University of Warsaw
Faculty of Physics

Adam Świniarski
Record book number: 427915

Transfer interferometer for laser frequency stabilization

Bachelor's thesis
in the field of Physics

The thesis was written under the supervision of
Dr. Mariusz Semczuk
Division of Optics, Institute of Experimental Physics

Warsaw, September 2023

Summary

The thesis describes the theory and construction of a laser frequency stabilization system employing a transfer interferometer to maintain the stability of slave lasers relative to a reference laser. This approach enables precise frequency stabilization of multiple lasers without relying on atomic or molecular transitions as a reference. Furthermore, its implementation is straightforward, utilizing readily available components.

The system is based on a Mach-Zehnder interferometer with a mirror mounted on a piezoelectric transducer (PZT) which is driven by a ramp signal generated by a digital-to-analog converter controlled by Raspberry Pi 4B. By altering the optical path length difference, interferometric signals are produced, which are subsequently digitized using an analog-to-digital converter. The extracted phases of these signals can then be compared to the phase of the reference laser signal. Based on this comparison, the microcontroller computes the necessary frequency corrections for the slave lasers, yielding an error signal that can be applied as a feedback to the controllers of these lasers.

To test the constructed setup, an 852 nm laser was used as a slave laser and a 767 nm laser, frequency stabilized to an atomic transition in potassium, served as a reference. The interferometric signal generated in the setup had variable periodicity, most likely caused by a slightly non-linear behaviour of the PZT. Introducing an empirical functional form of the response of the PZT to the applied voltage made it possible to extract the phase of the signal from the interferometric data - at a cost of increased computational complexity.

Keywords

laser, frequency, stabilization, transfer interferometer

Title of the thesis in Polish language

Interferometr transferowy do stabilizacji częstotliwości laserów

Contents

Chapter 1

Introduction

Lasers have found extensive applications in modern science and technology. They are indispensable in various fields, including telecommunications, analytical spectroscopy, pollution monitoring, medicine, manufacturing or research. However, to fully harness their capabilities, ensuring the long term stability of their frequency is crucial. Numerous factors like temperature, humidity or pressure variations, as well as mechanical vibrations can cause frequency fluctuations and frequency drifts of the emitted light. Therefore it is important to monitor the frequency and compensate its changes.

1.1. Stabilizing frequency to atomic or molecular transitions

One of the most widely adopted techniques makes use of characteristic atomic [1] or molecular [2] transitions that serve as references to which laser frequencies can be compared. They offer exceptional stability and exhibit narrow linewidths, allowing for simultaneous locking of multiple lasers to distinct transitions. To compare a laser frequency with the reference, the laser light is passed through a cell containing a desirable gas that absorbs photons with a frequency equal to that of the transition. The transmitted light is measured by a photodiode, and the closer the laser frequency is to the atomic resonance, the lower the measured intensity is. However, because of the thermal motion of atoms, the Doppler effect is present, so the resonance frequency is dependent on velocities of atoms and thus the absorption occurs over a broader range of frequencies, which lowers the lock precision. To address the problem, the method is modified by splitting the beam into a probe beam directly passing the cell and a stronger pump beam that travels in the opposite direction through the cell so it interacts with atoms moving at opposite velocities with respect to those interacting with the first beam. When the laser is nearly on resonance, both beams resonate with the same group of atoms and the absorption of the probe beam is reduced due to the population of atoms in excited state. As a result, absorption dips are observed at atomic transitions in the transmission spectrum. The described technique is called Doppler-free spectroscopy [3].

Unfortunately, atomic or molecular spectra consist of a finite selection of suitable transitions and thus available frequency references. To overcome this limitation, adjustments can be made by modifying the laser frequency using an appropriate frequency modulator, effectively shifting it into alignment with the desired reference. Another solution may be combining of the beam from a master laser that has already been stabilized to an atomic or molecular reference with that of another laser and stabilizing the resultant beat frequency. In both cases the shift is typically up to several tens of GHz which might not be sufficient in certain

situations.

1.2. Fabry–Pérot cavity

An alternative technique is called the Pound-Drever-Hall (PDH) method [4, 5]. A schematic of a typical experimental setup is shown in Fig. 1.1. The method utilizes FM (frequency modulation) spectroscopy [6]. The laser beam's phase is modulated using an electro-optic modulator driven by a sinusoidal signal from an RF oscillator. It introduces sidebands around the laser central frequency (a carrier component). Then it is directed into an optical cavity, in this case a Fabry–Pérot interferometer (FPI) [7] that consists of two highly reflecting mirrors. When light enters the cavity, it undergoes multiple reflections between the two mirrors. The cavity is set up in such a way that only specific wavelengths of light satisfy the resonance condition. The constructive interference occurs when the round-trip distance ($2L$) between the mirrors is an integer multiple of the wavelength λ of the light: $2L = m\lambda$. Some part of the light leaves the cavity and together with the light reflected from the cavity is measured by a photodetector. The signal consists of the two unaltered side bands and a carrier component shifted in phase. Then it is mixed with the RF oscillator synchronized with the light modulation phase and appropriately filtered to serve as an error signal that is fed to a PID controller which adjusts the laser frequency.

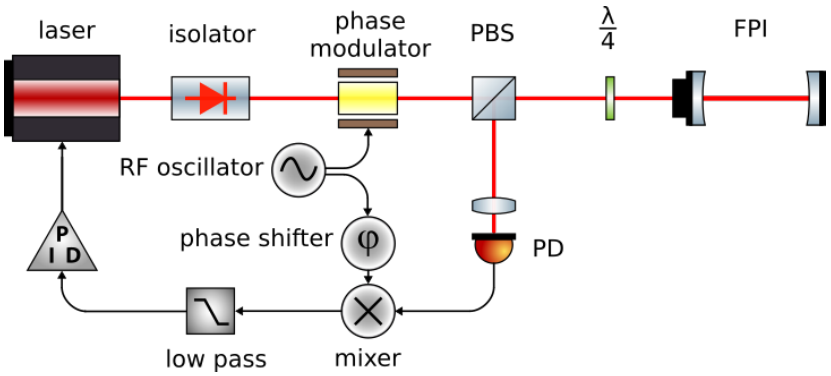


Figure 1.1: The Pound-Drever-Hall method schematic (by Kondephy - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53790278)

The method is very sensitive to small frequency shifts, thus it is highly precise. In contrast to locking to atomic resonances, it offers more flexibility as it allows for the stabilization to arbitrary frequencies within the tuning range of the laser and the resonance characteristics of the optical cavity. However, the setup is relatively complex and greatly sensitive to environmental factors like temperature or pressure variations and mechanical vibrations. There are some solutions that can eliminate the problems, for example the Fabry–Pérot cavity can be made from an ultra-low expansion (ULE) glass (e.g. Zerodur) that is known for its exceptionally low coefficient of thermal expansion in the vicinity of a characteristic temperature [8]. Also, the setup can be closed in a box to isolate it from the environment.

There exists another approach [9] utilizing a Fabry–Pérot cavity. It transfers the stability of a master laser to multiple slave lasers. The reference laser can be stabilized to one of the frequencies of the FPI as it has been discussed above. Then we can use another cavity whose one mirror is attached to a piezo-electric transducer (PZT), which can scan the length between the mirrors and thus change the characteristic frequencies of the setup. It can be

locked to the frequency of the reference laser that is sent to the cavity. It is done by observing the transmitted signal and moving the mirror to a position, for which the laser frequency corresponds to one of the resonance frequencies of the cavity, which manifests in the maximum transmission. Then a few other lasers can be simultaneously stabilized to a given frequency of such a regulated cavity. It eliminates the problem with the drifts of the cavity length caused by the environment.

1.3. Mach-Zehnder interferometer

There is yet another, a bit easier, possibility that is deeply discussed in the thesis. In the paper from the University of Toronto [10] a transfer interferometer was utilized to convey the frequency stability of an already well stabilized (e.g. using a proper atomic resonance reference) reference laser (also called a master laser) to multiple slave lasers. The setup schematic from the article is presented in figure 1.2. They used a Mach-Zehnder interferometer built from two 50:50 beam splitter cubes (BS) and a single mirror attached to a piezoelectric transducer (PZT) that is driven by a ramp signal generated by a digital-to-analog converter (DAC) controlled by an Arduino-compatible microcontroller (Digilent uC32). The displacement of the mirror introduces a phase shift in the longer path and thus it results in interferometric signals measured by two photodetectors after separating the beams using a dichroic mirror.

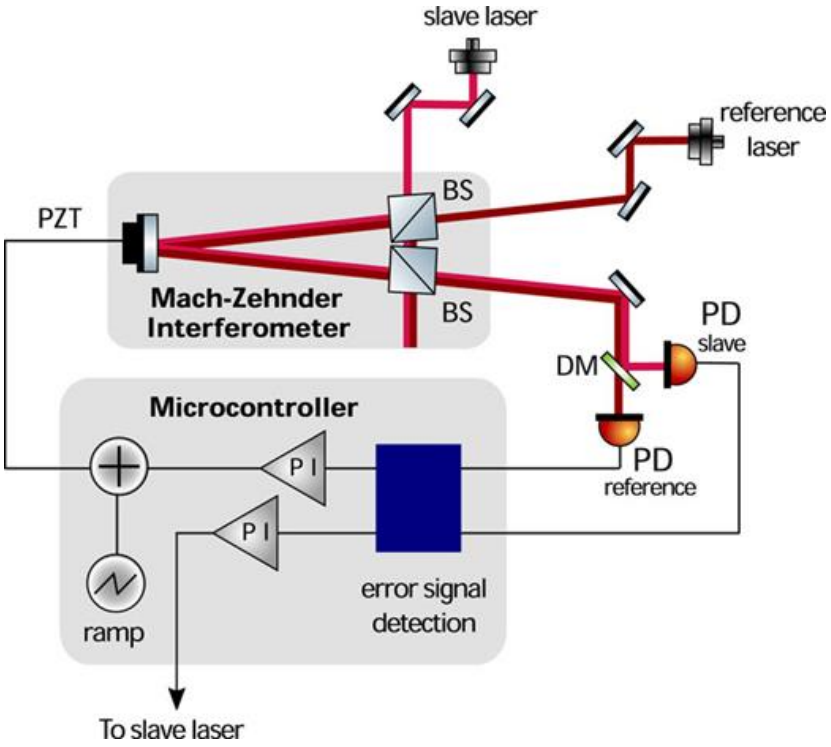


Figure 1.2: A transfer interferometer setup for slave laser frequency stabilization using a stabilized master laser as a reference. BS – beam splitter cube, PZT – piezoelectric transducer, DM – dichroic mirror, PD – photodetector, PI –proportional-integral controller (source: [10])

Then the signals are digitalized by an analog-to-digital converter (ADC), sent to the microcontroller and analysed by fitting sine functions to them and extracting their phases. It turns

out that the difference of these phases is proportional to the frequency deviation, so it may serve as an error signal that is subsequently fed into a proportional-integral (PI) controller implemented in the software and sent to slave lasers' controllers to correct the frequency by manipulating the current. Also, the reference laser phase is stabilized to maintain a constant interferometer length and therefore compensate effects of thermal drifts. It is done by comparing the current phase with the set one and adding some offset to the ramp signal to regulate the average position of the mirror at the PZT.

This approach is relatively simple and cheap in contrast to the other methods presented before primarily because it does not require expensive, high-reflectivity mirrors, only components available in every optical laboratory. Moreover, it was reported to allow for frequency stabilization better than 1 MHz [10].

1.4. Motivation

The Quantum Gases Laboratory, led by Dr. Mariusz Semczuk, is currently constructing an experiment aimed at producing a mixture of silver with potassium and cesium. It can possibly allow to generate ultracold molecules with an enormous dipole moment [11]. However, to create a magneto-optical trap for silver, light with a wavelength of 328 nm is required [12]. Unfortunately, currently there are no lasers with such a wavelength that have spectrum narrow enough, are stable, tunable, and of sufficient power. Therefore, a proposed solution is to generate this specific wavelength through a process known as second harmonic generation (SHG) that effectively halves the wavelength of 656 nm. To obtain the 656 nm wavelength, a sum frequency generation (SFG) process is considered, combining the wavelengths of lasers at 1114 nm and 1596 nm. The schematic of the idea is shown in Fig. 1.3.

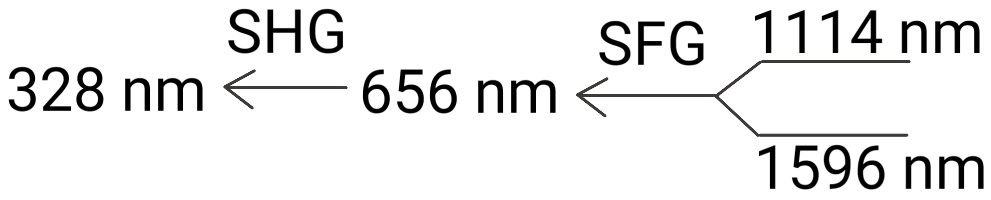


Figure 1.3: In order to get laser light of wavelength 328 nm, two wavelengths, 1114 nm and 1596 nm, are combined to produce 656 nm light, which is then doubled in frequency, resulting in a 328 nm wavelength. SHG – second harmonic generation, SFG – sum frequency generation.

However, the frequency stabilization of these lasers poses a challenge. Traditional methods, such as relying on atomic or molecular transitions, are not applicable in this context, as in the vicinity of these wavelengths there are no well-established atomic or molecular frequency references. Therefore, alternative approaches are being considered. One of them is to stabilize one of these lasers to the frequency comb and then employ a Mach-Zehnder interferometer to transfer the frequency stability from that laser to the second one. We can also use the light of 656 nm wavelength as an input to the M-Z interferometer and send the error signal to one of these lasers while the other one is stabilized to the frequency comb. Last but not least, the spectroscopy of silver atoms may be performed using the 328 nm light.

Chapter 2

Theoretical background

In contrast to the interferometer presented in the research paper [10] that I rely on, previously shown in Fig. 1.2, ordinary non-polarizing beam splitter cubes (BS) were replaced with polarizing beam splitter cubes (PBS). BS transmits only a fraction of incident light power, regardless of polarization, and the rest is reflected. In the case of the paper that I mentioned the BS transmitted and reflected 50% respectively. However, PBS transmits only the p-polarization (from the German parallel – parallel to the plane of incidence) and reflects the s-polarization (from the German senkrecht – perpendicular to the plane of incidence), as shown in Fig. 2.1, so not only it works as a beam splitter but also as a linear polarizer.

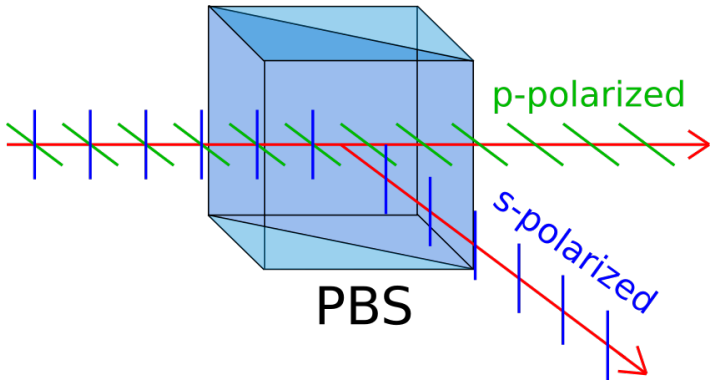


Figure 2.1: The principle of operation of a polarizing beam splitter. In an ideal case the p-polarized component (green lines) is fully transmitted and the s-polarized one (blue lines) is fully reflected.

In order to achieve the 50:50 beam splitting functionality in a PBS, it is necessary to use a linearly polarized light and a half-wave plate (HWP) set at an appriopriate angle and inserted just before the PBS. A half-wave plate is made of a birefringent material which has different refractive indices for light polarized along two orthogonal axes called "fast" and "slow". When linearly polarized light enters a half-wave plate, the two orthogonal polarization components, one aligned with the fast axis and the other with the slow axis, experience different refractive indices. As a result, they travel at different speeds through the material. In the case of a half-wave plate it introduces a phase shift of half a wavelength or π rad between the two polarization components, as shown in Fig. 2.2. As it can be seen, if the incident beam is linearly polarized at an angle α with respect to the slow axis or β to the fast axis, it effectively rotates the polarization by 2α clockwise or 2β anticlockwise. To visualise it, the linear polarization with its characteristic components were drawn in two points of maximum

amplitudes just after light passes the HWP.

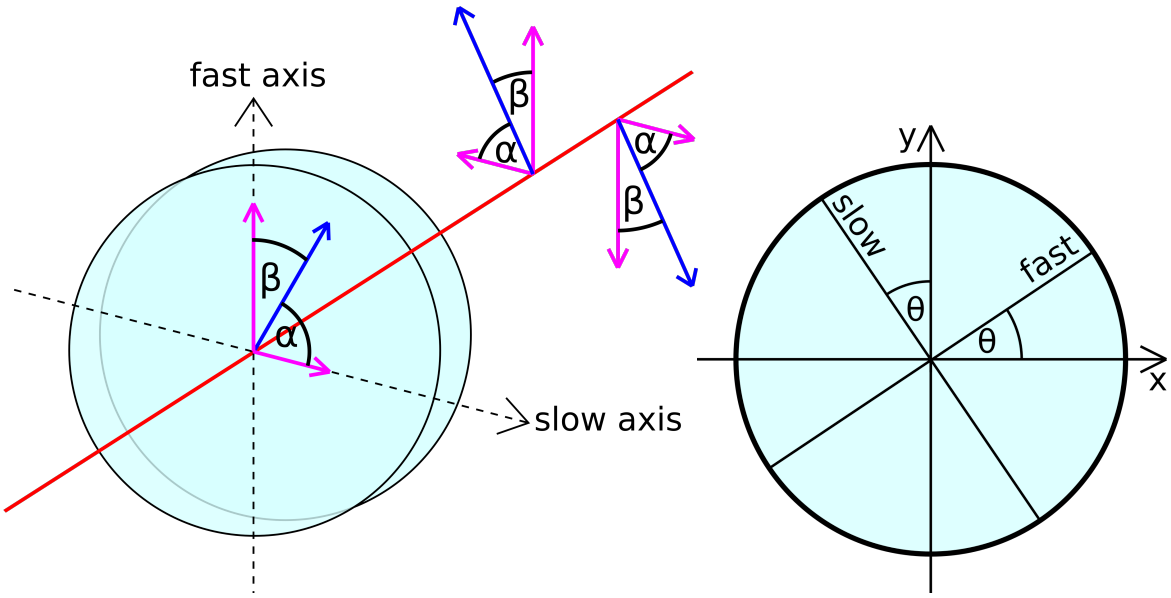


Figure 2.2: A half-wave plate introduces a phase difference of π rad between polarization components along orthogonal "fast" and "slow" axes. The blue arrows symbolize linear polarization of light that passes the half-wave plate. The magenta arrows are the "fast" and "slow" components of the linear polarization. On the right, it has been placed in a coordinate system where x is along p-polarization (horizontal), and y along s-polarization (vertical).

Mathematically, the operation of a half-wave plate can be described using Jones calculus [13]. Let $\begin{bmatrix}1\\0\end{bmatrix}$ represent the p-polarization and $\begin{bmatrix}0\\1\end{bmatrix}$ the s-polarization. Then assuming the HWP is oriented at an angle of θ with respect to the horizontal direction denoted as x, such as in the right image in Fig. 2.2, we can write down the following Jones matrix:

$$J_{HWP}(\theta) = \begin{bmatrix}\cos(2\theta) & \sin(2\theta)\\ \sin(2\theta) & -\cos(2\theta)\end{bmatrix} \tag{2.1}$$

To obtain a 50:50 PBS operation and if incident light is p-polarized, one has to rotate the half-wave plate, such that the fast or slow axis is oriented at 22.5° angle with respect to the polarization direction. Then p and s components are equal after passing the HWP and the PBS transmits and reflects beams of equal powers:

$$J_{HWP}(22.5^{\circ})\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \tag{2.2}$$

Having that in mind let's analyse a simplified schematic of a triangular Mach-Zehnder interferometer shown in Fig. 2.3. A laser beam typically is already linearly polarized. To regulate its power one can rotate the HWP placed before the PBS. As it was discussed above, it rotates the polarization changing s and p components' intensities. Moreover, it ensures the p-polarization of transmitted light which then can be rotated by another HWP oriented such

that the next PBS transmits and reflects equal-power beams and thus behaves like a 50:50 BS.

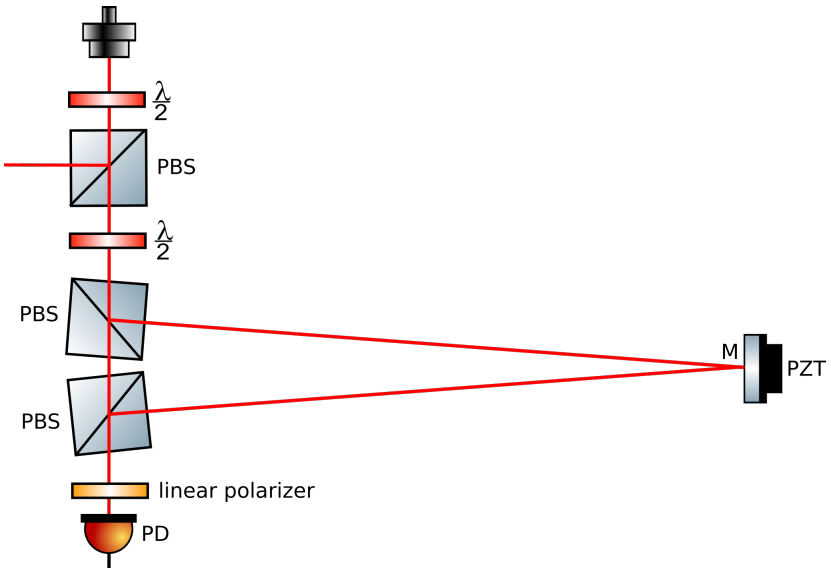


Figure 2.3: A simplified schematic of a triangular Mach-Zehnder interferometer. PBS – polarizing beam splitter cube, M – mirror, PZT – piezo-electric transducer, PD – photodiode. Half-wave plates are denoted as $\frac{\lambda}{2}$.

There is yet another PBS. They are both slightly rotated. Light traverses two distinct paths. The short one goes straight through both PBS cubes. The second one reflects twice at the PBS cubes and once at the mirror mounted on a piezoelectric actuator (PZT) that is driven by a sawtooth signal and dynamically alters the path length over time. At the output the beams are shifted in phase by a constant $\Delta\phi_0$ resulting from a fixed difference in the optical path length ΔL_o between the two paths, phase changes caused by reflections at the mirror and PBS cubes, and a varying mirror displacement $\Delta x(V)$ dependent on the voltage supplied to the PZT. Let's assume $\Delta x(V)$ is much smaller than the path length, so that the overall path change can be approximated by $2\Delta x(V)$. In the result the relative phase shift is:

$$\Delta\phi = \Delta\phi_o - 2k\Delta x(V), \tag{2.3}$$

where k – wave number.

The beams meet again at the output of the second PBS that transmits all the light traversing the short path as it is only p-polarized, and reflects light traveling the long path, which is s-polarized. At the output the beams are orthogonally polarized and can't interfere with each other. That's why a linear polarizer is used to ensure the same polarization states. Maximum intensity can be obtained when it is oriented at 45° or right between p and s polarizations. Alternatively, PBS with HWP before it can be used, because PBS ensures p-polarized transmitted light and HWP together with the PBS can regulate the output power.

Let's assume the half-wave plate at the input is oriented such that the p and s polarization components are the same. We can describe it by a Jones vector [13]: $\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$. For a half-wave plate with fast axis at angle $\theta = 22.5°$ with respect to the horizontal axis (Fig. 2.2) the following Jones matrix describes its influence on the polarization:

$$J_{HWP}(22.5^{\circ}) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.4}$$

Since only the p-polarized component is transmitted by a PBS, at the output, before the linear polarizer, a Jones vector for the shorter path is: $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. For the longer path it is: $\frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ e^{i\Delta\phi} \end{bmatrix}$ as the s-polarization is reflected only and there is the relative phase difference.

For a linear polarizer with transmission axis at an angle θ to the horizontal axis the following Jones matrix is given:

$$J_{LP}(\theta) = \begin{bmatrix} \cos^2\theta & \cos\theta\sin\theta \\ \cos\theta\sin\theta & \sin^2\theta \end{bmatrix} \tag{2.5}$$

When $\theta = 45^{\circ}$, then it is:

$$J_{LP}(45^{\circ}) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{2.6}$$

The overall result of the phase change and passing through the linear polarizer angled at 45° can be calculated as follows:

$$J_{out}(\Delta\phi) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ e^{i\Delta\phi} \end{bmatrix} = \frac{e^{i\frac{\Delta\phi}{2}}}{\sqrt{2}} \begin{bmatrix} \cos\frac{\Delta\phi}{2} \\ \cos\frac{\Delta\phi}{2} \end{bmatrix} \tag{2.7}$$

The intensity is proportional to the Jones vector module squared, so the output signal is:

$$I_{out} = I_{in}\cos^2(\frac{\Delta\phi}{2}) \tag{2.8}$$

For the future's sake, let's express the intensity 2.8 with a sine function using well-known trigonometric identities:

$$I_{out} = \frac{1}{2}I_{in} + \frac{1}{2}I_{in}\sin\left(2k\Delta x(V) - \Delta\phi_0\right) \tag{2.9}$$

In reality, polarizing beam splitter cubes aren't perfect and they transmit a bit of the s-polarization or reflect some p-polarization. It may be so the intensity doesn't completely drop to zero and has some small offset.

In general:

$$I(\Delta x) = A\sin(2k\Delta x + \varphi) + B \tag{2.10}$$

For low voltages, the PZT displacement can be approximated by a linear relationship. Unfortunately, in order to observe full interference fringes, the signal amplitude should be of the order of 100 V and that is not valid anymore. Based on the data, as it will be shown later, the following exponential formula turns out to be a good approximation:

$$\Delta x(V) = a \cdot \left(e^{bV} - 1\right) \tag{2.11}$$

It is evident that for small voltage, it exhibits a linear response as expected.

A ramp signal ranging from 0 V to V_{max} and with frequency $f = \frac{1}{T}$ can be described by:

$$V(t) = \frac{2V_{max}}{T}t \quad \text{for t} \in (0, \frac{1}{2}\text{T}) \tag{2.12}$$

If one inserts it into 2.11, it results in:

$$\Delta x(t) = \Delta x_{max}\left(e^{ct} - 1\right), \tag{2.13}$$

where $c = 2V_{max} \cdot f$.

Finally, the signal observed on the photodiode can be written as:

$$I(t) = A\text{sin}\left(we^{dt} + \phi\right) + B \tag{2.14}$$

To assess the quality of interference signals a parameter called the fringe visibility [14] can be calculated. It is defined by:

$$\text{Vis} = \frac{I_{\text{max}} - I_{min}}{I_{max} + I_{min}} \tag{2.15}$$

Chapter 3

Extracting phases and calculating the error signal

In order to stabilize lasers using the presented interferometer, one has to find a proper error signal that is proportional to the frequency deviation. It turns out it can be obtained from phases of the master and slave lasers. However, they have to be somehow extracted from interference signals. Assuming the PZT is perfect or at least approximately linear in a given range of voltages, the signal is described by a sinusoid. An effective algorithm based on the least squares method is proposed. Once the phases are known, the error based on these phases can be computed and sent to a feedback loop to compensate slave laser frequency.

3.1. Error signal

Let's consider first-order fluctuations of the phases of the reference (R) and slave (S) lasers' beams caused by an infinitesimally small interferometer length change δL. If we assume that the reference laser frequency is well-known ($\delta k_R = 0$), then they are given by the following equations:

$$\delta\phi_R = k_R \delta L \tag{3.1}$$

$$\delta\phi_S = L\delta k_S + k_S \delta L \tag{3.2}$$

If we divide them by k_R and k_S respectively and subtract the first one from the second one, we get:

$$e = \frac{\delta\phi_S}{k_S} - \frac{\delta\phi_R}{k_R} = L \cdot \frac{\delta f_S}{f_S} \propto \delta f_S \tag{3.3}$$

The result is directly proportional to the slave laser frequency fluctuation and independent of the changes of the path length difference or laser power variations. The calculated value is a perfect candidate for the error signal. To minimize the frequency deviation one has to minimize the error signal.

3.2. Phase extracting algorithm

Everytime the interferometer is scanned with the PZT, the Fast Fourier Transform (FFT) is performed on the collected interference signal data and the peak frequency is found. Once

the frequency is known, a three-parameter least square sine fitting algorithm [15] is applied.

A sine function can be decomposed into a sum of cosine, sine, and a constant terms:

$$y_n = A\sin(2\pi ft + \phi) + B = A_0\cos(2\pi ft_n) + B_0\sin(2\pi ft_n) + C_0 \tag{3.4}$$

In order to fit the sine function to the collected data y_1, y_2, ..., y_N at times t_1, t_2, ..., t_N, the square sum of the differences between the measured values and the fitted ones must be calculated and then minimized:

$$\sum_{n=1}^{N} \left(y_n - A_0\cos(2\pi ft_n) - B_0\sin(2\pi ft_n) - C_0\right)^2 \tag{3.5}$$

Now, matrix D has to be created as follows:

$$D = \begin{bmatrix} \cos(2\pi ft_1) & \sin(2\pi ft_1) & 1 \\ \cos(2\pi ft_2) & \sin(2\pi ft_2) & 1 \\ \vdots & \vdots & \vdots \\ \cos(2\pi ft_N) & \sin(2\pi ft_N) & 1 \end{bmatrix} \tag{3.6}$$

Each consecutive matrix column contains cosine and sine functions calculated at all the times and the last one has 1 in all rows. It is constructed in such a way that by multiplying the matrix by $s_0 = \begin{bmatrix} A_0, B_0, C_0 \end{bmatrix}^T$ consisting of the proper coefficients, we get an N-dimensional vector $y = \begin{bmatrix} y_1, y_2, \ldots, y_N \end{bmatrix}^T$ representing the collected data.

The solution that minimizes 3.5 may be obtained by calculating the following vector:

$$\hat{s}_0 = \begin{bmatrix} \hat{A}_0 \\ \hat{B}_0 \\ \hat{C}_0 \end{bmatrix} = \left(D^T D\right)^{-1} \left(D^T y\right) \tag{3.7}$$

Now we can return to the original expression of the fitting function $y_n = A\sin(2\pi ft_n + \varphi) + C$ and see that the phase is given by:

$$\phi = -\arctan\left(\frac{\hat{B}_0}{\hat{A}_0}\right) \tag{3.8}$$

3.3. Feedback loop

As a feedback loop a Proportional-Integral-Derivative (PID) controller may be used. It is designed to maintain a desired setpoint by continuously adjusting a control input based on the error $e(t)$ that is the difference of the setpoint and the current process variable, in this case $e(t) = \frac{\delta\phi_S(t)}{k_S} - \frac{\delta\phi_R(t)}{k_R}$. The controller consists of three components:

1. **Proportional (P)** – it is proportional to the error.

   $$P(t) = K_p \cdot e(t)$$

   where: K_p – proportional gain coefficient.

   Its drawback is that it does not ensure zero error at the steady state.

2. **Integral (I)** – it accounts for the accumulation of past errors over time. It eliminates the steady-state error entirely when configured correctly.

$$I(t) = K_i \cdot \int_0^t e(\tau)\, d\tau$$

where: K_i – integral gain coefficient,

However, it makes the system respond more slowly to changes in error and may lead to oscillations and overshoot if it accumulates too much error over time.

3. **Derivative (D)** – it accounts for the rate of change of the error. It dampens oscillations, prevents overshooting and responds quickly to changes in error.

$$D(t) = K_d \cdot \frac{de(t)}{dt}$$

where: K_d – derivative gain coefficient

Unfortunately, it is very sensitive to noise and can lead to instability of the system.

The schematic of the PID controller is shown in Fig. 3.1. In the case of the transfer interferometer the actuator is a controller of a slave laser and the process is the calculation of the differences between slave and master interference phases and that difference is the error, which serves as a feedback.

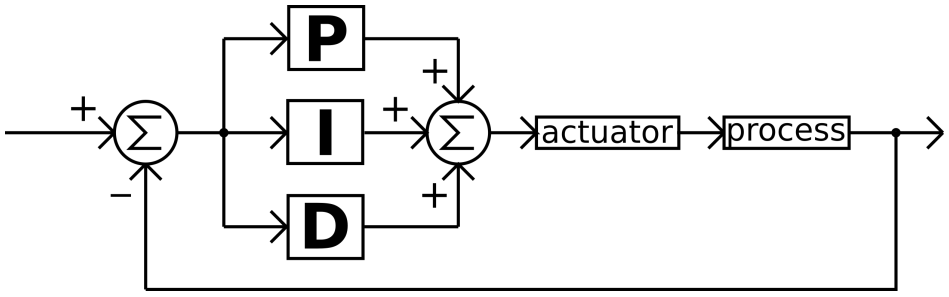


Figure 3.1: PID schematic

The overall control signal of the PID controller is the sum of the proportional, integral, and derivative terms:

$$u(t) = P(t) + I(t) + D(t)$$

To achieve the desired control performance, the PID controller gains (K_p, K_i, K_d) must be appropriately tuned. Often P and I components are sufficient and there is no need for the D component.

The PID controller may be implemented directly in the software or it can already be a part of the hardware controlling a laser. In the second case, the program only needs to generate $e(t)$, which is sent to the device and appropriate gains are chosen within it.

Chapter 4

Experimental procedure

The experimental setup consists of two parts: optical and electronic. The first one is a Mach-Zehnder interferometer for generating interference signals for master and slave lasers. The second one is used for controlling the length of the interferometer, digitizing obtained signals, performing all required computations and generating error signal, which is used as a feedback and sent to the controller of the laser that is being stabilized.

4.1. Optical setup

The experimental setup is shown in Fig. 4.1. Two lasers were used. The first one, later on called master or reference laser, is Toptica PRO 767, an external-cavity diode laser (ECDL) with diffraction grating, for which central wavelength of the diode was tuned around potassium D2 transition (767 nm) and whose frequency was stabilized via sub-Doppler saturation spectroscopy. The second one is PHOTODIGM PH852DBR TO-8 with central wavelength tuned around cesium D2 transition (852 nm). It is controlled by Koheron CTL200.

The master laser beam is collimated by two plano-convex lenses with focal lengths 2 mm (already in the collimator) and 50 mm. In order to regulate the beam power and ensure the horizontal polarization, the beam passes through a half-wave plate and a PBS before it enters the interferometer as it was discussed earlier. The slave laser beam traverses a similar path but in this case there was no need to additionally collimate the beam.

In contrast to Fig. 2.3, two additional mirrors were added to reduce the area occupied by the interferometer. The longer interferometer's arm is 1 m long while the mirrors are 29 cm away from the PBS cubes and 21 cm from the mirror at PZT. If it was set like in Fig. 2.3, the mirror at the piezoelectric transducer (PZT) should be 50 cm away from the PBS cubes in order to obtain the same arm length, so the interferometer would be nearly two times longer. The PBS cubes are 4.5 cm away from each other, so the total path length difference is 95.5 cm.

The piezoelectric actuator is driven by a sawtooth signal generated by a digital-to-analog converter controlled by Raspberry Pi 4B and then amplified by TEM-Messtechnik miniPiA 103.

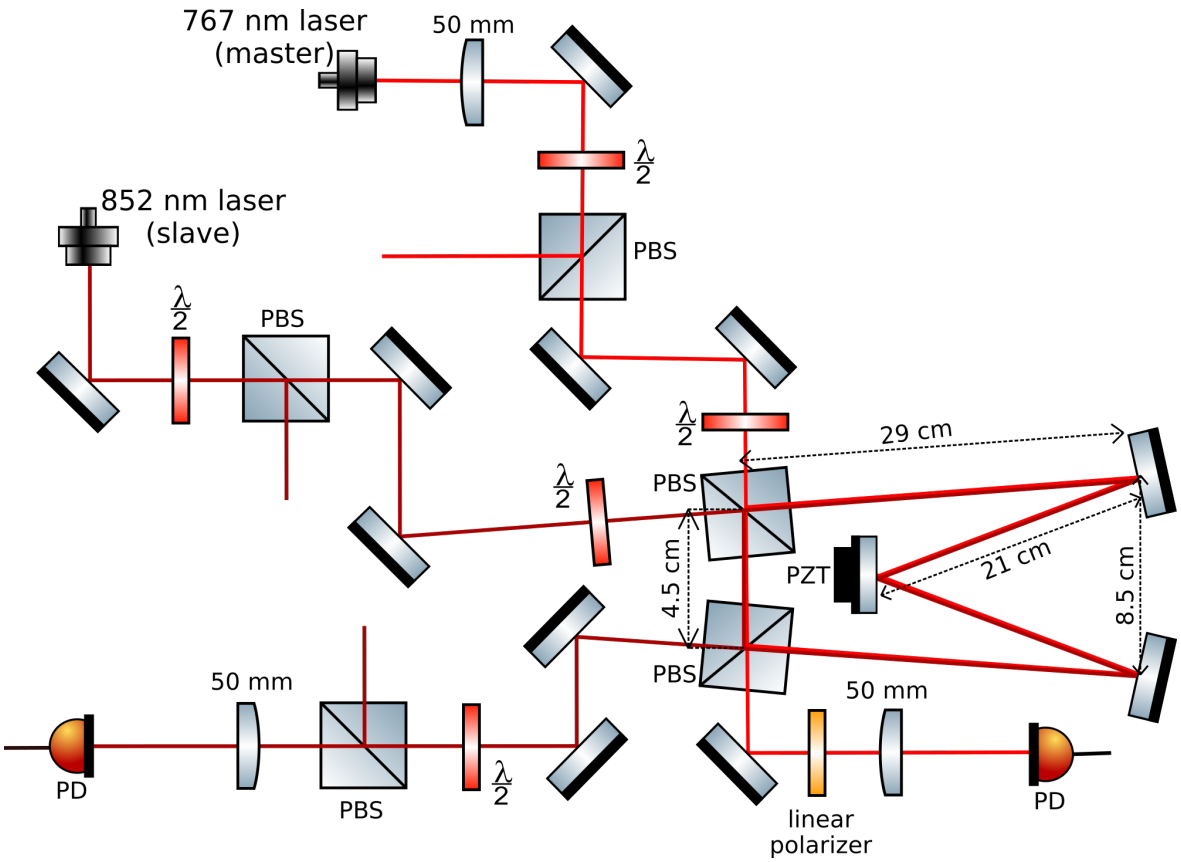


Figure 4.1: Experimental setup. PBS – polarizing beam splitter, PZT – piezo-electric transducer, PD – photodiode, $\frac{\lambda}{2}$ – a half-wave plate. The path length difference is about 1 m.

The photo of the real setup is shown in Fig. 4.2.

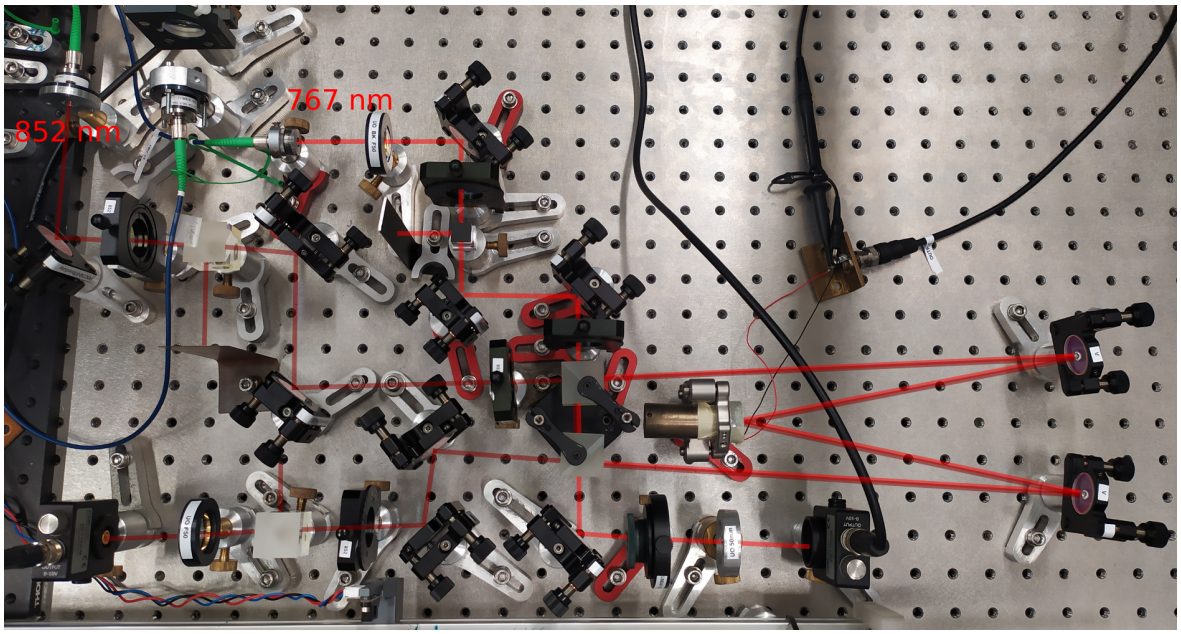


Figure 4.2: Photo of the interferometer.

At the master laser interferometer's output there is a linear polarizer rotated at 45° that ensures the same polarization of both beams. It is crucial for interference to occur. Then the beam is focused by a 50 mm plano-convex lens (Union Optics BK7) at the photodetector (Thorlabs DET10A Si Biased Detector). The detected signal is observed on RIGOL DS1054 oscilloscope and also sent to the analog-to-digital converter. In the case of the slave laser beam, PBS cube with HWP before it were used at the output to ensure the same polarization of the beams, because there were no more available linear polarizers in the lab. It is also focused by a plano-convex lens ($f = 50$ cm) at another photodetector of the same kind.

4.2. The microcontroller, analog-to-digital and digital-to-analog converters

The microcontroller, Digilent uC32, used in the original paper, has been discontinued and is no longer available in any store. Likewise with the analog shield, it is also unavailable. That's why new hardware had to be found. Basic versions of Arduino microcontrollers, like Uno or Leonardo, have insufficient RAM. Possibly Arduino Due or Mega should be sufficient but eventually Raspberry Pi 4B (RPi 4B) was chosen due to its vast computational capabilities. The comparison of these two boards is presented in Tab. 4.1.

Table 4.1: Comparison of Raspberry Pi 4B and Digilent UC32 specifications

Specs	**Raspberry Pi 4B**	**Digilent UC32**
CPU	Broadcom BCM2711	Microchip PIC32MX795F512L
CPU Clock Speed	1.8 GHz	80 MHz
Number of Cores	4	1
RAM	2GB	32 KB

In the case of the Rpi 4B CPU clock is nearly 23 times faster and it has 4 cores, each supporting one thread. Also, there is nearly 63 times more RAM.

A digital-to-analog converter (DAC) that was chosen is the Adafruit MCP4728 [16]. It has four 12-bit channels and communicates through the I^2C interface (Inter-Integrated Circuit). In the case of the analog-to-digital converter (ADC), it is the Texas Instruments ADS1256 [17]. It has eight single-ended or four differential 24-bit channels and uses SPI (Serial Peripheral Interface).

The electronic setup for reading and generating the signals is presented in Fig. 4.3.

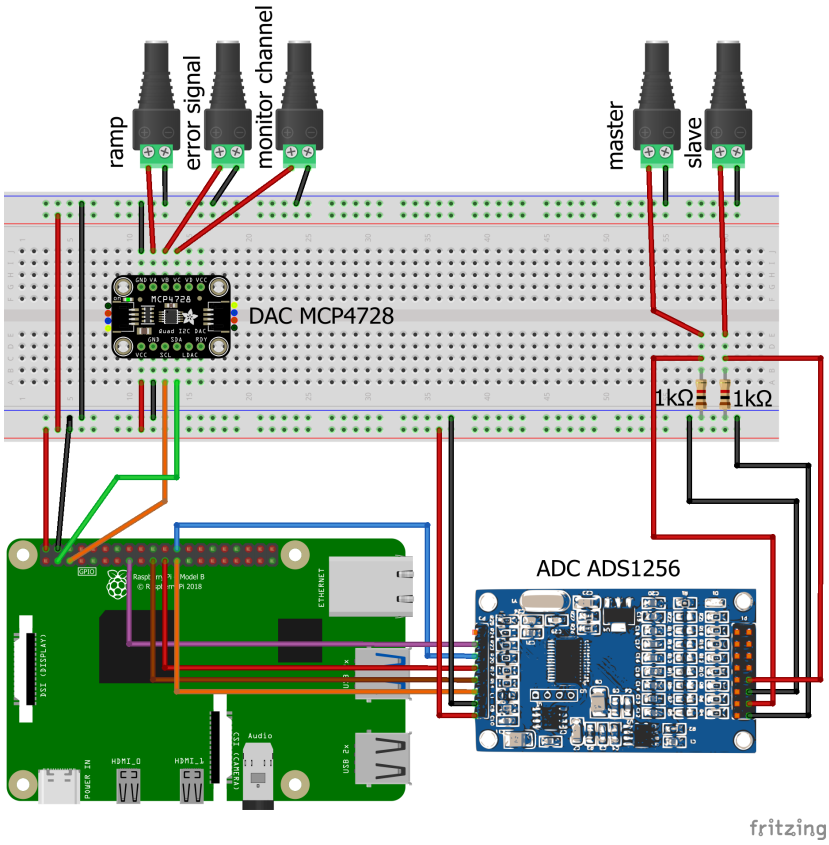


Figure 4.3: Electronic system for reading data and generating signals. Raspberry Pi 4B controlls digital-to-analog (MCP4728) and analog-to-digital (ADS1256) converters. The DAC generates ramp signal driving the PZT, error signal correcting slave laser frequency and the third channel is for monitoring fit parameters on the oscilloscope. The ADC reads interference signals of master and slave lasers and digitizes them.

The first channel of the DAC is used to generate the ramp signal driving the PZT and scanning the interferometer length. The second one provides the error signal correcting the slave laser frequency and the third one can be used to check on the oscilloscope if the function is properly fitted to the signal in real time.

The ADC channels were used in the differential mode. To eliminate periodic noises that occured, the interferometric signals were measured using 1 kΩ resistors between the common ground and positive nodes.

4.3. The code

The code is available online on GitHub [18]. The main program with all functionalities is called transfer_interferometer_v3.cpp. The logic is based on the program in Arduino syntax [19] used in the paper that I rely on.

In the first attempt the converters were programmed in Python but as they worked too

slow, C++ was used instead. Initially, WiringPi library was chosen but it was deprecated in 2019, so it has not been updated for the newest RPi 4B OS: Bullseye. While it worked well for the DAC, there was an issue with the ADC because somehow the SPI communication couldn't be established. Eventually, the BCM2835 library [20] was used instead. Additionally, for the ADC ready-made drivers [21] (BCM2835_Driver and ADS1256_Driver) were used. Both converters worked as expected. However, as it turned out later, it also is not dedicated for RPi 4B, because its CPU architecture is BCM2711 and the library was created for BCM2835 archiecture present in older versions of RPi that had only 1 CPU core and thus one thread. For a long time the author of this thesis was unaware of that fact because, upon checking the CPU specifications in the terminal, it incorrectly showed up that the architecture was BCM2835, while on the CPU chip on the board it is written "BCM2711".

The difference compared to the approach from the referenced work 1.2 is that all the operations are performed within a single code in C++ , whereas in the paper the microcontroller sent prepared sine and cosine tables through serial communication to a computer, where the D matrix was calculated in Python and then sent back to the microcontroller.

In the main infinite loop, there are two separate iterations, one for generating the rising edge of the sawtooth signal and reading the interferometric signals, and the other one for generating the falling edge. After the ramp up is done and the data has been collected, the Fast Fourier Transform (FFT) algorithm is used to find the peak frequencies of the signals. It is done by the function shown below:

```
1   double get_fit_freq(int array[], int arraySize, int SKIP_LEFT, int
        STEPS_USE) {
2
3       int dataSize = STEPS_USE;
4
5       // FFTW setup
6       fftw_complex* in = (fftw_complex*)fftw_malloc(sizeof(fftw_complex) *
            dataSize);
7       fftw_complex* out = (fftw_complex*)fftw_malloc(sizeof(fftw_complex) *
             dataSize);
8       fftw_plan plan = fftw_plan_dft_1d(dataSize, in, out, FFTW_FORWARD,
            FFTW_ESTIMATE);
9
10      for (int i = 0; i < dataSize; ++i) {
11          in[i][0] = array[SKIP_LEFT + i];
12          in[i][1] = 0.0;
13      }
14
15      // Execute FFT
16      fftw_execute(plan);
17
18      double SampleRate = 1.0;
19
20      // Frequency bins
21      double* fft_freqs = new double[dataSize];
22      for (int i = 0; i < dataSize; ++i) {
23          fft_freqs[i] = i * SampleRate / dataSize;
24      }
25
26      // The index of the maximum magnitude in the spectrum
27      int max_magnitude_index = 0;
```

```
28          for (int i = 1; i < dataSize/2; ++i) {
29              if (abs(out[i][0]) > abs(out[max_magnitude_index][0])) {
30                  max_magnitude_index = i;
31              }
32          }
33
34          double peak_frequency = fft_freqs[max_magnitude_index];
35
36          // Clean up FFTW resources
37          fftw_destroy_plan(plan);
38          fftw_free(in);
39          fftw_free(out);
40          delete[] fft_freqs;
41
42          return peak_frequency;
43      }
```

Listing 4.1: Function for finding fitting frequencies using the Fast Fourier Transform

In the code [19] that I relied on, the frequencies are not calculated by the program, instead they are guessed at the very beginning of the code, before the main loop. Then they can be again set by trial and error during the operation of the program. Here, a different approach has been tried.

Once the frequency is known, the D matrix 3.6 is calculated. The following function is responsible for that:

```
1       vector<float> get_fitting_matrix(int n_steps, float fit_freq, int
            skip_left = 10, int skip_right = 10) {
2           // steps to use
3           int n_use = n_steps - skip_left - skip_right;
4
5           MatrixXf D(n_use, 3);
6           for (int i = skip_left; i < n_steps-skip_right; ++i) {
7               float time = static_cast<float>(i);
8               float theta = PI2 * time / n_steps * fit_freq;
9               D(i - skip_left, 0) = sin(theta);
10              D(i - skip_left, 1) = cos(theta);
11              D(i - skip_left, 2) = 1.0f;
12          }
13
14          MatrixXf DTD = D.transpose() * D;
15          MatrixXf compute_matrix = (DTD.inverse() * D.transpose()).block(0, 0,
                2, n_use);
16
17          vector<float> flattened_array(2 * n_steps, 0.0f);
18
19          for (int i = skip_left; i < n_steps - skip_right; ++i) {
20              flattened_array[i]   = compute_matrix(0, i-skip_left);
21              flattened_array[n_steps + i] = compute_matrix(1, i-skip_left);
22          }
23
24          return flattened_array;
25      }
```

Listing 4.2: Function for calculating the D matrix

Chapter 5

Measurements and results

A typical view of the interference fringes is shown in Fig. 5.1. In this case the ramp signal (yellow line) is generated by RIGOL DG4162 and then amplified by TEM-Messtechnik miniPiA 103 (navy blue line). Light blue signal is the interference of the 767 nm laser and the purple one originates from the 852 nm laser interference.

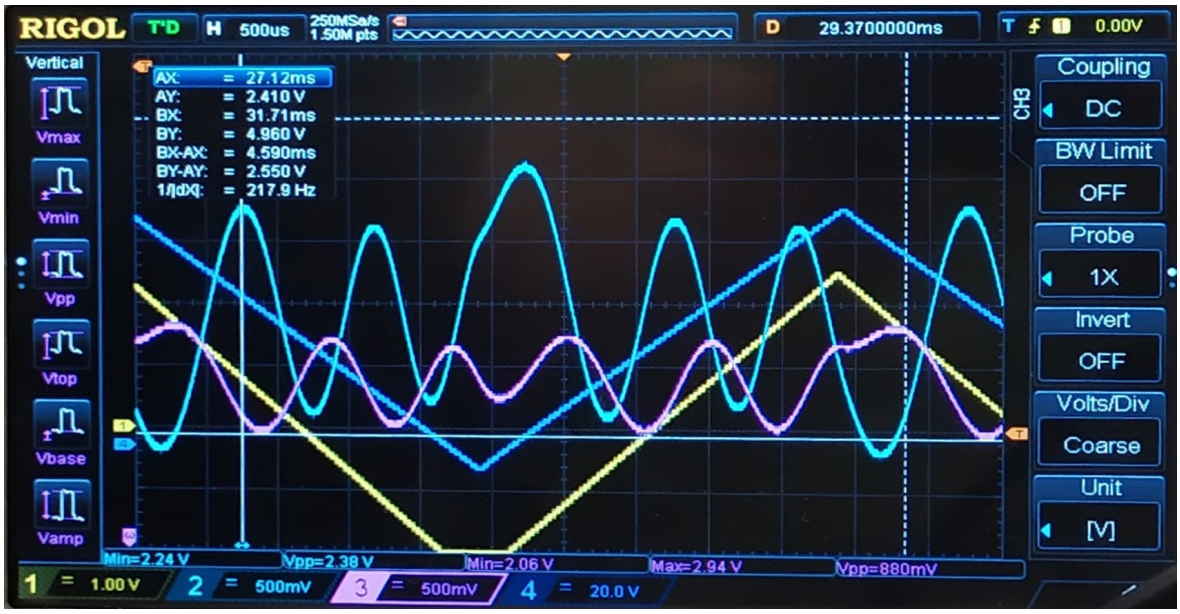
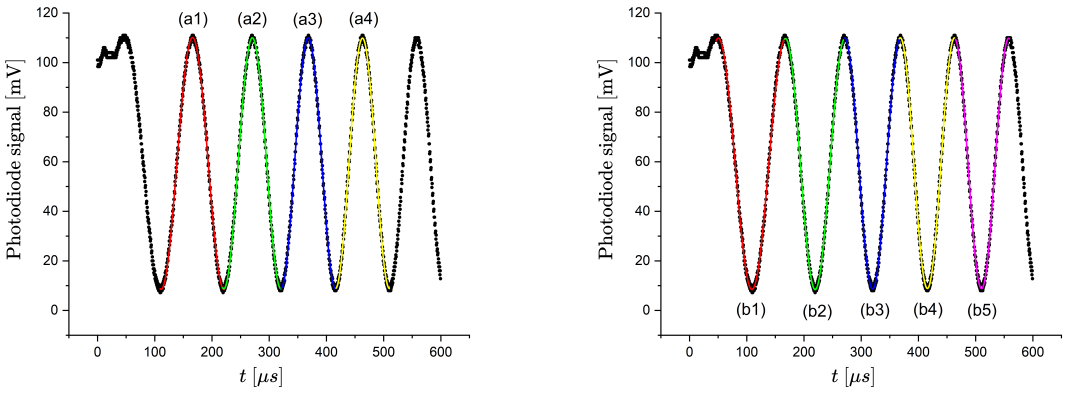


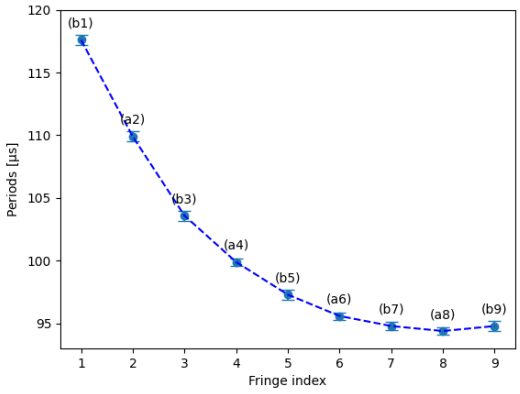
Figure 5.1: Interference signals seen on the oscilloscope. Yellow – ramp signal before the amplification, navy blue – amplified ramp signal, light blue – interference fringes for the potassium laser, purple – interference fringes for the cesium laser.

Fig. 5.2a and 5.2b show an interference signal when the PZT was driven by a 800 Hz ramp signal ranging from 0 to 10 V before the amplification. It can be seen that the fringes are not equally wide. Their width seems to decrease as the ramp voltage increases. To evaluate the change, to each individual fringe a sine function was fitted. In Fig. 5.2a the sines were fitted to four separate peaks of the signal (denoted as (a1) – (a4)) and in Fig. 5.2b to five valleys ((b1) – (b5)). The consecutive periods were plotted in Fig. 5.2c. They clearly decrease and the rate of change decreases too. As it was discussed before, it is probably caused by the hysteresis of the PZT displacement versus voltage relation. The PZT has been borrowed from another lab, so the exact model name is unknown and its characteristic can't be verified in a

datasheet.



(a) (b)



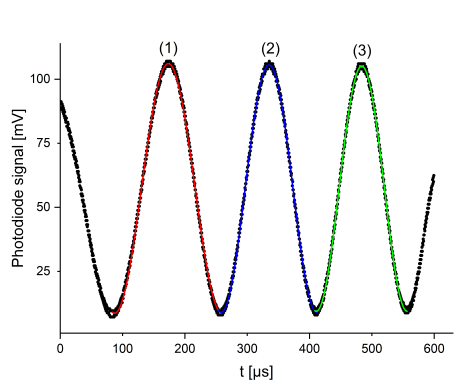
(c)

Figure 5.2: (a) – (b) Interference fringes for a 767 nm laser generated by a ramp signal with a frequency of 800 Hz and the voltage range 0 – 10 V (before the amplification). The visibility is 87.8%. Sines were fitted to each separate fringe in two ways, in (a) to the peaks (dentoed as a1, a2, a3, a4) and in (b) to the valleys (b1, b2, b3, b4, b5). The obtained periods for individual fringes with error bars are presented in (c). Their decrease is evident.

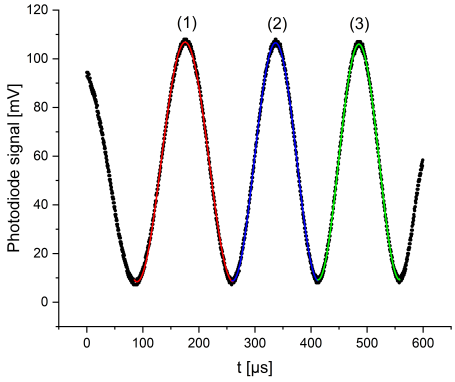
There was also a problem that the amplifier couldn't handle so high input voltage, which caused overheating and the distortion of the signal after some time. Because of that, the voltage range had to be lowered to 7 V. Figures 5.3 represent a few more example signals. Their widths were also estimated by fitting sines to the individual fringes. The obtained periods have been plotted in Fig. 5.4. The results confirm that it is not a coincidence and in each situation the period drops by about 10 - 13% between fringe 1 and 2, and by 5 - 6% between 2 and 3, so that is not a negligible effect.

Because of the non-linearity, it is impossible to fit one sine function to all the fringes within a rising slope of the ramp. A solution might be limiting the range of the fit by skipping some number of steps from the left and right. As it can be seen in Fig. 5.2c, for higher voltages, fringe widths don't change that much. Setting the fitting range to that region can be a good idea. It has been tried using the sine fitting algorithm proposed in the section 3.2 and the

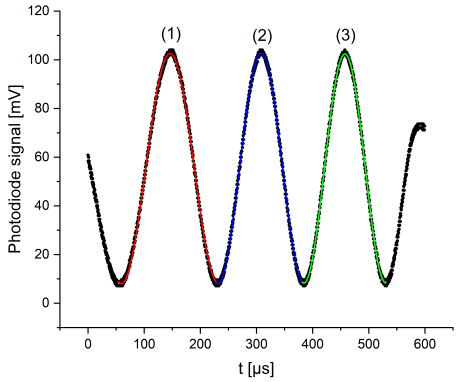
results are showed in Fig. 5.5. This fit is not perfect but may be sufficient for calculating differences in phases.



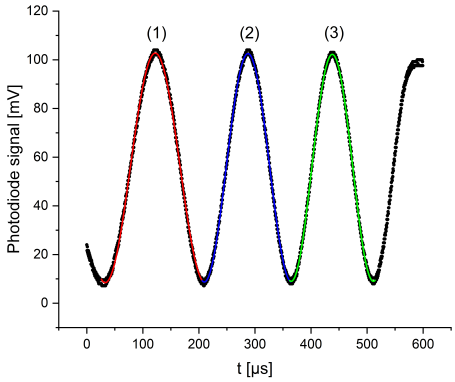
(a) Image 1
Visibility ≈ 87.4%



(b) Image 2
Visibility ≈ 87.5%



(c) Image 3
Visibility ≈ 87.1%



(d) Image 4
Visibility ≈ 87.1%

Figure 5.3: Output signals at the photodiode vs time for a ramp signal of the frequency of 800 Hz and voltage range 0 – 7 V. The visibilities are over 87% in each case. Sines were fitted to each individual fringe (1), (2) and (3) separately.

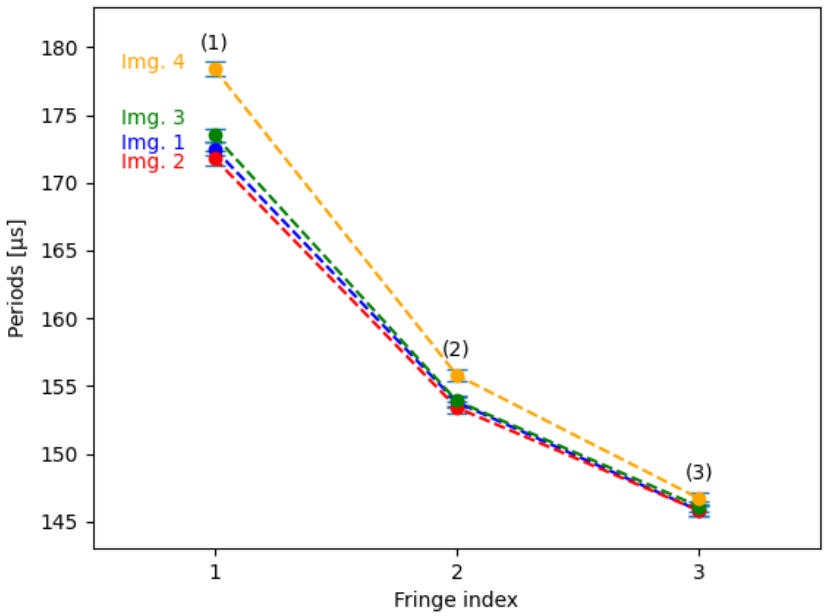


Figure 5.4: Widths of individual fringes (1), (2) and (3) obtained for signals in Images 1, 2, 3 and 4 in Fig. 5.3

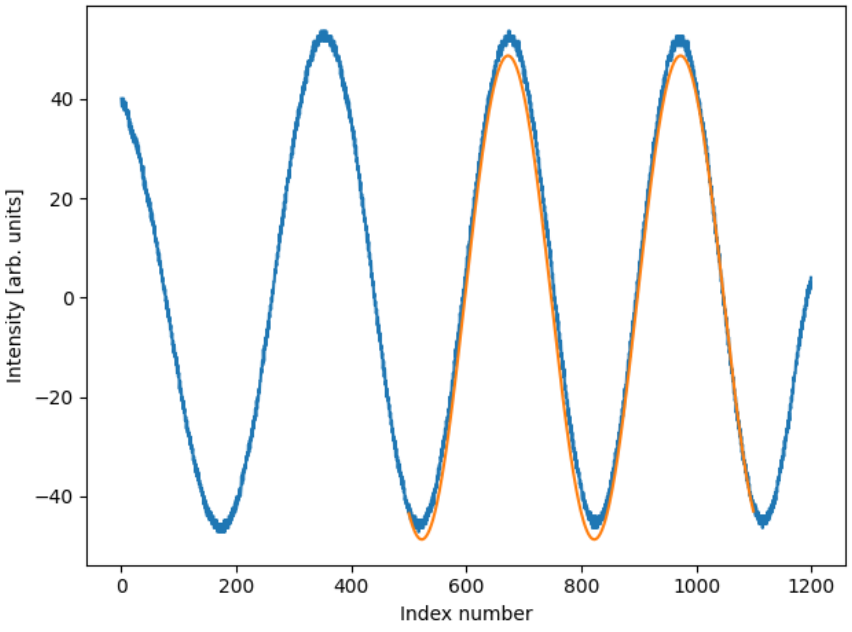


Figure 5.5: Intensity vs index number of data points. The sine fitting range was decreased to smaller number of points.

The other solution may be using the modified function 2.14 that has been discussed in chapter 2, which assumes the PZT is non-linear and its response may be modeled by an exponential function. Fig. 5.6 shows the result of fitting it to the potassium laser interference

signal. The visibility is over 88.4% and adjusted R-square very close to 1. The function appears to fit the data very well. It suggests the proposed model explains the origin of these differences in periods.

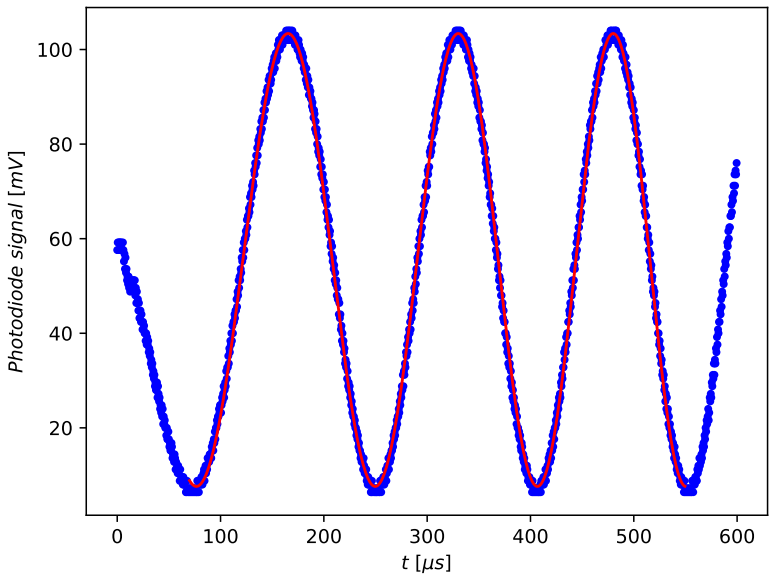
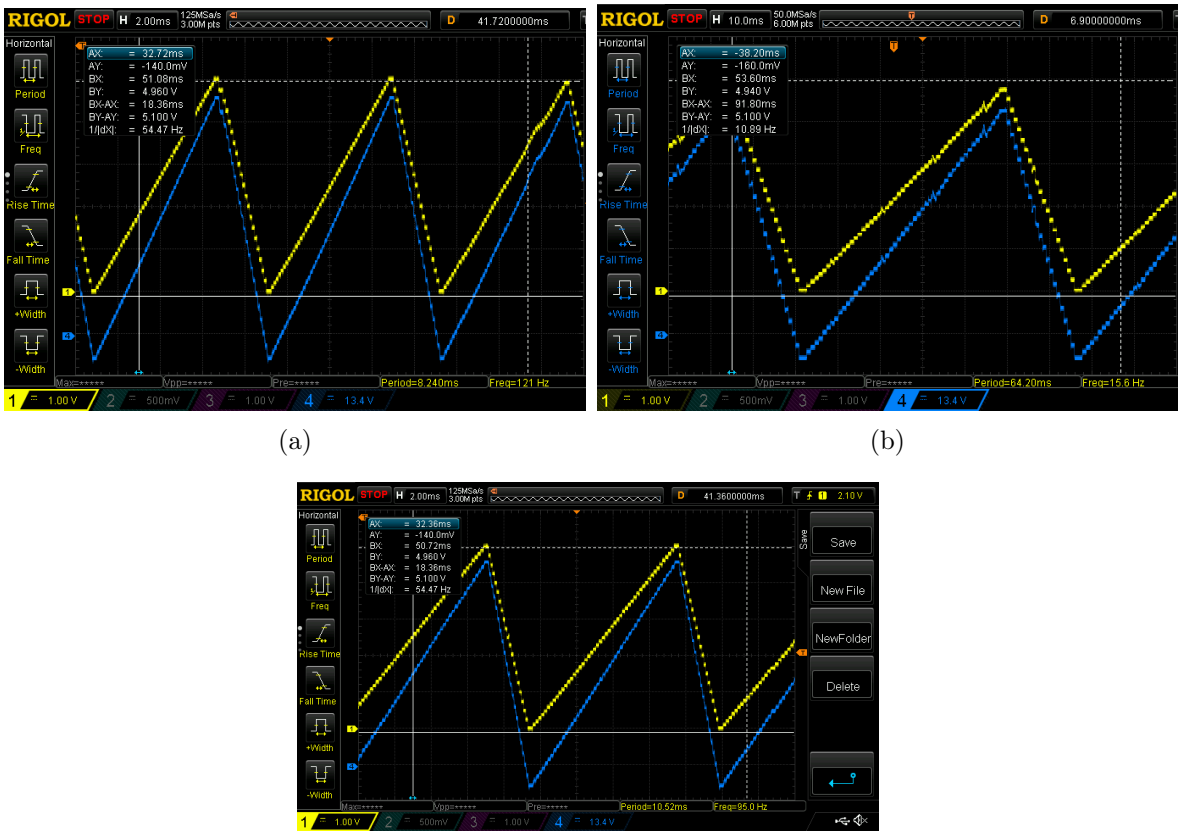


Figure 5.6: Least squares fit for the interference signal. The ramp signal's frequency is 800 Hz and the voltage ranges from 0 V to 7 V. Visibility $\approx 88.4\%$, Adj. $R^2 \approx 0.9992$.

However, fitting such a function is much more computationally intensive than the previous algorithm for sine fitting. For example in Python its execution time for 400 points was about 9200 μs, while for the previous one it was ca. 700 μs for the same number of points, that is over 13 times less. For precise laser frequency stabilization, the speed of the program is of crucial importance, that is why the former method is preferred as its important advantage is much better performance.

In order to be able to take control of the range of the signal that we want to fit the sine to in real time and also compensate the temperature drifts, the generator had to be replaced with the digital-to-analog converter. As Python is considered to be much easier than C++, the converters were firstly programmed in that language. The DAC resolution is 12-bit, so it can provide 4096 discrete output voltage levels. However, the maximum resolution couldn't be used because then the program was extremely slow and thus the signal frequency was significantly too low. The frequency can't be just set directly in the software as it is dependent on the overall code's efficiency and any additional delays that may be set between individual steps by the user. In this case no extra delays were set, because even without them, the program was not fast enough to ensure the desired frequency. Eventually, the resolution of the ramp had to be lowered to increase the frequency. The most optimal solution turned out to be 50 equal steps up and 20 down. If there was no falling slope, oscillations would be produced and therefore disrupt the signal of interest. Moreover, the I^2C clock speed was increased from default 100 kHz to almost 1 MHz. The results are shown in Fig. 5.7. When only the ramp signal was generated 5.7a (MCP4728_sawtooth.py), its frequency was 121 Hz. After adding signal readings by the ADC in each subsequent iteration on the rising slope, the frequency

dropped to just 15.6 Hz 5.7b. The code was modified by adding a queue for reading data (converters_multithreading.py) and then the obtained frequency raised to 95 Hz as shown in Fig. 5.7c. However, it was still insufficient to stabilize the laser with acceptable precision, especially since the necessary operations for calculating phase differences, generating error signal and PI control have not yet been added. It would substantially further slow down the program.

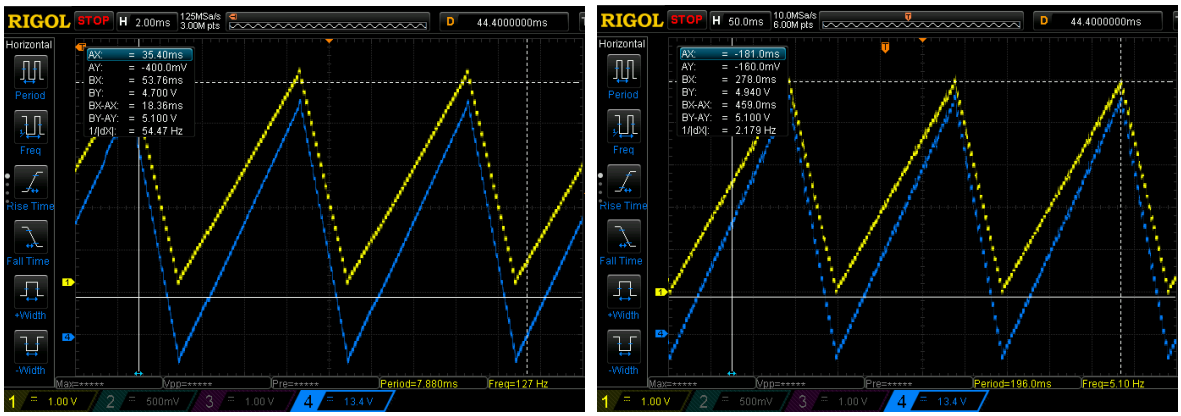


(a) (b)

(c)

Figure 5.7: The comparison of python programs' performances. Yellow – ramp signals generated by the DAC consisting of 50 voltage levels up and 20 down, blue – the signals after the amplification. (a) When only the ramp signal was generated, its frequency was 121 Hz. (b) After adding signal readings by the ADC in every step at the rising slope, the ramp signal's frequency dropped to 15.6 Hz. (c) Using a queue in Python to digitize the data, the frequency substantially increased to 95 Hz.

For the above reasons, C++ language was chosen instead. The results for the same settings are shown in Fig. 5.8. Unfortunately, when only generating the ramp (MCP4728_sawtooth.cc), its frequency was 127 Hz, so very similar to that in Python. For the complete program (transfer_interferometer_v3.cpp) with all the functionalities and necessary computations, the frequency was only ca. 5 Hz, so way too low. It has also been tried to stretch the I^2C clock speed above 1 MHz. Then the ramp signal, when only generating it, was around 280 Hz. However, it was less stable then and the frequency often jumped to lower values uncontrollably.



(a) (b)

Figure 5.8: The comparison of C++ programs performances. (a) When only generating the ramp signal, its frequency was 127 Hz. (b) The complete program for generating, reading, analysing the signals from both lasers, calculating phases and implementing PI controllers turned out to be very slow. The ramp frequency dropped to just 5.1 Hz.

The main program does not yet work as expected. There were many errors and most of them has already been resolved, however when the phases were saved to a file in real time, they showed up as NaN (not a number). As it was shown, the algorithm works well when applied to the saved data from the oscilloscope. It does not however work properly incorporated to the main program. The cause has not yet been found.

Chapter 6

Conclusions

The visibility of the observed interference fringes exceeded 87%, which is a pretty good quality. According to the presented theory the interferometric signals should be sinusoidal, but it turned out the period decreased as the applied voltage to the PZT increased. The model was modified by assuming that the piezo-electric actuator is not perfectly linear and its response may be approximated by an exponential function. It enabled to fit the function to a few fringes with high precision and the adjusted R^2 for that fit was very close to 1. Unfortunately, this approach can't be used for the phase extraction for laser stabilization purposes as it is considerably slower and thus prevents achieving acceptable accuracy. However, the fitting range can be changed in the software such that the sinusoidal relation sufficiently approximates the measured signal in the chosen range. Another solution might be trying to use other piezo-electric transducers that exhibit less hysteresis and choose the best one.

As Python is a high-level language and does not allow for further optimization, C++ was chosen instead because it is more efficient. Nevertheless, the performance of the code was still very slow. In the case of only generating ramp signal without reading and analysing data, there was not much difference with respect to the Python code. The DAC, that was used, communicated with RPi using I^2C interface which is capable of transmitting data up to 3.4 Mbps in the High-speed mode. Thus it was possible to increase the communication speed above 1 MHz achieving ramp signal of frequency up to 280 Hz, however it was less stable and the frequency fluctuated uncontrollably.

There is yet another kind of communication interface called SPI. Its speed may be up to 60 Mbps. That is why using DAC communicating through SPI instead of I^2C would be greatly advisable. It should allow to generate signals of significantly higher frequencies and a better resolution could be set. Moreover, the main code itself needs to be carefully analysed and optimized, because the frequency of the ramp signal dropped below just 6 Hz when all the operations were added. For comparison, in the paper that I based my research on, it was reported that the ramp frequency of almost 200 Hz was achieved, even though two lasers were stabilized simultaneously, as opposed to just one as in my code.

To speed up the code, parallel programming might be considered. The tasks could be split into separate threads. RPi 4B supports four individual threads. However the way of programming the converters should be changed as available libraries for SPI and I^2C are based on old CPU architecture that was present in previous versions of Raspberry Pi and supports only one thread.

Unfortunately, no laser has yet been stabilized using the developed system, because there were a lot of time-consuming problems and bugs that had to be resolved and eventually there has not been enough time left. However, the described troubles and possible solutions may be helpful to resolve them in the future and make the code more efficient and flawless.

Using Raspberry Pi seemed like a good idea due to its robust specifications compared to standard microcontrollers, but unfortunately, it posed many additional challenges. With Arduino, ready-made libraries for many popular converters can be easily found and there is no need to program them from scratch. However, many Arduino boards are not suitable for this project because they have too little RAM and a too low CPU clock frequency. Nevertheless, there are some exceptions that could be an alternative and replace the discontinued Digilent uC32 microcontroller. These are for example Arduino Due and Mega, whose specifications have been compared in Tab. 6.1.

Table 6.1: Comparison between Digilent uC32, Arduino Due and Arduino Mega specifications

Specs	**Digilent UC32**	**Arduino Due**	**Arduino Mega 2560 Rev3**
CPU Clock Speed	80 MHz	84 MHz	16 MHz
RAM	32 KB	96 kB	8 kB
operating voltage	3.3 V	3.3 V	5 V

Also, two less powerful microcontrollers could be used, one for each laser, to split tasks between them, but their operation should be somehow synchronized, which can pose additional difficulties.

Bibliography

[1] Wojciech Gawlik and Jerzy Zachorowski. Stabilization of diode-laser frequency to atomic transitions. *Optica Applicata*, 34(4):607–618, 2004.

[2] Lakhi Sharma, Atish Roy, Subhasis Panja, and Subhadeep De. Stabilizing frequency of a diode laser to a reference transition of molecular iodine through modulation transfer spectroscopy. *Atoms*, 11(5), 2023.

[3] Thomas Rieger and Thomas Volz. Doppler-free saturation spectroscopy. *Max Planck Institute fur Quantenoptek, Garching. http://www.ph.tum.de/studium/praktika/fopra/text/userguide-05.en.pdf*, 2004.

[4] R.W.P. Drever, John Hall, F. Kowalski, James Hough, G.M. Ford, A.J. Munley, and Hywel Ward. Laser phase and frequency stabilization using an optical resonator. *Appl. Phys. B*, 31:97–105, 06 1983.

[5] Eric D. Black. An introduction to Pound–Drever–Hall laser frequency stabilization. *American Journal of Physics*, 69(1):79–87, 01 2001.

[6] Gary C Bjorklund, MD Levenson, W Lenth, and C Ortiz. Frequency modulation (fm) spectroscopy: theory of lineshapes and signal-to-noise analysis. *Applied Physics B*, 32:145–152, 1983.

[7] Nur Ismail, Cristine Calil Kores, Dimitri Geskus, and Markus Pollnau. Fabry-pérot resonator: spectral line shapes, generic and related airy distributions, linewidths, finesses, and performance at low or frequency-dependent reflectivity. *Opt. Express*, 24(15):16366–16389, Jul 2016.

[8] J. Alnis, A. Matveev, N. Kolachevsky, Th. Udem, and T. W. Hänsch. Subhertz linewidth diode lasers by stabilization to vibrationally and thermally compensated ultralow-expansion glass Fabry-Pérot cavities. *Phys. Rev. A*, 77:053809, May 2008.

[9] S. Subhankar, A. Restelli, Y. Wang, S. L. Rolston, and J. V. Porto. Microcontroller based scanning transfer cavity lock for long-term laser frequency stabilization. *Review of Scientific Instruments*, 90(4), apr 2019.

[10] Shira Jackson, Hiromitsu Sawaoka, Nishant Bhatt, Shreyas Potnis, and Amar C. Vutha. Laser frequency stabilization using a transfer interferometer. *Review of Scientific Instruments*, 89(3), mar 2018.

[11] Michał Śmiałkowski and Michał Tomza. Highly polar molecules consisting of a copper or silver atom interacting with an alkali-metal or alkaline-earth-metal atom. *Phys. Rev. A*, 103:022802, Feb 2021.

[12] G. Uhlenberg, J. Dirscherl, and H. Walther. Magneto-optical trapping of silver atoms. *Phys. Rev. A*, 62:063404, Nov 2000.

[13] "Jones calculus". spie.org. https://spie.org/publications/fg05_p57-61_jones_matrix_calculus?SSO=1. Retrieved 2023-09-13.

[14] "Fringe Visibility". phys.libretexts.org. https://phys.libretexts.org/Bookshelves/Optics/BSc_Optics_(Konijnenberg_Adam_and_Urbach)/05%3A_Interference_and_coherence/5.10%3A_Fringe_Visibility. Retrieved 2023-09-17.

[15] Krzysztof Kaźmierczak and Radosław Przysowa. Standard sine fitting algorithms applied to blade tip timing data. *Journal of KONBiN*, 30(1):21–30, 2015.

[16] *MCP4728 datasheet. http://ww1.microchip.com/downloads/en/devicedoc/22187e.pdf. Retrieved 2023-09-10.*

[17] *ADS1256 datasheet. https://www.ti.com/lit/ds/symlink/ads1256.pdf. Retrieved 2023-09-10.*

[18] See https://github.com/adamswiniarskixc/Transfer_interferometer_for_laser_frequency_stabilization for the Raspberry Pi codes.

[19] https://github.com/vuthalab/Transfer-Interferometer/tree/master.

[20] Mike McCauley. BCM2835 library. https://github.com/janne/bcm2835/tree/master.

[21] Zatrac. ADS1256-driver. https://github.com/Zatrac/ADS1256-driver/tree/master.