone. This results in one averaged PEM period which is already corrected for the offset. The data throughput is thus reduced by a factor \(Q_{chop}\) (see fig 5.9c). It is possible to add several (say \(k\)) chopper periods by choosing \(M_{asr} = k2U_{chop}\). This adding of several chopper periods is not possible with the integral offset correction configuration. The disadvantage of the differential offset correction configuration is that the offset subtraction is point by point. From each sample in the chopper open zone the corresponding sample in the chopper closed zone is subtracted. The latter values are not all the same because of the presence of noise. As a consequence the noise in the chopper closed zone is added to the chopper open zone, as opposed to the case of the integral offset correction configuration, where a representative averaged value for the entire chopper closed zone can be subtracted.

5.2.4 Timing aspects of chopper related signals.

The terms 'open zone', 'closed zone' and 'transient zone' relate to the intensity of a laser beam resulting from modulation by a chopper. We call the variation of this intensity as a function of time the chopper status signal. The chopper status signal has a high level when the laser beam is not blocked, a low level when the beam is blocked, and some value in between when the beam is partially blocked.\(^4\) The chopper status signal has the same frequency as the chopper control signal which is supplied to the chopper unit, but in general there will be a phase difference between the two signals. As can be seen in fig. 5.10 this

\(^3\)This is not entirely correct, in reality the add/subtract signal is a little bit shifted in phase so that at the moment of evaluation the signal has the proper level (see fig. 5.8).

\(^4\)The term 'blocked' is used in a general sense, not only applicable to mechanical choppers
Figure 5.10: Schematic representation of the chopper status signal and some of the signals generated by the CSG. The input frequency of the CSG is 128 times higher than the PEM frequency. The clock signals that are generated comprise signals with a frequency 16, 32, 64 and 128 times larger than the PEM frequency, and are not displayed in the figure.

Figure 5.11: Schematic representation of a chopper. Only one slit is drawn. The position of the laser beam is indicated by a dot. By moving the chopper in either of the indicated directions the phase difference between chopper control signal and chopper status signal is altered.

Phase difference destroys the synchronization between the different zones of a chopper period and the gate and add/subtract signals. For a mechanical chopper this problem can be solved by moving the chopper with respect to the laser beam (see fig. 5.11), which alters the phase difference between chopper control signal and chopper status signal. For the chopper function incorporated in the TDL this is not possible. If the chopper function of the TDL is used, the only means to restore the synchronization is to alter the time delay between a trigger pulse of the trigger signal and a rising edge of the chopper control signal. A future version of the CSG will have the possibility to control this time delay via software.

5.2.5 Bottlenecks

Typical examples of hardware characteristics that can be bottlenecks are memory sizes and connection speeds. Whether a certain characteristic of the acquisition configuration constitutes a bottleneck depends on the user wishes, such as temporal resolution, total measurement time, the number of samples per PEM period, presence or absence of a chopper, and on the PEM frequency.

The ASR memory, PhyBUS memory and PC memory all transfer data and receive data. In addition the data in the ASR and PC memories is processed resulting in a data reduction. With the transfer of data from one module to the next an available time and a duration time can be associated. A bottleneck occurs when the available time is less than the duration time. As an example we consider an (interleaved) PhyBUS memory, consisting of two blocks of $n_{bytes}$ bytes. For a measurement without a chopper the rate $v_{fill}$ at which one memory
block is filled via the PhyPAD is

\[ v_{\text{fill}} = f_{\text{PEM}} N_{\text{asr}} S_{\text{size}} / M_{\text{asr}} \]

where \( S_{\text{size}} \) is the word size, which can be 2 or 4 bytes, depending on the value of \( M_{\text{asr}} \). The time it takes to fill the memory block is thus

\[ t_{\text{fill}} = \frac{n_{\text{bytes}}}{v_{\text{fill}}} = \frac{n_{\text{bytes}} M_{\text{asr}}}{f_{\text{PEM}} N_{\text{asr}} S_{\text{size}}} \]

This time is equal to the time \( t_{\text{available}} \) that is available for the other memory block to transfer its data to the PC memory. The time \( t_{\text{duration}} \) needed to transfer a memory block to the PC memory is

\[ t_{\text{duration}} = \frac{n_{\text{bytes}}}{v_{\text{highway}}} \]

where \( v_{\text{highway}} \) is the speed of the highway connection expressed in bytes per second. The available time should be larger than the duration time:

\[ t_{\text{available}} > t_{\text{duration}} \Rightarrow \frac{M_{\text{asr}}}{f_{\text{PEM}}} > \frac{N_{\text{asr}} S_{\text{size}}}{v_{\text{highway}}} \]  \hspace{1cm} (5.1)

This illustrates that the connection speed of the highway cable can be a bottleneck if the user wants a temporal resolution that conflicts with condition (5.1). However, if the total measurement time is limited so that all data required fits into the Phydas memory, the same connection speed is not a bottleneck. If real-time monitoring of the results is required, a processing time has to be added to the duration time, resulting in a more strict requirement for the minimum value of the temporal resolution.
Chapter 6

Software Setup

This chapter focuses on the structure of the software and program flow, and not so much on algorithms and implementation details. However, to elucidate some structures, or the reason they have been employed, examples are used.

The first part of this chapter provides a task description of the software, thereby splitting the primary task in some subprocesses. After that the implementation (on a macro level) is discussed. Finally, some tools intended for the developer are discussed.

6.1 Task description

The primary task of the software is to enable a user to perform the desired measurement. The user wishes with respect to a measurement comprise a number of items:

- Measurement type: measurement including a chopper, or not including a chopper.
- Measurement time.
- Temporal resolution.
- Parameters pertaining to real-time monitoring of the results, such as the time between two screen updates, and the number of points to plot for each screen update.

Apart from this, the user may wish to conduct a wavelength scan, which means that a measurement defined by the above items must be repeated for a number of wavelengths. Some combinations of the user wishes may cause a bottleneck to become effective. The software should preclude the possibility of the user selecting such a combination.

In order to execute the measurement, first the user wishes have to be translated into a hardware state and a software state. A hardware state is a collection of parameters that determine how the interfaces are initialized. A software state is a particular choice for the measurement flow. Fig. 6.1 gives an impression of a measurement flow. Important subprocesses of the measurement flow are the acquisition flow and the processing flow.

6.1.1 Acquisition flow

The task of the acquisition flow is to retrieve data blocks that have been acquired by the interfaces. With respect to the number of data blocks that are retrieved, and the location from which the blocks are retrieved a number of situations can be distinguished:
Figure 6.1: Example of the structure of a measurement flow

- The user wishes are such that the PhyPAD is not fast enough to enable interleaved operation. In this case only one data block is retrieved directly from the ASR memory.
- The user wishes are such that the highway connection is not fast enough to enable interleaved operation. In this case only one data block is retrieved from the PhyBUS memory.
- The user wishes are such that real-time monitoring of the results is possible. In this case several blocks are retrieved from the PhyBUS memory.

After a block has been retrieved it is processed by the processing flow, and the results are displayed. If real-time monitoring of the results is possible, the block size for the interleaved operation of the PhyBUS memory is chosen in accordance with the desired screen update time.

6.1.2 Processing flow

The processing flow is a subprocess of the acquisition flow, and its task is to convert a data block consisting of (averaged) sample values to an array of \((\Delta, \psi)\)-pairs. The number of sample values that corresponds with one \((\Delta, \psi)\)-pair depends on the number of samples \(N\) per PEM period, and on the kind of data reduction that has been applied by the ASR (integral or differential offset correction configuration). The processing flow comprises the following subprocesses:

- Calculation of fourier coefficients
- Phase shift correction. A phase difference between the PEM reference signal and the retardation will cause a phase shift of all fourier components. Furthermore the amplifying electronics can introduce phase shifts. These phase shifts are corrected by this process.
- Calculation of \(I_s\) and \(I_r\) from the fourier coefficients. This calculation depends on the model used for the retardation (see section 3.2).
- Calculation of \(\Delta\) and \(\psi\) from \(I_s\) and \(I_r\). This calculation depends on the configuration used for the angular settings \((P, M, A)\) (see section 3.2).

6.1.3 Calibration measurements

Before the experimental setup in conjunction with the acquisition system can be used for measurements, some calibration measurements must be performed. These calibration measurements involve the acquisition of data, however, the objective is not to obtain \(\Delta\) and \(\psi\)
from this data, but to obtain proper values for some parameters pertaining to the experi-
mental setup or acquisition system. An example of a calibration measurement is the calibration of
the chopper in order to provide the required synchronization between the zones of the chopper
status signal and the gate and add/subtract signals. Another example is the calibration of
phase shifts for the phase shift correction process mentioned in section 6.1.2.

6.2 Implementation

All software has been written in C, using the Labwindows/CVI environment. This is a C-
programming environment which facilitates management of user input, and the design of
graphical user interfaces. The first part of this section focuses on general aspects with respect
to user input and the way elements/structures of the C-programming language have been
used. This part also serves to introduce some (not standard) terminology. After that the
structure of the measurement flow is discussed.

6.2.1 General aspects

User input

Several types of user input can be distinguished:

- Numerical input. This concerns input of some numerical value with an intrinsic mean-
ing for instance the PEM frequency or the measurement time.
- Mode selecting input. The different (numerical) values that a mode selecting variable
can assume correspond with different modes of operation of some part of the program,
and have no intrinsic meaning. Often these values are given descriptive names in the
source code (by means of a #define compiler directive). The number of 'modes' that
can be selected is limited, and in the user interface a picklist, presenting all possibilities,
is used for mode selecting input. This picklist displays labels (descriptive names for
the options) rather than the numerical values associated with these labels.
- Action input. Action input is not stored in a variable, but invokes an action such as
the start of a measurement or the drawing of a graph.

Labwindows/CVI facilitates the management of all these types of input.

Use of functions and variables

A function can be considered as an implementation of some process. The form of the process
is defined by its input and output parameters. An implementation of a process is a particular
method of obtaining the output parameters from the input parameters. In the software the
formal parameters of a function are usually the input and output parameters of the associated
process, so that the prototype\(^1\) of a function reflects the form of the associated process.
Most functions also depend on some global variables. These global variables are usually
mode selecting variables that can be used to choose a slightly different implementation of the
process associated with the function. As a simple example we consider a function that is used
to calculate \(\Delta\) and \(\psi\) from \(I_s\) and \(I_a\) for configuration II.

\(^1\)The prototype of a function is the function name in conjunction with the list of formal parameters.
The prototype of this function is

\texttt{Icis.to_deltapsi_config2(float Ic, float Is, float *delta, float *psi)}

As discussed in section 3.2.2 two different \((\Delta, \psi)\)-pairs correspond with an \((I_c, I_s)\)-combination. For this reason the function also depends on a global mode selecting variable called \texttt{ambiguity_config2}. The value of this variable determines which of the two possible \((\Delta, \psi)\)-pairs is chosen. Changing the value of this variable thus (slightly) changes the implementation of the process.

\subsection*{6.2.2 Functionalities}

To emphasize the concept of a separation between \textit{form} and \textit{implementation} of a process a construction called \textit{functionality} has been designed. A functionality is the ensemble of a collection of functions with an identical formal parameter list, a function pointer variable, and an array of structures containing some administrative information for each function (see fig. 6.2). A function pertaining to a functionality is called a \textit{member} or \textit{member function} of that functionality. The measurement flow only contains the function pointers associated with functionalities, and not the function names of the member functions. The function pointers reflect the form of the process corresponding with the functionality, and can be made to point to any of the member functions. The member function prototypes also reflect the form of the associated process, but are a level closer to the implementation level since a particular implementation is already associated with this prototype, which cannot be changed at run-time. In contrast, the implementation associated with a function pointer is not fixed, and can be changed at run-time.

In order to be able to effectively make use of the member functions some administrative information is kept for each member. This information comprises:

- A function pointer variable, containing the address of the function.
- A name. This is a string variable that is filled with the name that is used for the function in the source code.
- A label. This label can appear as an element of a picklist in the user interface, allowing the user to select a certain member of a functionality as the active one, by clicking on the label. If a member is selected the function pointer of the functionality is made to point to this member.
- A helptext. Right clicking on a label that is displayed in some picklist in the user interface displays the contents of the helptext variable.
- An initialization function pointer, containing the address of the \textit{initialization function}. If the member function depends on some global variables, the corresponding initialization function offers the possibility to set these variables. The initialization function is executed each time the user selects the member function (by clicking on the corresponding label). The initialization function can also be executed from within the developer interface (section 6.3.2). If no initialization function is required this variable is given the value \texttt{NULL}.

Fig. 6.3 shows a typical example of the initialization of the administrative information with respect to a member function.
I* collection of functions */

int Icis_to_deltapsi_empty(float Ic, float Is, float *delta, float *psi)
{
    return 0;
}

int Icis_to_deltapsi_config2(float Ic, float Is, float *delta, float *psi)
{
    /* function body */
}

int Icis_to_deltapsi_config3(float Ic, float Is, float *delta, float *psi)
{
    /* function body */
}

I* function pointer */

int (*Icis_to_deltapsi)(float, float, float *, float *);

I* structure with administrative information */

typedef struct {
    int (*functionpointer)(float, float, float *, float *);
    char name[100];
    char label[100];
    char helptext[1000];
    int (*initfunctionpointer)(void);
} Icis_to_deltapsi_functionality_struct;

I* array of structures with administrative information */
/* each element corresponds with a member function */

Icis_to_deltapsi_functionality_struct

Icis_to_deltapsi_functionality[max_numb_members];

Figure 6.2: Elements pertaining to a functionality. The information in this example applies to the
Icis_to_deltapsi_functionality but the information is similar for all functionalities.

Icis_to_deltapsi_functionality[index].functionpointer = &Icis_to_deltapsi_config2;
strcpy (Icis_to_deltapsi_functionality[index].name, "Icis_to_deltapsi_config2");
strcpy (Icis_to_deltapsi_functionality[index].label, "Config 2");
strcpy (Icis_to_deltapsi_functionality[index].helptext,
"Config 2, P=Pi/4, M=0, A=Pi/4");
Icis_to_deltapsi_functionality[index].initfunctionpointer = &config2_initfunc;

Figure 6.3: Example of initialization of an element of the array with administrative information of a
functionality.
All functionalities have a somewhat anomalous member called the *empty member*. The only statement in the function body of this member is the *return* statement. The function of the empty member will become clear later.

**Use of functionalities.**

The measurement flow consists of a lot of processes that require different implementations for different situations. Sometimes two implementations of a process differ only slightly from each other. In this case a global mode selecting variable is used. It is also possible that the methods used in different implementations of a process are quite different. In this case a functionality is used, where each implementation corresponds with a member of the functionality. Before the measurement flow is executed, each function pointer is initialized with the address of the appropriate member function.

**6.2.3 Implementation of the measurement flow**

In order to enable a measurement, user input is translated into a software and hardware state. The information in the software state is used to properly initialize the measurement software (the function pointers in the measurement flow) before it is activated, and the information in the hardware state is used to properly initialize the hardware before it is activated. The measurement software can be activated through a call to the *execute_measurement* function. The input parameters of this function comprise

- The hardware state. The parameters constituting the hardware state are used as input parameters for functions called by the *execute_measurement* function that initialize the hardware.
- An array of structures for the wavelength scan. Each element of the array corresponds with a wavelength, and contains all information that is required to set this wavelength.
- The number of wavelengths of the wavelength scan.

Figure 6.4 shows a schematic representation of the measurement flow. All names of function pointers associated with a functionality are followed by the names of the member functions that this function pointer can point to. Indentation of function pointer names indicates a nesting of processes. *All* members (except the empty member) of a functionality invoke the function pointers that are displayed one indentation level to the right of the function pointer of the functionality. A member of a functionality thus consists of an implementation part specific to that member and a part that invokes some subprocesses, common to all members of the functionality. Making the function pointer of a functionality point to another member, changes the implementation of the measurement flow. If the function pointer of a functionality is made to point to the empty member (which only contains a *return* statement in its function body), the process associated with this functionality and all subprocesses are deactivated. In this case we say that the *morphology* of the measurement flow is altered. This deactivation mechanism makes the measurement flow very flexible.

**Discussion of some functionalities**

Most of the names in fig. 6.4 are self-explanatory. However, some elucidating remarks will be mentioned here.
execute_measurement

<begin wavelength loop>
set_and_wait_for_wavelength
initialize_asr_for_acquisition
initialize_memory_for_acquisition
acquire_and_process_data_blocks
wait_for_external_trigger

<begin acquire data block loop>
services_rawdata_block
process_datablock

calculate_fourier_coefficients
phase_shift_correction
fourier_coefficients_services
fourcoefto_10lc1B
lcls_to_deltapsi
plotting_services_sw
flushing_services_sw

<end acquire data block loop>

plotting_services_mw
flushing_services_mw

<end wavelength loop>

Figure 6.4: Schematic representation of the measurement flow.
The `set_and_waitfor_wavelength` process is meant for measurements in which the TDL is used. If the used laser allows no wavelength setting this process is deactivated. Only the empty member of this functionality has been implemented as yet, but the functionality is included for future use.

The `acquire_and_process_data_blocks_from_file` member can be used for debugging purposes. When this member is active, the samples are retrieved from a file. By activating this member all hardware sources of error are deactivated, which can be of help when trying to isolate a source of error.

The `wait_for_external_trigger` process can for instance be used to synchronize the start of the measurement with the activation of a plasma. In this case the device activating the plasma should provide some sort of trigger pulse. Only the empty member of this functionality has been implemented as yet, but the functionality is included for future use.

The members of the `calculate_fourier_coefficients` process all yield the same results, but differ in speed. The vanilla member uses the plain vanilla expressions for the DFT to calculate the fourier coefficients. The strawberry member uses some algorithm that reduces the number of multiplications required to obtain the results, and is the fastest one if the number of components to be calculated is small. The melon member applies a Fast Fourier Transform, and is the fastest one if the calculation of a lot of components is required. The software automatically chooses the most expedient member for a particular situation.

The `services_rawdata_block` process is not activated during a normal measurement. It is activated during the calibration of the chopper, in which case the raw data signal covering a chopper period is displayed, so that the synchronization between chopper zones and add/subtract and gate signals can be checked. Furthermore, this process can be activated for debugging purposes with respect to the experimental and/or hardware setup.

The members of the `calculate_fourier_coefficients` process all yield the same results, but differ in speed. The vanilla member uses the plain vanilla expressions for the DFT to calculate the fourier coefficients. The strawberry member uses some algorithm that reduces the number of multiplications required to obtain the results, and is the fastest one if the number of components to be calculated is small. The melon member applies a Fast Fourier Transform, and is the fastest one if the calculation of a lot of components is required. The software automatically chooses the most expedient member for a particular situation.

The `services_rawdata_block` process is not activated during a normal measurement. It is activated during the calibration of the chopper, in which case the raw data signal covering a chopper period is displayed, so that the synchronization between chopper zones and add/subtract and gate signals can be checked. Furthermore, this process can be activated for debugging purposes with respect to the experimental and/or hardware setup.

The `fourier_coefficients_services` process is deactivated during a normal measurement. It is activated during the calibration of phase shifts.

If the `plotting_services_sw` process is activated, plotting of results occurs immediately after the acquisition and processing of a data block. If the `flushing_services_sw` process is activated, flushing of results to hard disk occurs immediately after the acquisition and processing of a data block. In this case the `plotting_services_mw` and `flushing_services_mw` processes are deactivated. If the 'connection' between hard disk and PC memory is not fast enough, all data blocks are stored in PC memory, and flushing occurs when all data blocks for one wavelength have been acquired. In this case the `flushing_services_sw` process is deactivated, and the `flushing_services_mw` process is activated.

### 6.2.4 Functionality management

Some functionalities correspond to items that are of interest to a user, such as for instance the `IcIs_to_deltapsi` functionality, that corresponds to the configuration of angular settings. The active member of these kind of functionalities is selected by the user, by clicking on a label in a picklist. Other functionalities are hidden from the user, and the members are activated or deactivated by the software. This is, e.g., the case for the `acquire_and_process_data_blocks` functionality. For the selection of functionalities one of the `functionality management functions` is used. An example of such a function is the `select_function_via_name` function, which takes the name of the member (that is the contents of the name field of the structure with administrative information for that member) as argument. For instance a call like

```plaintext
select_function_via_name("process.datablock.unchopped");
```
Figure 6.5: The processing flow and simulation flow. The names are the names of the function pointers pertaining to a functionality.

makes the function pointer `process_datablock` point to the member with the name `process_datablock_unchopped`. In addition a structure that keeps track of which members are selected is updated. This information is required by the developer interface (section 6.3.2), from within which an overview of the active functions can be displayed.

### 6.3 Developer Tools

Since the structure of the measurement flow is very general, adding new features to the software can often be achieved by adding a new member to a functionality. Supplying the administrative information for the member (see fig. 6.3) is all that is required to make the member appear as an option in the appropriate places of the user interface (and developer interface, see section 6.3.2).

To facilitate the maintainability of the software some developer tools are provided. For a developer wanting to add or modify ellipsometric models or calculations the simulation flow can be of help. For a developer seeking to enhance acquisition related issues the developer interface is a practical tool.

#### 6.3.1 Simulation flow

In addition to the processing flow a simulation flow is provided (see fig. 6.5). Whereas the processing flow transforms an array of sample values into a \((\Delta, \psi)\)-pair, the simulation flow starts out with a \((\Delta, \psi)\)-pair and produces an array of sample values. The user interface contains a simulation section that accepts a \((\Delta, \psi)\)-pair as input, executes the simulation flow, and uses the output of the simulation flow as input for the processing flow. The input \((\Delta, \psi)\)-pair and the output \((\Delta, \psi)\)-pair are displayed. The entire \((\Delta, \psi)\)-plane can be 'scanned' in this way. Furthermore, this simulation section offers the possibility to set simulation parameters, such as the retardation depth and the number of samples per PEM period, and to choose the active members of the functionalities constituting the processing and simulation flow. If a new member is added to a functionality of the processing flow, it can be tested by adding an 'inverse' function to the appropriate functionality of the simulation flow. Furthermore, the influence of noise on the accuracy of the results can be investigated by activating the `give_me_noise` functionality.

#### 6.3.2 Developer interface

User wishes with respect to a measurement are translated into a hardware state and software state. The parameters pertaining to these states determine the operation of the hardware
and measurement flow. These parameters are of vital importance to a developer, but only confusing to a user. For this reason these parameters are hidden to the user. However, this hiding of these parameters complicates the debugging of the software. For this reason the developer interface has been designed.

The developer interface consists of a window that can be activated from almost every section of the program. This window displays the current morphology of the software state (that is the active processes in the measurement flow), using indentation to emphasize the structure of this flow (see fig. 6.6). Also the hardware state parameters are listed, and can be changed. Clicking on a function in the software state display pops up a window that lists all members belonging to that functionality, and offers the possibility to select any of them (including the empty member) as the active one. It is also possible to execute the initialization function of an active member.

The developer interface thus offers the possibility to replace an implementation of a process by another implementation, or to deactivate the process. The hardware state parameters (which are used as input parameters of the active functions) can be viewed and modified, and the global mode selecting variables of the active functions can be modified by executing the corresponding initialization function.

The developer interface is typically invoked just before the execution of the measurement flow. The automatically chosen hardware and software state can be viewed, and altered if judged necessary (for instance for debugging purposes).
Chapter 7

Tests

The data acquisition part and experimental setup both consist of modules, that communicate with each other. For the system as a whole to function properly, each individual module has to function properly. This chapter describes some tests that have been performed on some of the modules, and on the functioning of the total setup. Also the performance of the setup with respect to feasible temporal resolutions etc. will be briefly discussed.

7.1 Experimental setup

The only element of the experimental setup that showed anomalies during testing is the PEM.

PEM

The PEM that has been used is a PEM-90 from Hinds Instruments Inc., with a modulation frequency of 31.1 kHz. This modulator includes a control unit, which allows the retardation depth to be set, and displays the value for the setting. This PEM has been tested by monitoring the detector intensity and the supplied PEM reference signal on an oscilloscope, using configuration II for the angular settings. No sample was used, which corresponds to $\Delta = 0$ and $\psi = \pi/4$. The harmonic model for the modulator (page 14) has been used in the simulation flow (section 6.3.1) to predict the detected intensity. It was found that for each setting of the retardation depth, the simulated intensity and the detected intensity did not match. However, by using a retardation depth in the simulation approximately 1.3 times larger than the setting displayed on the control unit, a match could be obtained. This factor 1.3 is only a rough indication of the relation between the displayed setting and the setting obtained via the simulation. The actual behavior of the PEM exhibited some slight variations from day to day. It was concluded that the setting displayed on the control unit is incorrect.

With respect to the reference signal it was found that the phase difference with the retardation can vary slightly ($\pm 5\%$ of a PEM period) from time to time, and that this phase difference depends on the setting for the retardation depth. Also it was found that the control unit failed to produce a reference signal if the setting for the retardation depth was chosen smaller than approximately 3 radians.
Figure 7.1: Testing of the suppression of the transient of a chopper period through the use of the gate input of the ASR. (a) Gate input not used, (b) Gate input used.

7.2 Hardware setup

The modules of the hardware setup are the ASR, the MPM, the FMU and the CSG. The FMU and the CSG have been tested by monitoring the signals they produce with an oscilloscope. The ASR has been tested by using a signal supplied by a function generator as analog input. It was found that small input voltages (in the interval (-10 mV, 10 mV)) occasionally generate spikes. The gate and add/subtract inputs have been tested using the function generator signal as well. Furthermore, they have been tested using the detector signal. (see fig. 7.1)

7.3 Software setup

The modules of the software setup are the processes of the measurement flow. The acquire_and_process_data_blocks_from_file function allows the measurement flow to be tested independently of the hardware. The processing flow has been tested by using a file with \((\Delta, \psi)\)-pairs as input for the simulation flow, and using its output as input for the processing flow. Fig. 7.2 shows an example of the results.

Time testing

If real-time monitoring of the results is required, a limited time is available for the processing flow to finish its required actions. The time needed to process a data unit corresponding to one \((\Delta, \psi)\)-pair is dependent on factors such as the number of samples per PEM period, the number of frequency components to be calculated and the type of PC that is used. The software provides an easy way to test duration times of the processing flow. Table 7.1 gives an impression of some of these duration times for the used PC (a Pentium II, 166 Mhz, 128 Mbyte internal memory).

7.4 Total setup

Performance

This section will give an impression of the performance of the total setup that is present. The PEM operates at a frequency of 31.1 kHz. Therefore, the best temporal resolution is 32 \(\mu s\). With this temporal resolution real-time monitoring of the results is not possible, and the number of data points is limited by the size of the PhyBUS memory, which currently consists
Figure 7.2: Results of testing the processing flow for configuration II. (a) A collection of \((\Delta, \psi)\)-pairs that have served as input for the simulation flow. (b) The corresponding collection of \((\Delta, \psi)\)-pairs produced by the processing flow. Note that from the two possible \((\Delta, \psi)\)-pairs (because of the inherent ambiguity of configuration II), the one in the left half plane has been chosen. (c) The results when the noise generation in the simulation flow has been activated. Note that the accuracy of the results depends on the location in the \((\Delta, \psi)\)-plane.

<table>
<thead>
<tr>
<th>(N)</th>
<th>(n)</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>3</td>
<td>0.38</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>0.43</td>
</tr>
<tr>
<td>128</td>
<td>3</td>
<td>0.55</td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 7.1: Some duration times for different values of the number of samples \(N\) per PEM period, and the number of frequency components \(n\) to be calculated. The duration times are the times required to process a data block corresponding to 2500 \((\Delta, \psi)\)-pairs.
of 2 blocks of 2 Mbyte. If 128 samples per PEM period are used the maximum number of
data points for this temporal resolution is 8192, which corresponds to 0.26 seconds.
For measurements including a chopper the best temporal resolution is 0.38 milliseconds. With
this temporal resolution real-time monitoring of the results is possible if the number of samples
per PEM period is 128 or less, provided that the front-end data reduction configurations
(integral of differential offset correction configuration) are used.

Testing

The integrity of the system as a whole has been tested by performing measurements on a
calibration sample. First a calibration for the phase shift between the PEM reference signal
and the actual retardation was required. To this end a quarter wave plate \((\Delta, \psi) = (\frac{\pi}{4}, \frac{\pi}{4})\) has
been used as sample. Next, a calibration sample with \((\Delta, \psi) = (1.40 \text{ rad}, 0.63 \text{ rad})\) has been
used. The experimental values for \(\Delta\) and \(\psi\) were \((\Delta, \psi)_{\text{measured}} = (1.38 \text{ rad}, 0.65 \text{ rad})\), which
is in good agreement with the actual values. These results were obtained using configuration II
and the harmonic model for the modulation. The retardation depth was 5.75 rad. A chopper
and the corresponding data reduction configurations were used. The number of samples per
PEM period was 128. Figure 7.3 gives an impression of the acquired data.
Chapter 8

Conclusions and Recommendations

This chapter lists some conclusions that can be drawn from the previous chapters, and discusses some suggestions for further developments.

8.1 Conclusions

The main conclusion that can be drawn is that a new implementation of a data acquisition and processing system for PME setups has been realized, which appears to operate correctly and satisfies the design specifications.

The new system has been realized by implementing changes in software and hardware.

8.1.1 Hardware

The changes in the hardware comprise:

- Replacement of obsolete PhyDAS interfaces by newer models. The old type ASR has been replaced by a newer model, and a combination of 2 dual ported memories has been replaced by a MPM.

- More efficient use of PhyDAS interfaces. The add/subtract and gate input of the ASR have been used to enable front-end data reduction for measurements including a chopper, which makes it possible to monitor the results of these measurements real-time. To enable the use of the gate and add/subtract inputs possible the CSG had to generate some extra signals, of which the characteristics have been designed.

8.1.2 Software

The software has been written in C, using Labwindows/CVI to create the graphical user interface. The functionality construction has been designed and used to obtain a distinct separation between form and implementation of processes, which improves the maintainability of the software. To add new ellipsometric models to the program, no knowledge of the rest of the program or of specific Labwindows/CVI user interface management facilities, is required. The new model appears automatically as an option at the appropriate locations in the user interface (and in the developer interface). In order to further enhance the developer-friendliness of the software, a developer interface has been implemented, which facilitates debugging of the software by offering the possibility of deactivating modules or replacing modules by others at run-time.
8.2 Suggestions for improvements

The data acquisition system is ready to be used to perform measurements. However, some items remain to be implemented or improved:

- The structure of the measurement flow supports wavelength scans, but not all components required to perform such scans have been implemented. A member function has to be added to the `set_and_wait_for_wavelength` functionality, which accepts a structure containing the appropriate information, and subsequently uses this information to drive a hardware component to set the desired wavelength. Furthermore, a module should be written that allows the user to compose an array of wavelengths and translates each wavelength into the appropriate information required by the `set_and_wait_for_wavelength` functionality. Furthermore, a hardware component (e.g. a multi channel DAC) has to be added which can generate the signals needed to set the wavelength.

- Presently, a new type of CSG is available that offers the possibility to control characteristics of the generated output signals via software. With this type of CSG it is possible to control the time delay between a trigger pulse and the chopper control signal by software. By replacing the present CSG with this new type CSG, the chopper calibration could be performed automatically.

- To enable the use of the TDL (and the corresponding MCT detector that exhibits a drift in its offset) a proper offset-removing circuit has to be designed and implemented. This circuit should be combined with a programmable gain amplifier, so that the analog input range of the ASR can be used efficiently irrespective of the experimental conditions.

- Some small sections of the software have not been thoroughly tested yet. A test for these sections should be performed. The software contains a module called `documentation.c` that documents what is to be tested or what can be improved with respect to these sections.
References

[1] National Instruments, Labwindows/CVI version 5.1


[11] F. van Nijmweegen, TUE, Department of Applied Physics, group BLN.
Appendix A

Derivations

A.1 Derivation of (4.15)

The expression
\[
\sum_{k=0}^{N-1} e^{i2\pi k(l-n)/N} = \sum_{k=0}^{N-1} \left(e^{i2\pi(l-n)/N}\right)^k
\]
has the form of a geometric series \( \sum_{k=0}^{p} a^k \), which has the sum
\[
\sum_{k=0}^{p} a^k = \frac{1 - a^{p+1}}{1 - a}
\]
provided that \( a \neq 1 \). The situation that \( e^{i2\pi(l-n)/N} = 1 \) only occurs when \( l - n \) is a multiple of \( N \), say \( rN \) (\( r \) integer). So we find that
\[
\sum_{k=0}^{N-1} e^{i2\pi k(l-n)/N} = \begin{cases} 
N-1 & l - n = rN \\
\sum_{k=0}^{N-1} e^{i2\pi(l-n)/N} & l - n \neq rN
\end{cases}
\]
\[
= N \sum_{r=-\infty}^{\infty} \delta(l - n - rN)
\]

A.2 Derivation of (4.22)

We start from eq. (4.20):
\[
\hat{P}(f_n) = \frac{2}{N^2} \left( \sum_{k=0}^{N-1} (g_k + e_k)c_N^{kl} \right)^2 + \frac{2}{N^2} \left( \sum_{k=0}^{N-1} (g_k + e_k)s_N^{kl} \right)^2
\]
We first consider the first term, while omitting the factor \( 2/N^2 \):
\[
\left( \sum_{k=0}^{N-1} (g_k + e_k)c_N^{kl} \right)\left( \sum_{l=0}^{N-1} (g_l + e_l)c_N^{kl} \right) = \sum_{kl} (g_k + e_k)c_N^{kl}(g_l + e_l)c_N^{kl}
\]
We split the summation $\sum_{kl}$ in two parts ($\sum_{kl} = \sum_{k=l} + \sum_{k\neq l}$):

$$\sum_{k} (g_k + e_k)^2 c_N^{n_k} + \sum_{k \neq l} (g_k + e_k)(g_l + e_l) c_N^{n_k} c_N^{n_l}$$

$$= \sum_{k} (g_k^2 + e_k^2 + 2g_k e_k) c_N^{n_k^2} + \sum_{k \neq l} (g_k g_l + g_k e_l + g_l e_k + e_k e_l) c_N^{n_k} c_N^{n_l}$$

$$= \sum_{k} (g_k^2 + e_k^2 + 2g_k e_k) c_N^{n_k^2} + \sum_{k \neq l} (g_k g_l + 2g_k e_l + e_k e_l) c_N^{n_k} c_N^{n_l}$$

Including the second term of (4.20) and including the factor $\frac{2}{N^2}$ we get

$$\hat{P}(f_n) = \frac{2}{N^2} \left( \sum_{k} (g_k^2 + e_k^2 + 2g_k e_k) c_N^{n_k^2} + \sum_{k \neq l} (g_k g_l + 2g_k e_l + e_k e_l) c_N^{n_k} c_N^{n_l} \right)$$

Furthermore $c_N^{n_k^2} + s_N^{n_k^2} = 1$:

$$\hat{P}(f_n) = \frac{2}{N^2} \left( \sum_{k} (g_k^2 + e_k^2 + 2g_k e_k) + \sum_{k \neq l} (g_k g_l + 2g_k e_l + e_k e_l) (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l}) \right)$$

Using the steps

1. Summation combination $\sum_{k=1} + \sum_{k \neq l} = \sum_{kl}$ and

   $$c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l} = c_N^{n_k^2} + s_N^{n_k^2} = 1 \quad \text{for } k = l$$

2. $\sum_{kl} a_k a_l = (\sum_k a_k)^2$

we group terms containing only signal ($g$'s) and containing also error ($e$'s):

$$\hat{P}(f_n) = \frac{2}{N^2} \left( \sum_{k} g_k^2 + \sum_{k \neq l} g_k g_l (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l}) \right)$$

$$+ \frac{2}{N^2} \left( \sum_{k} e_k^2 + 2g_k e_k \right) + \sum_{k \neq l} (2g_k e_l + e_k e_l) (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l})$$

$$\stackrel{1}{=} \frac{2}{N^2} \left( \sum_{k} g_k g_l (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l}) \right)$$

$$+ \frac{2}{N^2} \left( \sum_{k} e_k^2 + 2g_k e_k \right) + \sum_{k \neq l} (2g_k e_l + e_k e_l) (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l})$$

$$\stackrel{2}{=} \frac{2}{N^2} \left( \sum_{k} g_k c_N^{n_k^2} \right)^2 + \sum_{k} g_k s_N^{n_k^2}$$

$$+ \frac{2}{N^2} \sum_{k} e_k^2 + \frac{2}{N^2} \sum_{k \neq l} e_k e_l (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l}) + \frac{2}{N^2} \sum_{k \neq l} 2g_k e_l (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l})$$

For the last term we write:

$$\frac{2}{N^2} \sum_{kl} 2g_k e_l (c_N^{n_k} c_N^{n_l} + s_N^{n_k} s_N^{n_l})$$

$$= \frac{2}{N^2} \sum_{k=0}^{N-1} \left( 2e_l c_N^{n_l} \sum_{k=0}^{N-1} g_k c_N^{n_k} + 2e_l s_N^{n_l} \sum_{k=0}^{N-1} g_k s_N^{n_k} \right)$$

$$= \frac{2}{N^2} \left( \frac{N}{2} a_n \sum_k e_k c_N^{n_k} + \frac{N}{2} b_n \sum_k e_k s_N^{n_k} \right)$$

$$= \frac{2}{N} \left( a_n \sum_k e_k c_N^{n_k} + b_n \sum_k e_k s_N^{n_k} \right)$$
So

\[ \tilde{P}(f_n) = \frac{2}{N^2} \left( \left( \sum_k g_k c_{nk} \right)^2 + \sum_k g_k s_{nk} \right)^2 \] 

\[ + \frac{2}{N^2} \sum_k e_k^2 \]

\[ + \frac{2}{N} \left( a_n \sum_k e_k c_{Nk} + b_n \sum_k e_k s_{Nk} \right) \]

\[ + \frac{2}{N} \left( a_n \sum_k e_k c_{Nk} + b_n \sum_k e_k s_{Nk} \right) \] 

\[ \frac{1}{2} \left( a_n^2 + b_n^2 \right) = P(f_n) \]

\[ E(f_n) \]

A.3 Derivation of the standard deviation of error 3

The variance of a stochastic quantity \( x \) is defined as the expectation value of the square of difference of \( x \) and the mean of \( x \):

\[ \text{var}(x) = \left\langle (x - \bar{x})^2 \right\rangle \]

For the variance of error 3 we can write

\[ \text{var}(\text{error 3}) \]

\[ = \text{var} \left( \frac{2}{N} \left( a_n \sum_k e_k c_{Nk} + b_n \sum_k e_k s_{Nk} \right) \right) \]

\[ \approx \frac{4}{N^2} \left( \text{var}(a_n \sum_k e_k c_{Nk}) + \text{var}(b_n \sum_k e_k s_{Nk}) \right) \]

\[ = \frac{4}{N^2} \left( a_n^2 \sum_k \text{var}(e_k) \text{var}(c_{Nk}) + b_n^2 \sum_k \text{var}(e_k) \text{var}(s_{Nk}) \right) \]

In the first step the variance of a sum is written as the sum of two variances. Since the quantities \( \sum_k e_k c_{Nk} \) and \( \sum_k e_k s_{Nk} \) are not independent this splitting is analytically incorrect. It is assumed, however, that the numerical effect of this approximation is small. We apply the definition of the variance, inserting zero for the means of the numbers \( e_k, c_{Nk} \) and \( s_{Nk} \):

\[ \text{var}(\text{error 3}) \]

\[ = \frac{4}{N^2} \left( a_n^2 \sum_{k=0}^{N-1} \left\langle e_k^2 \right\rangle \left\langle c_{Nk}^2 \right\rangle + b_n^2 \sum_{k=0}^{N-1} \left\langle e_k^2 \right\rangle \left\langle s_{Nk}^2 \right\rangle \right) \]

\[ = \frac{4}{N^2} \left( a_n^2 N \text{P}(e) + b_n^2 N \text{P}(e) \right) \]

\[ = \frac{4}{N^2} \left( a_n^2 + b_n^2 \right) \text{P}(e) \]

\[ = \frac{4}{N} \text{P}(f_n) \text{P}(e) \]

The standard deviation of error 3 is thus

\[ \text{sd}_{\text{error 3}} = 2 \sqrt{\text{P}(f_n) \text{P}(e) / \sqrt{N}} \]

Since the expectation value of error 3 is assumed to be zero, this standard deviation is the equal to the root mean square value (RMS) of error 3.
A.3.1 Comparison of the RMS values of errors 1 and 3

Dividing the RMS of error 3 by the RMS of error 1 yields

\[
\frac{\text{RMS}_{\text{error } 3}}{\text{RMS}_{\text{error } 1}} = \frac{2\sqrt{P(f_n)P(e)/\sqrt{N}}}{2/N P(e)} = \sqrt{P(f_n)/P(e)\sqrt{N}}
\]
I would like to thank the following people:

Jean-Charles Cigal, André Duif, Klaas Kopinga, Loek Baede, Gerrit Kroesen, Hans Freriks, Marcel Hemerik, Carole Maurice, Gabriela Paeva, Fred van Nijmweegen, Gerard Harkema.

A random generator has been used to determine the order of these names. However, the first two times I did not like the produced order, and I ran the random generator again.