Fast Rotational Control in Levitated Optomechanics for Creation of Rotational Fock States

Ka Kit Kelvin Ho

A thesis presented for the degree of
Master in Science

Supervisor: Professor Peter F. Barker

Department of Physics and Astronomy
University College London
April 2023

UCL DEPARTMENT OF PHYSICS & ASTROMOMY

Submission of coursework for Physics and Astronomy course **PHAS0097/PHAS0096/PHAS0048** 2022/23

Please sign, date and return this form with your coursework by the specified deadline.

DECLARATION OF OWNERSHIP

I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.

I confirm that all work will also be submitted electronically and that this can be checked using the JISC detection service, Turnitin®.

I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

Signed 

PrintName Ka Kit Kelvin Ho

Dated 13 April 2023

Title	Date Received	Examiner	Examiner's Signature	Mark

Acknowledgments

This project was only made possible by the support and guidance of Prof. Peter Barker. Thank you for taking me on board and showing me the ropes. To Antonio, who always provides positive encouragements despite his otherwise grumpy demeanour. To Markus, who provides comic relief with his deadpan humour and has taught me the very important life skill of learning how to tell apart Austrian and German sausages. To Jonathan, who was always there in C15 and have given me so much help in the lab. To Hayden, for reassuring me that it will all turn out fine, and reminding me how good McDonald's in Hong Kong is. To Fiona, who has provided much needed help when the Red Pitaya was driving me crazy.

Special mention to Prof. Tania Monteiro for your lectures on optomechanics and the enlightening discussions on the theory side of things. Also to Julian, for your chill vibes, coffee machine and the one time on Friday night where you let me back into the building. And of course, to Vincenzo, who is always there to entertain us and to distract us from the fact that his fume cupboard is still sitting in the mezzanine corridor after months.

Most importantly, I have to thank my family and friends for their unwavering support despite not knowing what it is that I am actually doing.

Abstract

The boundary between quantum and classical effects has never been well-defined in the history of the theory. Quantum mechanics has passed all tests involving microscopic objects with flying colours, yet is remarkably different from our everyday experience of reality. The theory permits the existence of a macroscopic quantum system, and careful preparation of an experiment should be able to bring about such a system. A proposed system is to levitate a nanorotor by an optical tweezer and rotate it to a high angular frequency. When the light field is turned off, it is expected that a measurement would reveal that the nanorotor has settled into a well defined rotational state. Beyond being a test of fundamental quantum mechanics, it has interesting applications in creating more accurate force sensors. This project has built tools to create such a macroscopic quantum system, and has demonstrated changes in rotational peaks as a result of a change in the trapping field's polarisation.

(Word Count: 7499)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Levitated optomechanics concerns the use of light to control the motion of a levitated nanoparticle. It is a growing field which has recently enjoyed numerous scientific breakthroughs [1][2]. In particular, it had been remarkably useful in the study of fundamental Physics, such as non-equilibrium and nano-thermodynamics [3], and mesoscopic quantum mechanics [4].

The advent of quantum mechanics has not only revolutionised our comprehension of physics and technology, but it has also transformed our perception of reality [5]. According to the Copenhagen interpretation, a quantum system remains in a state of superposition until observed by an observer. However, it does not explain how large a quantum system can be. Although quantum theory is highly effective in describing microscopic systems, it fails to account for quantum effects on a macroscopic scale, despite the theory not precluding such effects. This inconsistency is vividly illustrated in the renowned thought-experiment of Schrödinger's cat [6][7]. Despite the remarkable success of quantum mechanics in improving our comprehension of nature and catalysing technological advancement, much of it remains a mystery that is yet to be unravelled.

A key challenge in creating quantum systems with macroscopic objects is environmental decoherence. However, optomechanical experiments can levitate an object, while cooling methods can remove thermal effects from the system [8]. This field has achieved significant milestones, such as the realisation of ground-state cooling [9][10] and the creation of highly accurate sensors [11]. Optomechanics in the rotational and librational regime, which addresses rotational degrees of freedom instead of translational degrees of

freedom, is an area that has been relatively less explored in levitated optomechanics [12].

This project built tools to test for quantum effects on a much larger scale, compared to previous endeavours via rotational degrees of freedom [13][14][15][16][17]. The objective is to levitate non-spherical nanoparticles and rotate them with the use of a linearly polarised trapping field. By quickly varying the field polarisation angle, the particle can be brought to high rotational speeds [18]. It is hypothesised that the particle will only be allowed be able to rotate at discrete rotational frequencies due to its quantum nature, and this discrete nature will be observable upon rotating the particle at high rotational speeds.

The project so far has achieved success in the construction of the apparatus required in performing this experiment. The first part of this thesis would describe the theoretical basis underpinning this project, giving a brief overview of levitated optomechanics, how Stokes vector and Mueller Calculus are used as a mathematical expression of optical elements and polarisation effects, and how an optical centrifuge operates. The second part describes the experimental aspect of the project, including the construction of a polarimeter, the characterisation of the electro-optical polarisation controller and the building of the trapping and detection system.

Part I

Theory

Chapter 2

Optomechanics

Optomechanics is concerned with the interaction between light and matter. While light has no mass, it has momentum, and the collective action of light interacting with matter imparts momentum that is called radiation pressure. This force can be brought upon to act on a mechanical oscillator, and the interaction between the optical and the mechanical system thus forms an optomechanical system.

Levitated systems offer the unique advantage of being highly isolated from the environment. This is achieved by holding the object (which acts as a mechanical oscillator) in an intense optical field. Thus, levitated systems suffer from no clamping losses and the mechanical centre-of-mass motion is decoupled from internal defects [19]. One notable success of this system is in achieving cooling to the ground state from room temperature in 2020 by Delić et al [9].

The lack of tethering to the environment also gives levitated systems unique degrees of freedom, such as rotational degrees of freedom. Recent work in exploring such aspects include simultaneously cooling six dimensions of a levitated nanoparticle by Pontin et al [20].

2.1 Trapping Potential

To achieve levitation, the particle sits in an optical field $\mathbf{E}$ and experiences a gradient force given by:

$$\mathbf{F} = \frac{\alpha}{4}\nabla[\mathbf{E}^2], \tag{2.1}$$

where α is the real part of the particle's polarisability. The force is positive in the direction of the gradient, which means it points towards the direction at which the intensity of the electric field grows, which suggests that the force has a restorative nature. Thus, by taking the additional simplifying assumptions that the focused beam has a Gaussian profile and that the particle's displacement is small with respect to the beam waist, the force can be treated as a linear restoring force [1]:

$$F_j = -k_j j \qquad j \in \{x, y, z\}, \tag{2.2}$$

where x, y, z are the orthogonal translational degrees of freedom. The usual convention is to take z as the beam propagation direction and x, y to be the direction of motions which are transverse to the propagation direction. This allows us to treat the movement of the particle as a harmonic oscillator.

An example of polarisability is given by that of a sphere with volume $V = \frac{4}{3}\pi R^3$ [19]:

$$\alpha = 3\epsilon_0 V \frac{n^2 - 1}{n^2 + 2}. \tag{2.3}$$

It is clear that the polarisability grows with volume and refractive index. This gives the unique ability to tune the force by performing trapping experiments with particles of different size and different material.

2.2 Mechanical Oscillator

The motion of a trapped particle can be treated as a damped harmonic oscillator. As such, an equation of motion can be written:

$$\ddot{x} + \Gamma\dot{x} + \Omega^2 x = \frac{F(t)}{m}, \tag{2.4}$$

where $x(t)$ is the displacement of the particle as a function of time. This is a damped harmonic oscillator with mass m, a driving force given by $F(t)$ and a damping term given by Γ. The term Ω is related to the spring constant by $\Omega = \sqrt{\frac{k_q}{m}}$. Thus, the particle's motion would be governed by the solution to the equation $x(t)$. This can be measured by detecting the particle's scattered light. While the particle would oscillate at its eigenfrequency, its oscillation would also be affected by gas damping and fluctuating thermal Langevin noise. Thus, it is often easier to analyse the signal in frequency space, which

allows us to discern the individual contributions of different frequencies.

This can be obtained by taking the Fourier transform of the signal, defined as:

$$x(\omega) \equiv \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt, \tag{2.5}$$

with the inverse Fourier Transform defined as:

$$x(t) \equiv \frac{1}{2\pi} \int_{-\infty}^{\infty} x(\omega) e^{i\omega t} d\omega. \tag{2.6}$$

The equation of motion in the frequency domain is:

$$-\omega^2 x(\omega) + i\omega\Gamma x(\omega) + \Omega^2 x(\omega) = \frac{F(\omega)}{m} \tag{2.7}$$

Thus, the solution is:

$$x(\omega) = \frac{F(\omega)}{m(\Omega^2 - \omega^2 + i\omega\Gamma)} \tag{2.8}$$

$$= F(\omega)\chi(\omega) \tag{2.9}$$

Where $\chi(\omega)$ is defined as the mechanical susceptibility:

$$\chi(\omega) \equiv \frac{1}{m(\Omega^2 - \omega^2 + i\omega\Gamma)} \tag{2.10}$$

In practice, many time traces, each of time τ are taken, and the truncated Fourier transform of each signal is performed. This is defined to be[1]

$$\tilde{x}(\omega) \equiv \frac{1}{\sqrt{\tau}} \int_{0}^{\infty} x(t) e^{-i\omega t}. \tag{2.11}$$

Taking the average gives the spectral density $|\langle \tilde{x}(\omega)^2 \rangle|$. In the limit of $\tau \to \infty$, the Wiener-Khinchin theorem states that spectral density is equivalent to the noise power spectral density $S_{xx}(\omega)$ [21]:

$$\lim_{\tau \to \infty} |\langle \tilde{x}(\omega)^2 \rangle| = S_{xx}(\omega), \tag{2.12}$$

[1] In definitions like that given in [21], the Fourier transform and the corresponding truncated Fourier transform is given with the positive exponent $e^{i\omega t}$. However, since it is defined with the negative exponent in the earlier section, I am following the same convention and defining the truncated Fourier transform with a negative exponent. This is purely a matter of convention.

where the power spectral density (PSD) is defined to be the Fourier transform of the autocorrelation function:

$$S_{xx}(\omega) \equiv \int_{-\infty}^{\infty} \langle x(t)x(0)\rangle e^{-i\omega t} dt. \tag{2.13}$$

The two formulas above tell us that the area under the PSD is equal to the variance of the mechanical displacement:

$$\int_{-\infty}^{\infty} S_{xx}(\omega)\frac{d\omega}{2\pi} = \langle x^2 \rangle. \tag{2.14}$$

Suppose $F(\omega)$ is a stochastic thermal driving force. Then, the fluctuation-dissipation theorem relates the PSD with the temperature and the linear response of the system, given by the mechanical susceptibility defined earlier:

$$S_{xx}(\omega) = -2\frac{k_B T}{\omega}\mathrm{Im}\chi(\omega). \tag{2.15}$$

At weak damping ($\Gamma \ll \Omega$), which happens at high vacuum conditions, the PSD thus displays Lorentzian peaks of width Γ at $\omega = \pm\Omega$. This explains the characteristic peaks observed in optomechanical PSDs [1][21].

Chapter 3

Polarisation

This section derives the mathematics for describing light with an arbitrary state of polarisation (SOP), as well as how it is possible to describe the transformation different optical elements have on a light's SOP. Much of the discussion is motivated by what is given in [22].

3.1 Stokes Parameters

The propagation of electromagnetic waves are described by the wave equation[1]:

$$\nabla^2 \mathbf{E} = \frac{1}{v^2} \frac{\partial^2}{\partial t^2} \mathbf{E}. \tag{3.1}$$

For a wave travelling in the z direction, the solutions are[2]:

$$E_x(z,t) = E_{0x} \cos(\omega t - \mathbf{k} \cdot \mathbf{r} + \delta_x), \tag{3.2}$$

$$E_y(z,t) = E_{0y} \cos(\omega t - \mathbf{k} \cdot \mathbf{r} + \delta_y). \tag{3.3}$$

Rewriting the phase constants as $\delta = \delta_y - \delta_x$, and taking the real intensity, the two equations can be expressed as[3]:

$$\frac{E_x(z,t)^2}{E_{0x}^2} + \frac{E_y(z,t)^2}{E_{0y}^2} - \frac{2E_x(z,t)E_y(z,t)}{E_{0_x}E_{0y}} \cos\delta = \sin^2\delta. \tag{3.4}$$

[1] This arises from coupling Maxwell's equations.

[2] Maxwell's equations (in free space) tells us that $\nabla \cdot \mathbf{E} = 0$ and $\nabla \cdot \mathbf{B} = 0$, which upon substitution into the wave equation shows that electromagnetic waves are transverse and will have no component which is parallel to the direction of motion.

[3] The propagator gets cancelled out as a common factor on both sides of the equation

This is an equation of an ellipse and 3.4 is known as the polarisation ellipse. Light in general is thus elliptically polarised.

The orientation and ellipticity angles can also be used to describe the polarisation ellipse. These are related to the amplitude and phase by:

$$\tan 2\psi = \frac{2E_{0x}E_{0y}}{E_{0x}^2 - E_{0y}^2}\cos\delta, \qquad 0 \leq \psi \leq \pi, \tag{3.5}$$

$$\sin 2\chi = \frac{2E_{0x}E_{0y}}{E_{0x}^2 - E_{0y}^2}\sin\delta, \qquad -\pi/4 < \chi \leq \pi/4. \tag{3.6}$$

However, these polarisation ellipses are neither directly measurable nor observable. In general, only the intensity of light can be measured, and polarisation properties have to be determined based on the measured intensities. To relate intensities and polarisation parameters, the time average of 3.4 is taken:

$$\frac{\langle E_x(z,t)^2\rangle}{E_{0x}^2} + \frac{\langle E_y(z,t)^2\rangle}{E_{0y}^2} - \frac{2\langle E_x(z,t)E_y(z,t)\rangle}{E_{0_x}E_{0y}}\cos\delta = \sin^2\delta, \tag{3.7}$$

where the time-average $\langle\cdot\rangle$ is defined as:

$$\langle E_i(z,t)E_j(z,t)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^t E_x(z,t)E_y(z,t)dt. \tag{3.8}$$

Each of the time-averaged terms can be calculated:

$$\langle E_x^2(z,t)\rangle = \frac{1}{2}E_{0x}^2, \tag{3.9}$$

$$\langle E_y^2(z,t)\rangle = \frac{1}{2}E_{0y}^2, \tag{3.10}$$

$$\langle E_x(z,t)E_y(z,t)\rangle = \frac{1}{2}E_{0x}E_{0y}\cos\delta. \tag{3.11}$$

$$\tag{3.12}$$

Returning to 3.7, multiply both sides with $4E_{0x}^2E_{0y}^2$, and substituting the time-averaged values to obtain:

$$4E_{0y}^2\langle E_x(z,t)^2\rangle + 4E_{0y}^2\langle E_y(z,t)^2\rangle - 8\langle E_x(z,t)E_y(z,t)\rangle\cos\delta = (2E_{0x}E_{0y}\sin\delta)^2, \tag{3.13}$$

$$4E_{0y}^2\langle E_x(z,t)^2\rangle + 4E_{0y}^2\langle E_y(z,t)^2\rangle - 8\langle, E_x(z,t)E_y(z,t)\rangle\cos\delta = (2E_{0x}E_{0y}\sin\delta)^2, \quad (3.14)$$

$$2E_{0x}^2E_{0y}^2 + 2E_{0x}^2E_{0y}^2 - (2E_{0x}E_{0y}\cos\delta)^2 = (2E_{0x}E_{0y}\sin\delta)^2, \quad (3.15)$$

$$2E_{0x}^2E_{0y}^2 + (E_{0x}^4 + E_{0y}^4) - (E_{0x}^4 + E_{0y}^4) + 2E_{0x}^2E_{0y}^2 - (2E_{0x}E_{0y}\cos\delta)^2 = (2E_{0x}E_{0y}\sin\delta)^2. \quad (3.16)$$

Addition and subtraction of the term $E_{0x}^4 + E_{0y}^4$ allow us to form perfect squares, and upon grouping the terms:

$$\therefore (E_{0x}^2+E_{0y}^2)^2+(E_{0x}^2-E_{0y}^2)^2-(2E_{0x}E_{0y}\cos\delta)^2 = (2E_{0x}E_{0y}\sin\delta)^2, \quad (3.17)$$

Each term can be expressed as:

$$S_0^2 - S_1^2 - S_2^2 = S_3^2, \quad (3.18)$$

or:

$$S_0^2 = S_1^2 + S_2^2 + S_3^2, \quad (3.19)$$

where each term is equal to:

$$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \begin{pmatrix} E_{0x}^2 + E_{0y}^2 \\ E_{0x}^2 - E_{0y}^2 \\ 2E_{0x}E_{0y}\cos\delta \\ 2E_{0x}E_{0y}\sin\delta \end{pmatrix}. \quad (3.20)$$

These are known as the Stokes Parameters, in which S_0 represents the total intensity, S_1 represents the extent of which the light is horizontal or vertical polarised, S_2 represents the extent of which the light is $+45°$ or $-45°$ linearly polarised and S_3 represents the extent of which the light is left-hand or right-hand circularly polarised. Expressing them as $(S_0, S_1, S_2, S_3)^T$ is known as the Stokes vector, which is usually normalised by dividing the vector by the S_0. The normalised vector can be represented on the unit sphere in which S_1, S_2 and S_3 correspond to the x, y and z coordinates. The vector is subtended by an azimuthal angle of 2ψ and an inclination angle of 2χ. Such a representation is called the Poincaré Sphere, shown in figure 3.1.

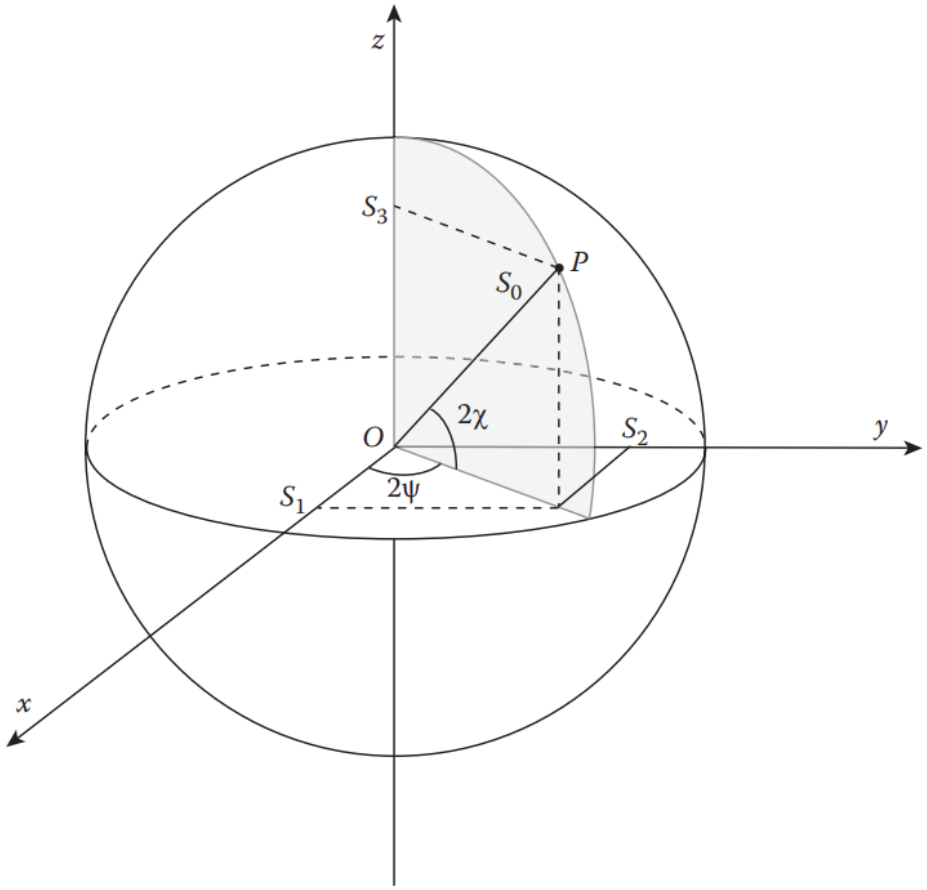

Figure 3.1: Poincaré representation of a Stokes vector, taken from [22].

Stokes parameters are an extremely useful way of describing polarisation as it relates measured intensities with the polarisation properties. It is also the most general description of polarisation, as it can describe unpolarised light or partially polarised light. The effect of optical elements on light can be described mathematically as transformations of the Stokes parameters.

3.2 Mueller Calculus

In a general optical system, light of initial Stokes parameters S_i enters into one or more optical elements which can be represented by a Meuller matrix M, and leaves the system as output S_o. This can be represented as:

$$S_o = MS_1 \tag{3.21}$$

If M represents a series of optical elements in which the initial light passes through in the order of $M_1, M_2, M_3, ..., M_n$, the above representation becomes:

$$S_o = (M_n ... (M_3 (M_2(M_1 S_i)))) = M_n ... M_3 M_2 M_1 S_i \tag{3.22}$$

3.2.1 Linear Polariser

A linear polariser modulates the amplitude E_x and E_y by a factor p_x and p_y respectively. The Mueller matrix representation is:

$$M_{LP} = \frac{1}{2}\begin{pmatrix} p_x^2 + p_y^2 & p_x^2 - p_y^2 & 0 & 0 \\ p_x^2 - p_y^2 & p_x^2 + p_y^2 & 0 & 0 \\ 0 & 0 & 2p_x p_y & 0 \\ 0 & 0 & 0 & 2p_x p_y \end{pmatrix} \tag{3.23}$$

Thus, a horizontal polariser which only permit E_x to pass through would have a Mueller matrix of:

$$M_{HLP} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.24}$$

Wave Plates

Wave plates are made of a birefringent material in which E_x and E_y encounter different refractive indices as it passes through the material. As such, E_x and E_y would travel at different speeds, resulting in a phase difference in the output light. Thus, wave plates have a fast and slow axis. If E_x is parallel to the fast axis and E_y is parallel to the slow axis, E_y would accumulate a material-dependent phase lag of ϕ with respect to E_x. As a wave plate's purpose is to affect the wave's phase, it can be parameterised by the effect it has on a wave's phase, and has a matrix representation of:

$$M_{WP}(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\phi & -\sin\phi \\ 0 & 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{3.25}$$

Common waveplates include the half wave plate (HWP) and the quarter wave plate (QWP).

The HWP ($\phi = \pi$) reverses the ellipticity and orientation angle, which means it can change the orientation of an incoming linearly polarised light. The matrix is written as:

$$M_{HWP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{3.26}$$

The QWP ($\phi = \frac{\pi}{2}$, fast axis vertical) has the property of transforming linearly polarised light into circularly polarised light and vice versa. The matrix is given by:

$$M_{QWP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.27}$$

3.2.2 Rotators

The aforementioned matrices take the fast axis to be parallel to the x axis. However, this is a very restricted application, and often the linear polarisers or wave plates are rotated to act on incoming waves in different ways.

A rotator represents a rotation in coordinates. This rotates the orientation angle but does not change the ellipticity angle as a coordinate transformation is not a physical process which affects the physical property of the light. A physical rotation by angle θ is represented by:

$$M_{rot}(2\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta & 0 \\ 0 & -\sin 2\theta & \cos 2\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.28}$$

Note that the physical rotation of θ corresponds to a rotation matrix 2θ due to the Stokes vector being in the intensity domain.

Thus, incoming light can be treated as being in the lab frame, while the optical element has its own frame. Light has to first be transformed from the lab frame to the optical element's frame, operated on by the optical element and then transformed back to the lab frame to be the output light. This can be represented as:

$$S_o = (M_{rot}(-2\theta) M M_{rot}(2\theta)) S_i \tag{3.29}$$

The associative nature of matrix multiplication allows the product

$$M_{rot}(-2\theta) M M_{rot}(2\theta)$$

to be evaluated for optical elements at different orientations. For example,

wave plates can be expressed as:

$$M_{HWP}(4\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 4\theta & \sin 4\theta & 0 \\ 0 & -\sin 4\theta & \cos 4\theta & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{3.30}$$

$$M_{QWP}(2\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2 2\theta & \sin 2\theta \cos 2\theta & -\sin 2\theta \\ 0 & \sin 2\theta \cos 2\theta & \sin^2 2\theta & \cos 2\theta \\ 0 & \sin 2\theta & -\cos 2\theta & 0 \end{pmatrix} \tag{3.31}$$

In general, for a wave plate with a ϕ phase shift, the matrix is expressed as:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2 2\theta + \cos\phi \sin^2 2\theta & (1-\cos\phi)\sin 2\theta \cos 2\theta & -\sin\phi \sin 2\theta \\ 0 & (1-\cos\phi)\sin 2\theta \cos 2\theta & \sin^2 2\theta + \cos\phi \cos^2 2\theta & \sin\phi \cos 2\theta \\ 0 & \sin\phi \sin 2\theta & -\sin\phi \cos 2\theta & \cos\phi \end{pmatrix}$$

Chapter 4

Optical Centrifuge

The Optical centrifuge is a tool invented in 1999 for research in molecular physics [23]. These centrifuges allow for the rotation of suspended molecules to rotational frequencies that are high enough to pull apart the molecule. Rotating molecules to specified frequencies allows for selective bonds to be broken, offering a way to create molecules with only the desired atoms [24]. Additionally, the optical centrifuge provides a way to create molecular super-rotors, which are molecules rotated to a very high frequency. This exhbitis unique quantum mechanical properties like periodic revivals and long-term wave packet evolution [25]. Additionally, the large energies associated with super-rotors offer a way to study molecule–molecule or molecule–surface scattering [26].

We similarly construct an optical centrifuge for rotating nano-particles. These are a thousand times larger than the molecules, and as a result have very different energy levels, frequencies and coherent lifetimes. Thus this offers a way to create quantum states with much longer lifetimes, and due to the nanoparticles' massive nature, it can be used to create highly sensitive quantum gyroscopic force sensors [13] [14], or even be used in quantum gravity tests.

4.1 Rotation

The process of rotating a nanorotor or a molecule can be understood as a scattering process [27]. Raman scattering is a form of inelastic photon scattering in which photons scattering from a molecule would have some

of its energy absorbed before being re-emitted. The energy absorbed by the molecule would subsequently cause the molecule to be promoted to a higher energy level. Such a process is known as a Raman transition [28][29]. However, there are certain selection rules to be followed. As each photon has an energy of one unit (in units of $\hbar$), a single Raman transition can only change the energy of the molecule by no more than 2 units. Such a process is employed in molecular physics to create an optical centrifuge, which exploits Raman transitions to raise a molecule's rotational angular momentum to high levels [25]. As each photon only has one unit of internal angular momentum, each transition can only raise the molecule's angular momentum J by 2 units. Thus, the rotational ladder has to be climbed successively and in many transitions, before reaching the desired rotational level.

The physical implementation of a rotation is achieved using polarised light. For example, when circularly polarised light is shone onto a molecule, each photon imparts a one unit of momentum onto the molecule. This gives a torque to the molecule, and due to the polarisation vector being perpendiular to the plane of propagation, the molecule would also rotate in the plane perpendiular to the direction of light propagation. However, this approach does not offer precise control over the rotation. Even as the light is switched off, the particle can continue to rotate as it still has residual momentum. Additionally, circularly polarised light has its polarisation vector rotating at a frequency comparable to the frequency of light, which is too high for use in gradually rotating nanoparticles to well-defined rotational frequencies. As a result, an alternative method has to be used to perform alignment by slowly varying the alignment of the levitated particle.

4.2 Alignment

However, if linearly polarised light is used instead, the polarisation vector in the electric field would cause the rotor to have an induced dipole moment that can be represented as $\hat{\alpha} \cdot \mathbf{E}(t)$, in which

$$\mathbf{p} = \hat{\alpha} \tag{4.1}$$

represents the polarisability tensor of the rotor. In turn, the induced dipole moment interacts with the electric field, with an energy of [30]:

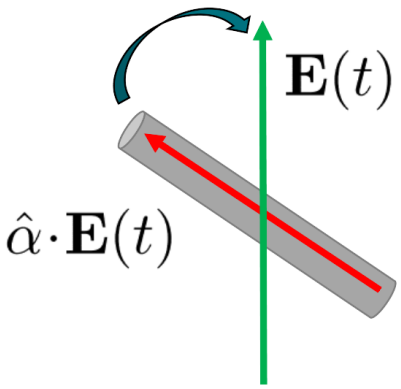

Figure 4.1: Diagram showing how the nanoparticle or nanorotor, shown as the grey cylinder with an induced dipole moment shown by the red arrow, has a torque which pushes it to align with the electric field shown as the green arrow.

$$U = -\frac{1}{2}\mathbf{E}(t) \cdot \hat{\alpha} \cdot \mathbf{E}(t) \tag{4.2}$$

The minimum of this energy is when the polarisability tensor aligns with the electric field. Assuming the rotor is of an axisymmetric object with a long axes, and the dipole moment is parallel to this long axes, the rotor would tend to line up with the electric field. When the rotor is not aligned, there would be a torque which pushes it to alignment. This is shown in figure 4.1.

To ensure that the rotor is to be adiabatically aligned with the light field, the field would first rotate very slowly, dragging the rotor along the polarisation axes in a slow and controlled manner. Subsequently, the rotation rate would accelerate, dragging the rotor to rotate at a higher and higher frequency. This brings the rotor into a higher frequency state. Once the desired frequency is reached, the light field can then be turned off, and measurements can be made to determine whether the particle has settled into a well-defined rotational Fock State or other states.

Part II

Implementation

Chapter 5

Polarimeter

The objective is to construct a free-space polarimeter which is able to identify the SOP of incoming light by measuring its intensity and calculating the Stokes parameters. Additionally, it is also desirable for the polarimeter to be fast and automated. The motivation for this will become clear in later sections, as we will be using a electro-optical modulating polarisation controller (EOM) to change the polarisation of the input laser. Characterising this device requires measuring the polarisation of light generated from many different input voltages. Thus, reducing the speed of each measurement will result in a faster characterisation of the EOM. Additionally, automation is essential as needing to manually perform every measurement for characterising the EOM would be too time-consuming.

We ultimately decided to use the rotating QWP method to measure polarisation. A discussion of the classical method is given in the appendix.

5.1 Rotating Quarter-Waveplate Method

Instead of requiring multiple optical elements and detectors, only a QWP on a rotating mount, a photodiode and a polariser is required for this method.

In this approach, incoming light passes through a QWP and then through a horizontal polariser before being measured by a detector. Mueller calculus gives matrix representations of different optical elements. The first 4×4 matrix represents a quarter wave plate, and the second represents a horizontal polariser. The output Stokes parameters are [31]:

$$\begin{pmatrix} S_0' \\ S_1' \\ S_2' \\ S_3' \end{pmatrix} = \frac{1}{2} \underbrace{\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\text{Polariser}} \times \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2 2\theta & \sin 2\theta \cos 2\theta & -\sin 2\theta \\ 0 & \sin 2\theta \cos 2\theta & \sin^2 2\theta & \cos 2\theta \\ 0 & \sin 2\theta & -\cos 2\theta & 0 \end{pmatrix}}_{\text{Quarter wave plate}} \times \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}. \quad (5.1)$$

The final intensity is given by S_0', which is:

$$I(t) = \frac{1}{2}\left(S_0 + S_1 \cos^2 2\theta + S_2 \sin 2\theta \cos 2\theta - S_3 \sin 2\theta\right). \quad (5.2)$$

Application of trigonometric identities and half angle formulas yield:

$$I(\theta) = \left[\left(S_0 + \frac{S_1}{2}\right) + \frac{S_1}{2}\cos 4\theta + \frac{S_2}{2}\sin 4\theta - S_3 \sin 2\theta\right], \quad (5.3)$$

where the coefficients are related to the Stokes parameters by:

$$A = \left(S_0 + \frac{S_1}{2}\right), \quad (5.4)$$

$$B = S_3, \quad (5.5)$$

$$C = \frac{S_1}{2}, \quad (5.6)$$

$$D = \frac{S_2}{2}. \quad (5.7)$$

These coefficients can be found by standard Fourier analysis:

$$A = \frac{1}{\pi}\int_0^{2\pi} I(\theta)d\theta, \tag{5.8}$$

$$B = \frac{2}{\pi}\int_0^{2\pi} I(\theta)\sin 2\theta d\theta, \tag{5.9}$$

$$C = \frac{2}{\pi}\int_0^{2\pi} I(\theta)\cos 4\theta d\theta, \tag{5.10}$$

$$D = \frac{2}{\pi}\int_0^{2\pi} I(\theta)\sin 4\theta d\theta. \tag{5.11}$$

Thus the intensity can be represented as a truncated Fourier Series with coefficients A, B, C and D. In practice, the QWP rotate in discrete steps:

$$I(\theta_j) = A - B\sin 2\theta_j + C\cos 4\theta_j + D\sin 4\theta_j, \quad j = 1, 2, 3, ..., N, \tag{5.12}$$

where N is the number of points required for the rotator to move over a full/half period.

Considering the discrete Fourier series:

$$A = \frac{2}{N}\sum_1^N I(\theta_j), \tag{5.13}$$

$$B = \frac{4}{N}\sum_1^N I(\theta_j)\sin 2\theta_j, \tag{5.14}$$

$$C = \frac{4}{N}\sum_1^N I(\theta_j)\cos 4\theta_j, \tag{5.15}$$

$$D = \frac{4}{N}\sum_1^N I(\theta_j)\sin 4\theta_j. \tag{5.16}$$

The Stokes parameters can be recovered from:

$$S_0 = A - C, \tag{5.17}$$

$$S_1 = 2C, \tag{5.18}$$

$$S_2 = 2D, \tag{5.19}$$

$$S_3 = B \tag{5.20}$$

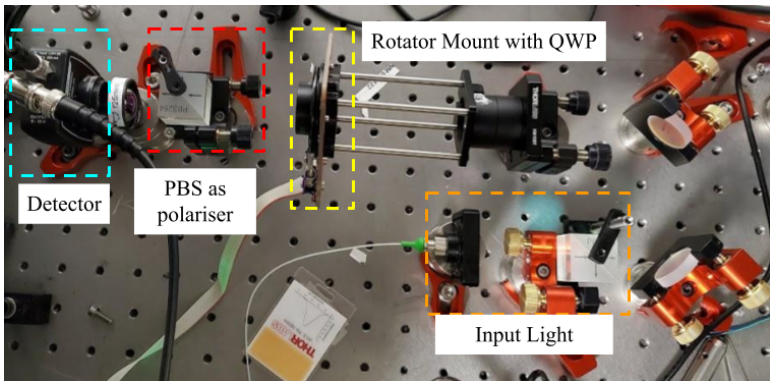

Figure 5.1: Initial design in which the piezoelectric resonant motor driven rotation mount was used. QWP in the figure stands for QWP. The figure shows the input light to be horizontally polarised light generated by passing a laser output through a polarising beam splitter.

5.2 Construction

5.2.1 Rotating the Quarter Waveplate

The method was first evaluated to be accurate by measuring the output intensity of horizontally linearly polarised light (HLP). HLP is produced by passing light from a 1550 nm laser through polarising beam splitter (PBS). A PBS has a reflectance and transmission arm, in which the reflectance arm allows S polarised components to pass through, while the transmission arm allows P polarised components to pass. S and P polarisation refers to the polarisation plane that is perpendicular and parallel to the plane of incidence respectively. Thus, light in the transmission arm will have a (normalised) Stokes vector of $(1, 1, 0, 0)$. The performance of the rotating quarter-waveplate method can be evaluated against this.

Automating QWP rotation was achieved by a rotation mount that is driven by a resonant piezoelectric motor, shown in 5.1. These are motors which can rotate to a specified angle in under one second with a repeatability of 873 µrad [32].

However, it was not possible to run the rotation over a well-defined interval such as from 0° to 180°. If we imagine the rotation corresponding to movements of the arms of a clock, the rotation mount would be a clock where it is only possible to call the arms to go to a certain number on the face, but impossible to govern how the arms go from its current position to

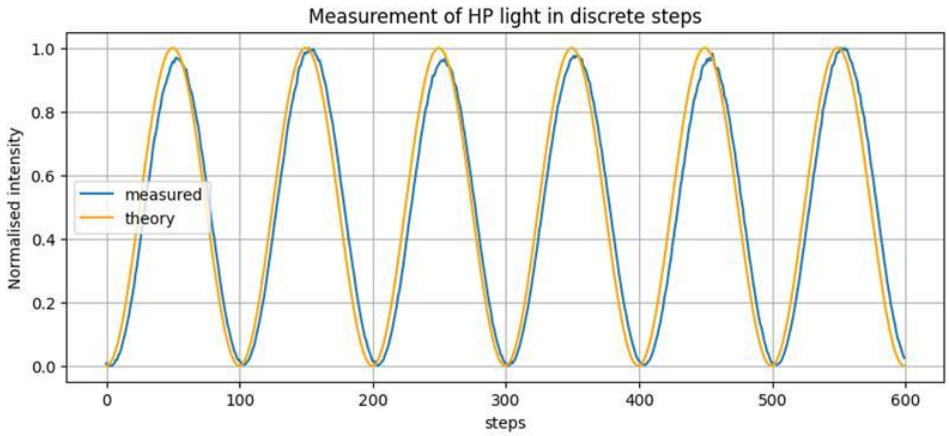

Figure 5.2: The phase mismatch and the peak mismatch is apparent in this measurement.

the specified number. For instance, while we wanted the mount to smoothly travel from its current position of 0° to 180°, going through $10°, 20°, 30°$... before finally arriving at 180°, what could happen is it travelling to 30°, pausing, and then rotating in the other direction to go to 180°. An example of an error is shown in figure 5.2.

Furthermore, the rotation mount was not supposed to be run continuously. Thus, when we run the motor continously, the rotation mount will occasionally not return to the initial position of 0°, which ruins subsequent measurements. It is paramount to have the QWP return to a position in which its fast axis is aligned horizontally.

Thus, the resonant motor rotation mount was replaced with a stepper motor. This allows the QWP to rotate 180° at a constant angular velocity. The Stokes parameters can be obtained by recording the intensity variation during the time that the waveplate is rotating.

A stepper motor is a brushless DC motor which rotates a single step with each pulse received. This allows the motor to have highly precise rotations without the need for a feedback or a 'memory' of its position. A full rotation can be achieved in 0.2 seconds by sending 200 on/off pulses with a delay of 1 ms between each pulse. This gives a fast and precise rotation which is suitable for its use in rotating the QWP. Driving it at a higher speed caused the motor to seize up occasionally or move in a sporadic manner. This might be because of the fact that the torque required to rotate at such a high speed

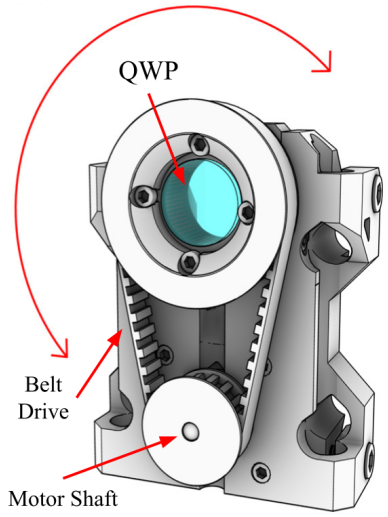

Figure 5.3: Diagram showing where the QWP, motor and mechanical belt drive are on the mount. Drawing taken from [33].

is too much for the motor.

The mount for driving a QWP with the stepper motor is composed of 3D printed parts taken from [33]. This is shown in figure 5.3. A QWP is attached to the rotating wheel of the CRM1/M caged rotation mount, which is then secured to the 3D printed stage. A 3D printed wheel is then affixed to the rotating wheel with four screws, to which a mechanical belt drive connects the 3D printed wheel with another wheel actuated by the motor shaft. This allows for the motor to actuate the QWP. When the motor is in its off state, the shaft is disengaged, allowing the QWP to be freely rotated to a desired starting orientation.

The stepper motor is controlled by an Arduino Uno, which is a microchip based on the ATmega328P microcontroller. It is a microcontroller which allows for users to write 'sketches' in C/C++, and upload them onto the Uno for execution. The device offers a way for users to quickly prototype hardware device which performs specific tasks. In our case, we used the Uno to receive a command through the serial port, which would then send pulses to the motor driver which drives the motor.

5.2.2 Intensity Meausurement

Initial measurement was performed by reading out the voltage recorded by a photodiode and displayed onto a picoscope, which is a 4 channel os-

cilloscope that connects directly to the computer for display and analysis. While it is possible to automate the recording and analysis process through using a library written by a fellow lab member and PhD student Markus Rademacher, there is considerable overhead involved in the process and it is not guaranteed that this method is fast enough for our purposes. Additionally, we wanted to use a device which can also serve as an output device, as this will become useful in later stages.

As a result, we opted to utilise a Red Pitaya, which is a four channel device with a FPGA programmable microcomputer [34]. Designed as a replacement for expensive laboratory equipment, it can function as an oscilloscope, signal generator, spectrum analyser and other instuments. It also provides an ethernet connection, which allow devices connected to the network to communicate with the device. We used the SCPI (Standard Commands and Programmable Instruments) protocol [35] to read the photodiode voltage into its buffer and then perform numerical calculations on the recorded readings.

It was possible to record the voltage accurately when the QWP was rotated by the piezoelectric rotator mount in discrete steps. The Red Pitaya fills its 16384 bits long buffer at a sampling frequency of 125MHz, and as there was a small delay after each step, this is enough time for the buffer to be filled entirely with the same voltage reading (with small fluctuations in noise). The average of the buffer would then give an accurate reading of the intensity when the QWP is at that orientation.

However, this method fails to work when the QWP is continuously rotated by the stepper motor. It was difficult to consistently window the same part of the buffer to get the intensity time trace. There is also a noise that the Red Pitaya picks up, in which after every few recorded points in the buffer, there would be a point at a value that is far from where the intensity should be. This causes the time trace to appear to have a 'noise floor' that is unphysical and destroys our ability to recover the Stokes parameters from the intensity trace.

Thus, the Arduino Uno is also used to measure the intensity. It measures the vootage as a function of the QWP angle. The Uno has 6 analogue pins, each of which can read a voltage from 0 to 5 volts and returns a number between 0 to 1024. Thus, we connected an analogue pin to the photodiode on the polarimeter. In the Arduino sketch, we added an additional command

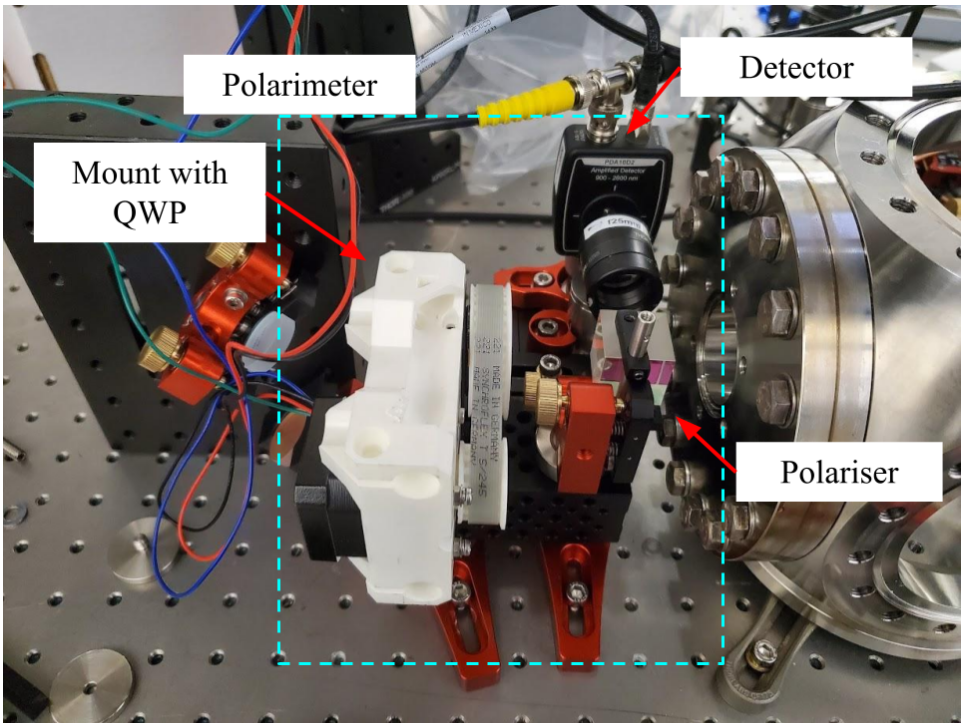

Figure 5.4: Polarimeter mounted on two blocks, which allows it to be slide in and out of a beam path and quickly measure the SOP of the beam. Beam propagation is from left to right.

after sending a pulse to the stepper motor to read the analogue pin. In our setup, the belt drive has a gear ratio of 2 to 1, so every full turn of the stepper motor at 200 steps causes the QWP to rotate 180 steps. This amount to reading the voltage at every $360°/200 = 1.8°$. With 200 points, this is also far more than the minimum 18 points (due to the Nyquist sampling theorem) required [31]. With the serial communication between the Uno and the computer, data can be sent back to the computer for calculating the Stokes parameters. The whole device is shown in 5.4.

5.2.3 Calculating the Stokes Parameter

The original program to calculate the Stokes parameters was simply converting the equation 5.16 into code. The stepper motor should rotate such that:

$$\theta_{j+1} - \theta_j = \omega(t_{j+1} - t_j), \tag{5.21}$$

which makes it easy to substitute the intensity values into equation 5.16.

However, there is also a requirement for the sinusoidal functions to have the same angular argument as the intensity distribution. That is to say, to obtain the coefficient B, it is required to multiply each of $I(\theta_j)$ with the corresponding $\sin 2\theta_j$ and sum across each value of j, where θ refers to the orientation of the QWP. As such, we must know the angle of the QWP that gives rise to the intensity reading and multiply it with the sine value of that particular angle. This turned out to be a non-trivial problem.

To achieve this, we first calibrated the QWP by passing horizontally light through it and rotating it until a maximum is measured. As a maximum would be recorded when the QWP has its fast axis aligned with the orientation of the linearly polarised light, this method allows us to ensure that the QWP has its fast axis aligned with the horizontal at $\theta = 0°$.

Usually, when the computer's serial port connects with the Arduino, it engages the stepper motor and advances the motor by a few degrees. Thus, calibration would be required to ensure the QWP is back in alignment. Furthermore, despite finding the intensity minimum, it could still be possible that the QWP remains one or two degrees away from having its fast axis aligned with the horizontal. We also notice that ultimately, the intensity is at a minimum regardless of whether the fast or the slow axis is horizontal. The intensity distribution is identical when measuring horizontally polarised light, as the distribution should be a cosine curve and the minimum appears every $\pi/4$ phase change. However, this will not be true for lights with other SOP, and we could be measuring with the wrong orientation the whole time. In general, the problem in our calculation is that instead of having the matching θ_j for both the intensity function $I(\theta)$ and the Fourier basis $\sin n\theta$ or $\cos n\theta$, this method of calibration makes it highly likely that we are taking the product $I(\theta_j + \phi)$ and $\sin(\theta_j)$ for example, which will give the wrong coefficients.

As such, we instead perform a fit whereby we fit with equation 5.12 the recorded intensity. Moreover, instead of just fitting the coefficients A, B, C and D, we fit an additional phase term ϕ that takes into account that the QWP may not begin rotating with the fast axis horizontal, but at a point where the fast axis is ϕ radians away from the horizontal. Thus, the function

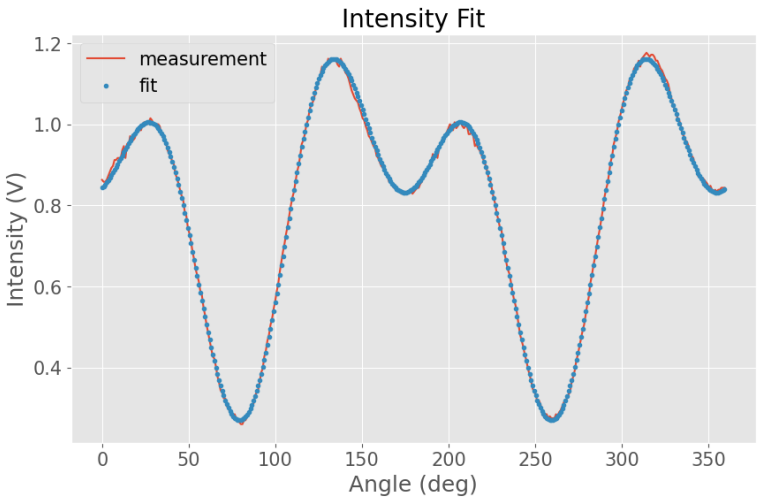

Figure 5.5: Comparing between the recorded intensity and the fit generated by the predicted coefficients.

we fit is:

$$I(\theta_j) = A - B\sin 2(\theta_j + \phi) + C\cos 4(\theta_j + \phi) + D\sin 4(\theta_j + \phi) \qquad (5.22)$$

Thus, we fit 5 parameters to 200 data points. This approach has shown to be quite accurate with low uncertainties and guarantees the reliability of the polarimeter. An example of a fit is shown in figure 5.22.

Chapter 6

Electro-Optical Modulator

6.1 EOSpace Electro-Optical Modulating Polarisation Controller

To perform polarisation change at high speeds, we are using an EOSpace in-line, 3-stage lithium niobate polarisation controller. This is an electro-optical modulator in which voltages can be applied to each of its 24 pins. Light is coupled into the device by single mode fibres that go in and leave the device. Varying the voltages applied onto each of the pins will change the refractive index of the crystals inside the device. This causes the crystals' birefringence to change. Thus, the device operates as a retarder with a phase shift that varies with the applied voltage. This allows for a change in the input light's polarisation.

The advantage of using such a device are that it requires a comparatively lower voltage input and has a response time of under 100 ns since its operating voltage is plus or minus 70 volts [36]. The device requires a significantly lesser amount of voltage than other polarisation controllers like free-space pockel cell electro-optial modulators, which requires thousands of voltages to achieve switching between TE and TM polarisation mode.

Additionally, as we seek to rotate the particle at MHz frequencies, the device will also need to have a response time at a comparable frequency, and as such its low response time makes it perfectly suited for this project.

Calibration is required to determine the input voltages that correspond to generating a desired polarisation. The device is sensitive to temperature fluctuations and movements of the connecting fibre. As such, a protocol

has to be devised to determine how the device performs, and to determine the right voltages to input into the device to obtain polarisations that are desired to achieve librational control.

6.2 Mapping

We use the polarimeter to obtain a map which shows the Stokes parameters obtained by applying a pair of voltages onto pin A and C of a stage in the polarimeter. The convention used here would refer to the voltage on pin A as $V1$ and that on pin B as $V2$. The device is used as a 3 stage device, so only 9 out of the 24 pins would be connected. The rest would be left floating. The specific pin layout is shown in table 6.1. As a result, a Stokes parameter $S = (S_1, S_2, S_3, S_4)$ can be written as a function of voltages V_1, V_2 applied, so $S = S(V_1, V_2)$. For a normalised Stokes parameter, $S_0 = 1$, so it is only required to create maps for $S_1(V_1, V_2)$, $S_2(V_1, V_2)$ and $S_3(V_1, V_2)$.

Most of our measurements perform a grid search over 11×11 points for a simple calibration test, 21×21 for a fairly comprehensive check and a 41×41 check which is quite thorough and time-consuming. If such a map does not provide a fine enough detail, it is always possible to interpolate between the points so we can obtain a finer-grained map.

An example of this procedure is demonstrated in figure 6.1. Across a range of input voltages between -30 to 30 volts for each of the two pins, a scan across 21×21 uniformly spaced voltages points is taken, requiring around 20 minutes. The error for each predicted parameter used to calculate the Stokes parameters is shown in figure 6.3. It can be seen that the points which appear to be disjointed with the others on the Stokes parameters map coincide with those in which the predicted phase is radically different from most other points on the plane. These can be removed from the distribution, and the missing points can be filled in by interpolation. It is assumed that if the phase is wildly different than its neighbours, the fitting function has made a mistake in its predictions. The result of the interpolation is shown in 6.4.

We can verify that the intensity is constant across different voltage inputs by measuring S_0, as shown in figure 6.2. This should be the case as the device's attenuation should not be a function of the input voltages. Furthermore, as trap frequencies vary as a function of trapping power, if the device

Pin	3-stage device
1	**1A**
2	**1B**
3	**1C**
4	NC
5	NC
6	NC
7	NC
8	NC
9	NC
10	**2A**
11	**2B**
12	**2C**
13	NC
14	NC
15	NC
16	**3A**
17	**3B**
18	**3C**
19	NC
20	NC
21	NC
22	NC
23	NC
24	NC

Table 6.1: Layout for the pins for a 3-stage device. NC means no connection.

attenuation varies with different input voltages, it would be impossible to determine whether the frequencies shift as a result of polarisation change or of intensity change.

6.3 Harvesting

The voltage maps allow us to find the right voltages which give us the desired polarisation. In the most basic case, we would want to generate elliptically polarised light and linearly polarised light. In the case of the linearly polarised light, we would want $S_3 = 0$, and we simply find a list of voltages in which $S_3 = 0$ (up to an acceptable level of tolerance). We

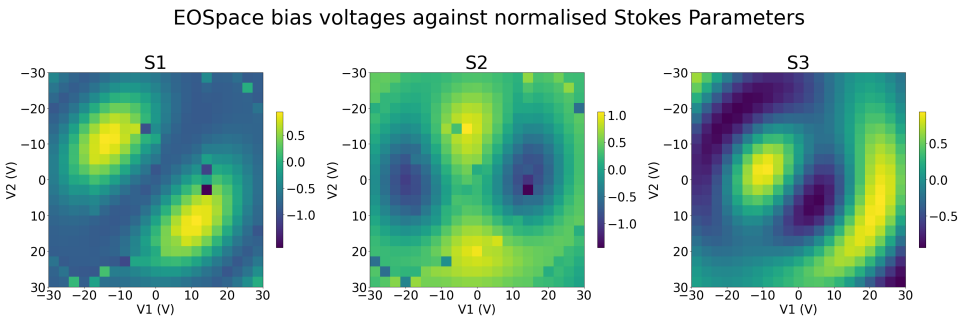

Figure 6.1: A voltage map with 21×21 points. Each voltage generates a Stokes vector. Upon normalising the parameters by dividing each with S_0, there will be 3 unique values S_1 (Left), S_2 (Middle) and S_3 (Right). One can picture each parameter as a function of V_1 and V_2, such as $S_1 = S_1(V_1, V_2)$.

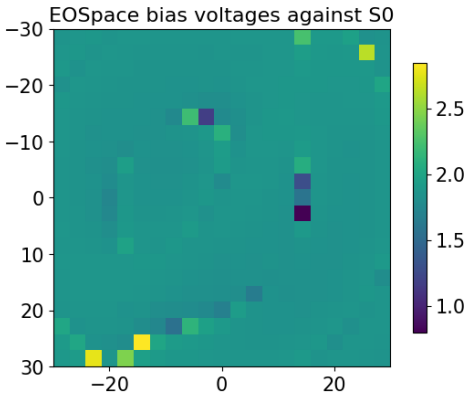

Figure 6.2: S_0 represents the intensity of light. The plot shows that this is mostly uniform across the plane, so regardless of the voltage input, the intensity is unchanged. The select few points which are different should be an error and this matches with the erroneous points shown in figure 6.3.

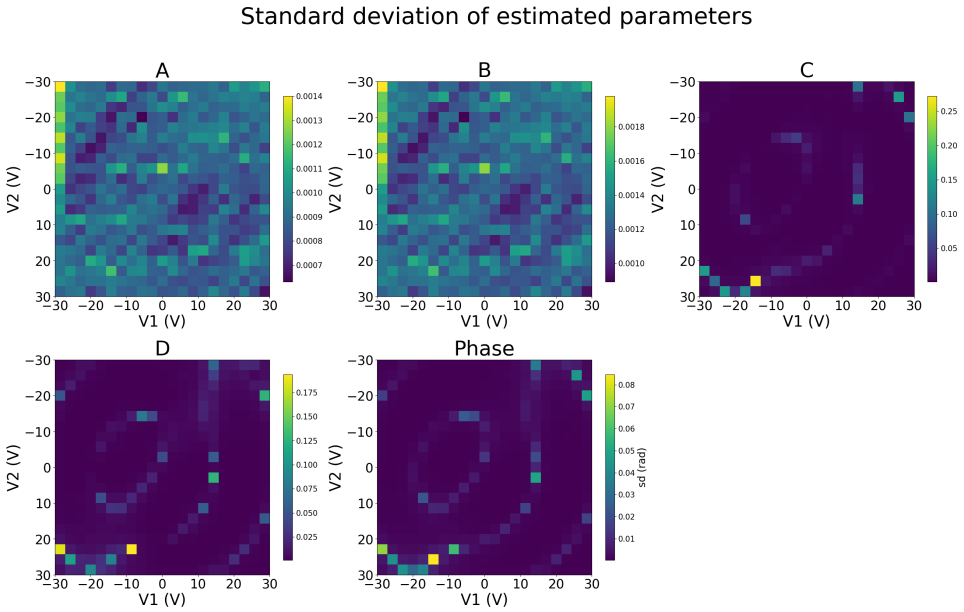

Figure 6.3: Error associated with each of the predicted parameters for each of the voltages tested, obtained as the standard deviation of the predicted values. This is obtained from the covariance matrix from the fit.

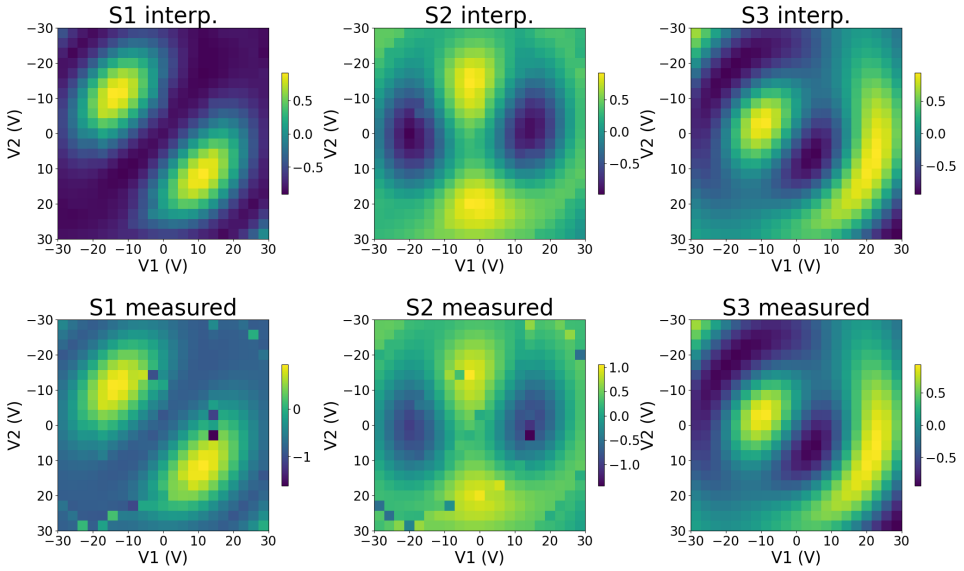

Figure 6.4: Interpolated data (Top), compared with the original data (Bottom). This is performed to ensure that the Stokes parameters vary smoothly as a result of changing input voltages.

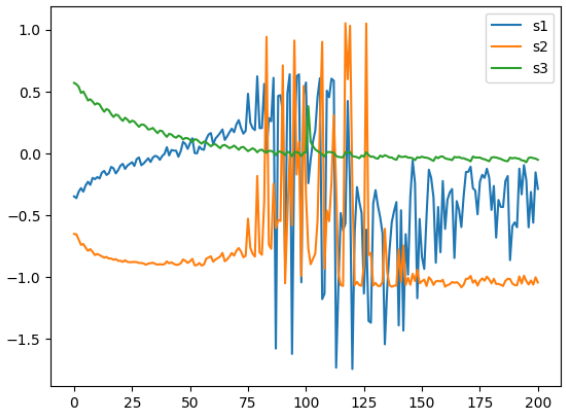

Figure 6.5: Fluctuations in the Stokes parameters over the course of the night, when the device is left uncovered (by foam for example). The horizontal axis refers to the n^{th} measurement, with each measurement spaced out over an equal interval of around 10 minutes. The vertical axis is for the value of S_1, S_2, S_3. It can be seen that there is both a periodic fluctuation and large fluctuations, possibly due to the air conditioning cycles and larger temperature fluctuations respectively.

find the S_1 and S_2 values which correspond to the voltages generating the $S_3 = 0$ points, which gives a list of known voltages to polarisation. This can similarly be done for finding circularly polarised light, except that the criteria would be $S_3 = \pm 1$ instead.

As one of the goals of the project is to generate linearly polarised light with an orientation that ranges across 180°, this can easily be done by finding all the points in which $S_3 = 0$, then sorting the points by their angle, which is calculated by taking the arc-tangent of S_1 and S_2.

6.3.1 Stability

A reason for needing to do the calibration is that environmental conditions affect how the polarisation controller operates. When the controller is placed on the optical table, it is noticed that fluctuations in the temperature will affect the output voltage. A measurement of the Stokes parameters of light generated with the same input voltages was done around every 10 minutes over the course of one night, and is shown in figure 6.5.

Thus, a large piece of foam was subsequently placed over the polarisation controller. Periodic measurements of the output SOP given the same

voltages over hours were recorded, and they show significantly lesser fluctuations than before. This is attributed to the foam blocking air drafts over the controller, which is a result of the air conditioning unit in the laboratory. This is seen in figure 6.6 Calibration is only required as we may move the device and stretch the fibres when modifying the experimental setup.

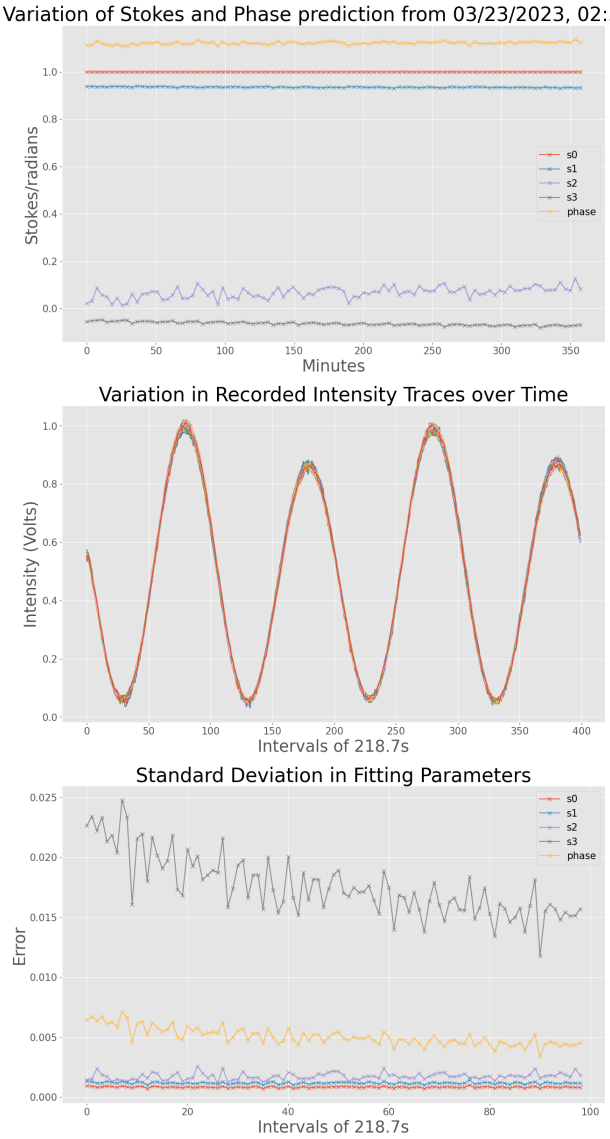

Figure 6.6: Different measurements over a long period of time taken after placing the foam over the polarisation controller. The top row are Stokes parameters' fluctuations, the middle row are the individual intensity time traces superimposed and the bottom row are the standard deviation of the predicted parameters.

Chapter 7

Trapping of Levitated Nanoparticles

7.1 Setup

The setup is shown in figure 7.1. The 1550 nm laser is coupled into a laser amplifier, which takes the 3.7 mW output and is able to amplify it to up to 2.2 W. The output is then fibre coupled into a circulator, in which the output of the circulator is coupled to the EOSpace polarisation controller. The polarisation controller has a maximum working power of 1W. The amplifier is outputting light at 1.999 W, which after the circulator's attenuation, should be below the polarisation controller's working power. The output fibre of the EOM is attached to a collimater, which increases the cross-sectional area of the beam. The light is then redirected by two mirrors, which allows for finer control over the beam's alignment. The beam is then taken through a periscope, created by mounting two mirrors facing each other at 45° on a block. This raises the beam such that it is on the same height as the window on the vacuum chamber. Between the periscope and the chamber also lies a small space which allows us to slide our polarimeter in so we can perform calibration for the EOM.

The vacuum chamber contains a rail in which at the centre, three plates are mounted with their faces perpendicular to the direction of propagation. This is shown on figure 7.2. At the centre of the front and back plate contains a hole in which high numerical aperture (NA) lenses can be mounted. The front lens is a 0.77 NA which focuses the collimated beam. This creates

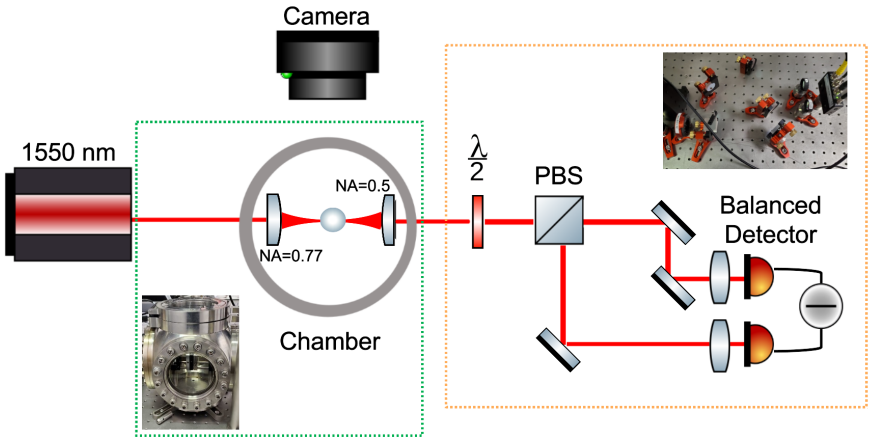

Figure 7.1: Diagram of the setup.

the Gaussian trapping potential which levitates nanoparticles. The beam diverges as it passes the focal point of the front lens. Thus, a 0.5 NA collection lens is required to re-collimate the beam. This allows the beam to exit the chamber with a constant cross-sectional area.

Between the front lens and the collection lens also sits another metallic plate that acts as a housing. When a nanoparticle is trapped, pressure in the chamber is lowered. Without the housing, the evacuating gas will likely knock the particle away from the trapping potential, causing us to lose the particle. As a result, the housing gives the nanoparticle protection and helps us with maintaining its position in the trap while the pressure reduces. There is also a hole on top of the plate, which allows for a camera that is placed on top of the chamber to 'see' what is inside the trap. As the particle scatters light from the trapping beam, the light can exit through the hole. A Chameleon CMLN 13S2N camera attached to a telescopic lens allows us to see the particle on a computer.

Upon exiting the chamber beam is taken down through another periscope. The beam then passes through a neutral density filter which reduces its power. This is necessary as the beam power might saturate or even damage the photodiodes. After passing through the filter, the beam passes into a balanced detector.

The balanced detection scheme, seen in figure 7.3, consists of a rotatable half waveplate, then a polarising beam splitter which splits the beam into a transmission and reflection path. The transmission path then has

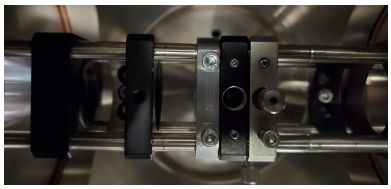

Figure 7.2: Top view of the setup inside the chamber, with 3 plates mounted on rails. On the 3 plates in the centre of the picture, the left one holds the front lens and the right one holds the collection lens. The particle would be levitated in the gap in the hold of the middle plate, between the left and right plate. This can be seen by the camera through the hole on the top side of the middle plate. The laser propagates from left to right.

two mirrors that guide the beam onto a 25 mm focal length aspheric lens that focuses the light onto the positive channel of the balanced detector. The reflectance path has one mirror that guides the beam to the negative channel of the detector. The detector is then able to compute the difference between the two channels, making it a very sensitive detection that can detect small changes in intensities due to changes in polarisation. This gives us a way to detect the librational motion of the nanoparticle, as a particle's rotation changes the polarisation of the scattered light, which causes the intensity transmitted and reflected by the PBS to change. By using the half waveplate to 'tune' the beam until both photodiode channels are at the same intensity, the difference can be amplified and detected. By taking the Fourier transform of the time trace, we can identify the rotational frequency modes of the particle.

7.2 Results

7.2.1 Trap Evaluation

The trap alignment was first tested by using only the amplified laser to trap. The testing determined that the beam was aligned well enough to trap but the initial lid that was used had a leak. The camera was also quite difficult to use as it was not particularly sensitive, and the focus was often difficult to obtain. This can be shown in figure 7.4, which is a picture of the same particle with different camera settings.

The power spectral density obtained for this trapped particle is shown in figure 7.5.

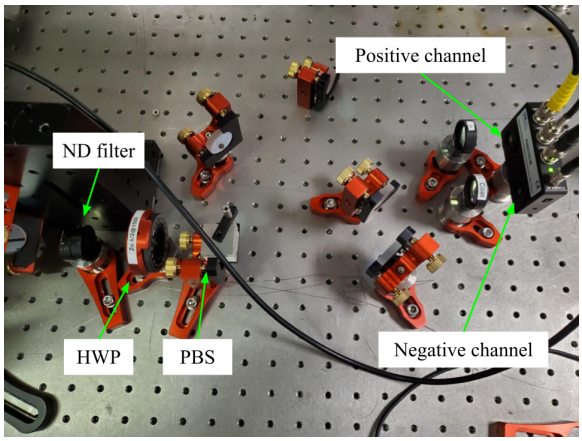

Figure 7.3: Balanced detection setup. HWP is the half waveplate, PBS is the polarising beam splitter, and ND filter is the neutral density filter.

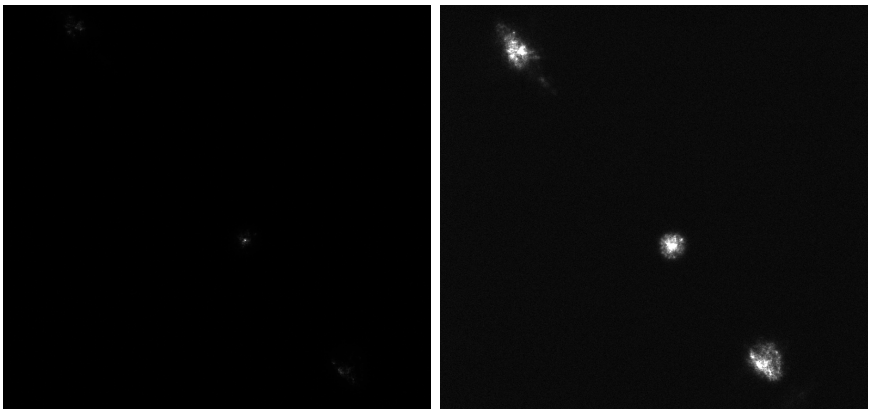

Figure 7.4: Adjusting the aperture and focus. The top and bottom spots are optics and the middle spot is the particle. We know as the spot is only present when a particle is trapped, and disappears when the beam is momentarily blocked (the particle will be lost even when the trapping potential is absent for a very small amount of time). The particle is well-focused but barely visible (Left) and defocused but bright (Right).

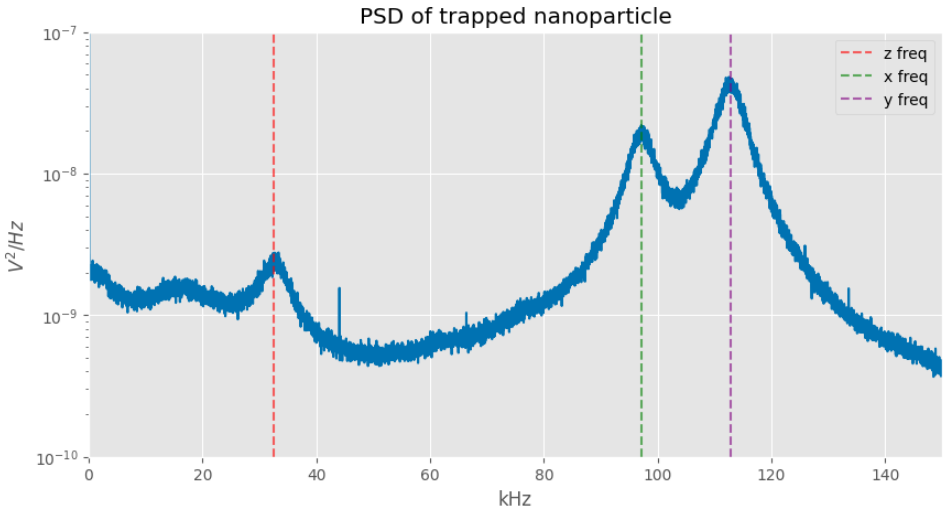

Figure 7.5: Power spectral density of a trapped particle displaying the x, y, z frequencies.

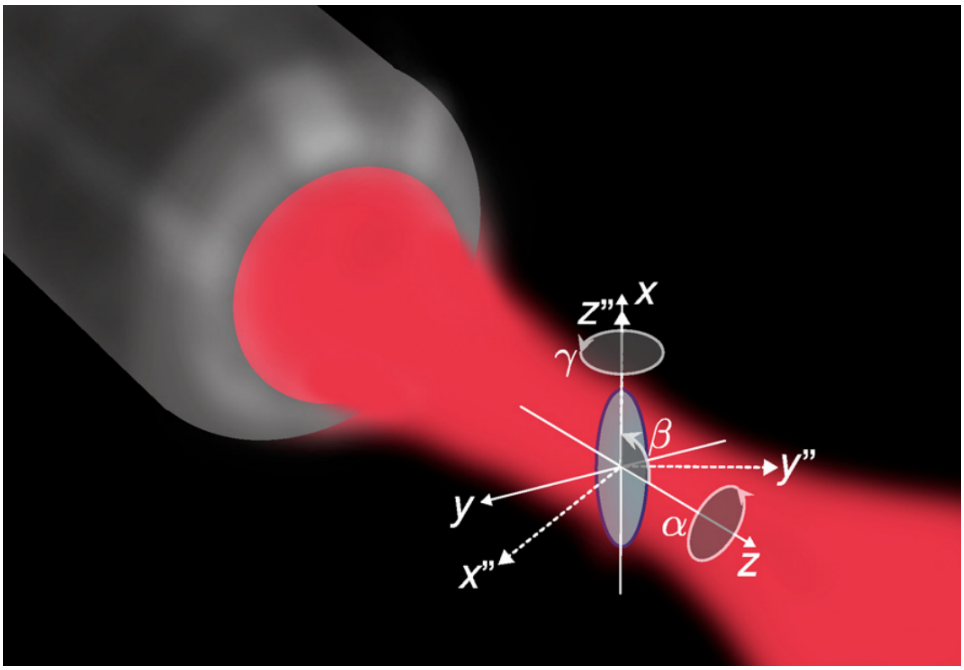

Figure 7.6: Diagram showing the translational axes of a molecule. When a particle is trapped its oscillation can be projected onto the 3 orthogonal Cartesian axes. Diagram taken from [37]

The polarisation controller was then connected to the laser to demonstrate trapping and rotational control. The polarisation controller introduces a loss of around 4 dB. Along with the fact that the polarisation controller has a maximum tolerable input power of 1 W, the resulting trapping field is much weaker. The lower intensity makes it more difficult to trap a particle.

Another problem was that it was not possible to bring the chamber's pressure to lower than 1 mBar without losing the particle. This is currently attributed to the trap misalignment. Before connecting the polarisation controller into the setup, it was possible to bring the trap to a lower pressure as the laser power was high, making the trap strong enough to keep the particle trap. However, with a weaker trap, alignment becomes important in ensuring the strength of the trap. Other lab members' experience was that it was possible to significantly increase trap frequencies by optimising the trap alignment. However, while some alignment was performed to increase the trap strength, there was insufficient time to do it to the extent at which the trapping strength increased to an extent which allows us to keep the particle at a much lower pressure. This hypothesis is strengthened by noticing that the trapping frequencies in the new setup are quite low. If these are indeed translational frequencies and not librational frequencies, this would imply that the trap is much weaker than before.

7.2.2 Polarisation Induced Frequency Shifts

Shifting in trap frequencies induced by a change in the trapping field's polarisation is observed. Figure 7.7 represents an experiment in which the power spectrum is measured for different input voltages. Prior characterisation has obtained the voltage pairs $(1, -1)$ V to generate approximately circularly polarised light and $(-0.8, 0.7)$ to generate linearly polarised light. The pair $(1, -0.9)$ generates elliptically polarised light. This is summarised in table 7.1. Each of these different polarisations results in a power spectrums in which the peaks shift and split. This is strong evidence for achieving rotational control of the particle via varying linearly or circularly polarised light.

There are strong reason to attribute the visible peaks to be translational peaks. Referring to figure 7.8, the peaks do not shift as a result of a change in pressure. If the peaks represent rotational frequencies, they should shift

V1	V2	Polarisation
1	-1	Circularly Polarised
-0.8	0.7	Linearly Polarised
1	-0.9	Elliptically Polarised

Table 7.1: Table demonstrating the different voltages and resulting polarisation.

due to a change in pressure, as a change in pressure is a change in the damping due to gas collisions, which affects the rotational frequency.

Figure 7.8 shows the PSDs of the same particle subjected to linearly polarised light and circularly polarised light. When the particle is trapped with linearly polarised light, two peaks that are close to each other are seen. These are attributed to the x and y oscillatory frequencies. The x and y oscillations have distinct frequencies as linear polarisation has an orientation. As a result the potential gradient has different strengths in the x and y directions, resulting in different oscillatory frequencies along the two axes.

On the other hand, when the particle is trapped with circularly polarised light, the particle is brought to spin along its central axis. Thus, its x and y (projected) position change at the same rate. This forms its rotational frequency, and it is not surprising that the two peaks will merge into a single peak.

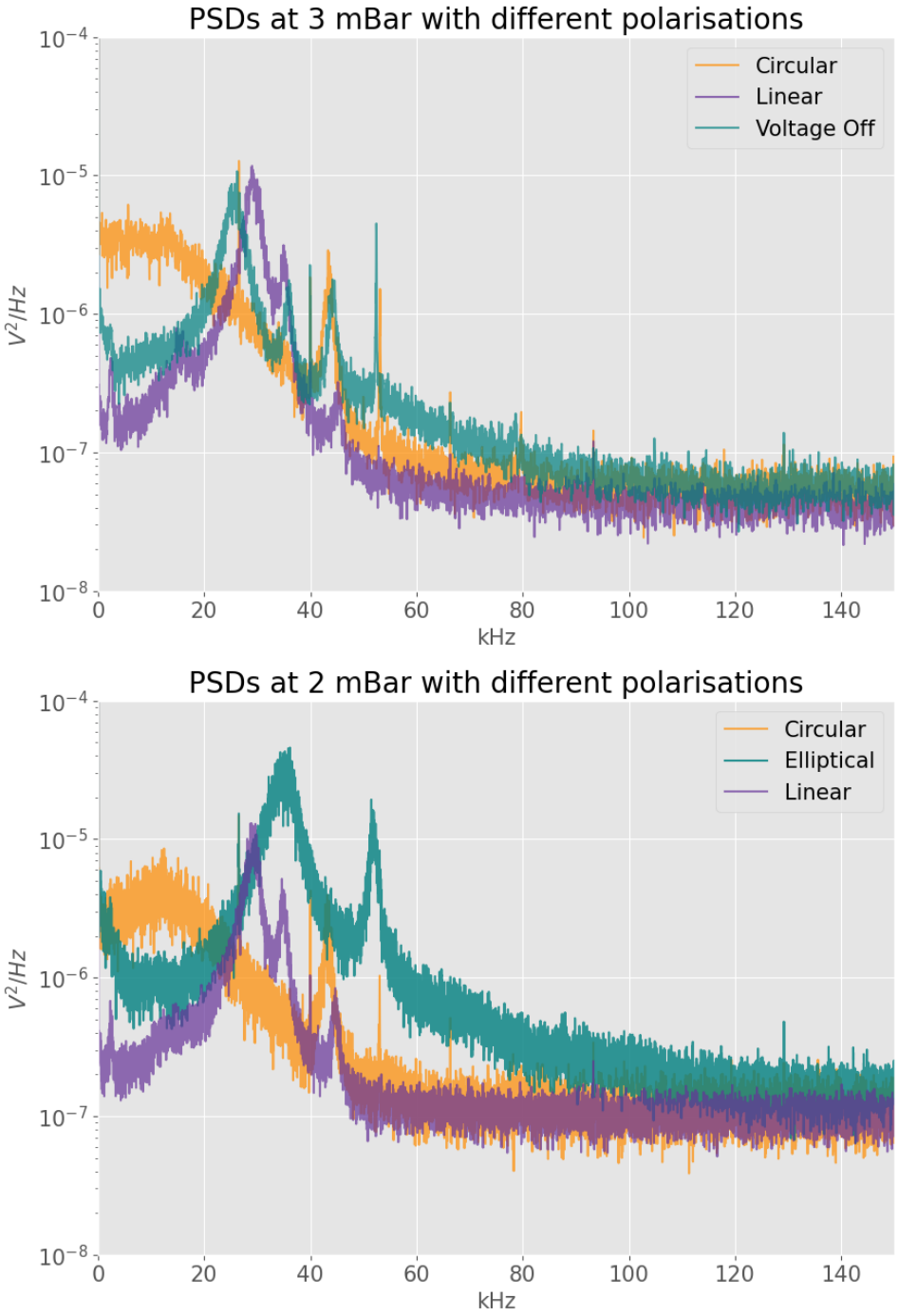

Figure 7.7: PSDs of the same particle being taken at different pressures. Plot on top represents the PSDs when particle is trapped with linearly polarised light, elliptically polarised light and circularly polarised light. Plot on bottom is the same but at a pressure of 2 mBar. "Voltage off" refers to the state when there is no voltage on the polarisation controller, and the light is mostly linearly polarised with slight ellipticity.

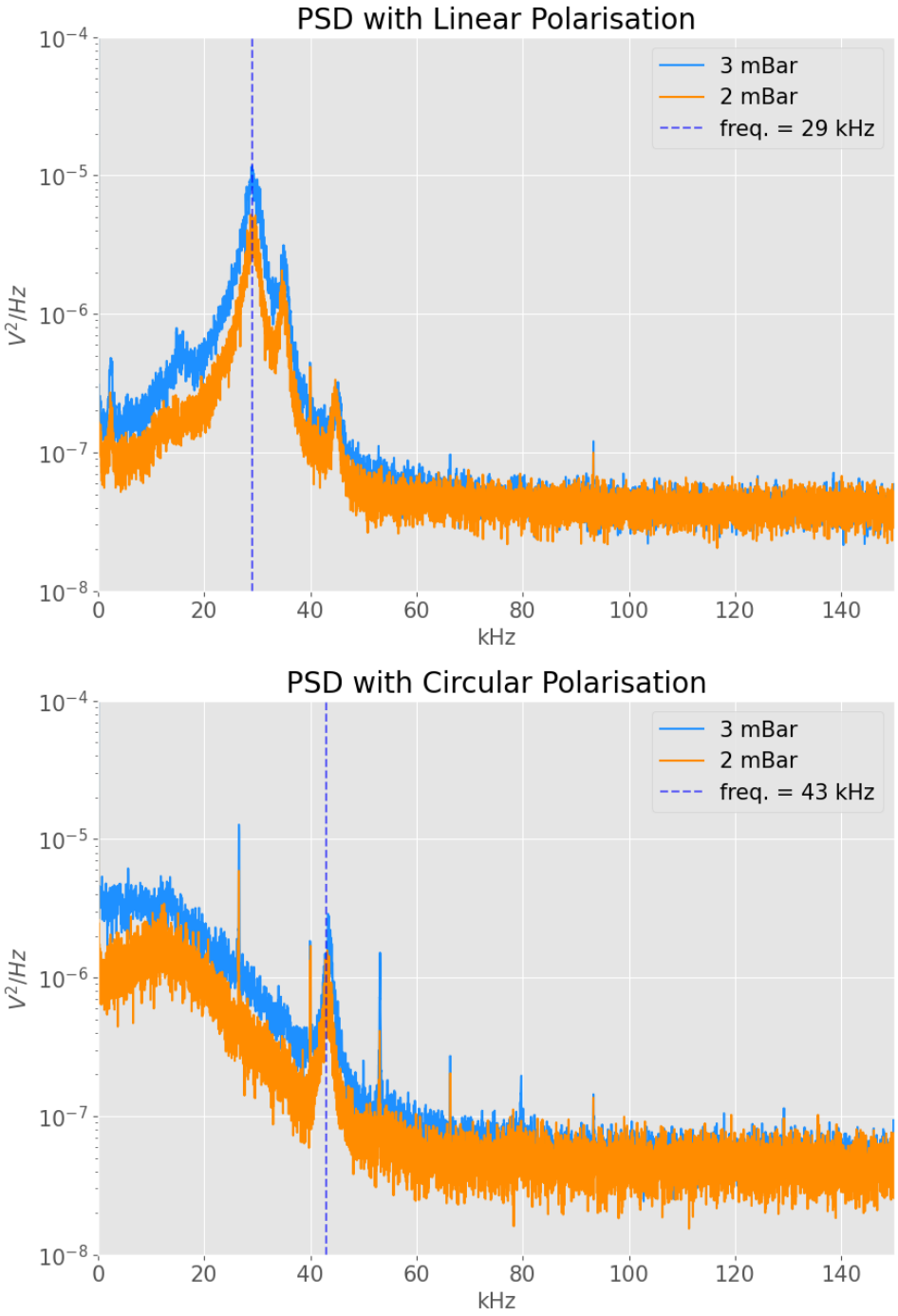

Figure 7.8: Comparing PSDs for same input voltages at different pressures. All plots are taken with the same trapped particle. The top represents when the particle is trapped with linearly polarised light, whereas the bottom represents the particle being trapped with circularly polarised light. The two PSDs in each plot represent the measurement being taken at 3 mBar and 2 mBar respectively.

Chapter 8

Conclusion

The project has been successful in achieving its main goal in the demonstration of fast rotational control of levitated nanoparticles. The pursuit of this objective has led to the construction of new devices and the development of techniques which will undoubtedly be invaluable in future research.

The free space polarimeter was constructed for the purpose of characterising the EOSpace electro-optical polarisation controller. Its property of being compact and robust, along with its fast measurement speed would mean it can be used for other optomechanical experiments. For example, some experiments involve investigating the anisotropic properties of different organic particles, and having the ability to measure the polarisation of scattered light exiting the chamber would give more information about the particle in these experiments.

Gaining the ability to use the polarisation controller may offer new ways to generate polarisation to perform trapping. Prior experiments conducted in the group, such as the recent, groundbreaking work in achieving six degrees of cooling of a levitated nanoparticle, have used a polarising beam splitter, QWP and half waveplate to generate a desired elliptically polarised light to perform the cooling [20]. Perhaps this polarisation controller, which can generate light of arbitrary polarisation, can offer new ways to achieve librational control.

Finally, polarisation change was directly used on a trapped nanoparticle. This opens up a new avenue to generate quantum states, and it is within reach to use this technique to generate rotational Fock states. An immediate application of this would simply be demonstrating the 'locking' of a particle

by periodically flipping between applying linearly polarised light and circularly polarised light, and demonstrating that this can help lock the particle in a desired frequency [38]. While it is already possible to generate linearly polarised light with its orientation angle varying quickly, the effect on the levitated particle is not conclusive, and as such more work has to be done to ensure that the observation is what is expected.

Creating quantum states is certainly within reach. However, more time would be required to improve the current trap and to determine the proper readout method. A possible solution is to use a different beam for readout and determining the particle's rotational motion. Only time will tell as to whether this will be the solution.

Appendix A

Classical Method of Polarisation Measurement

The classical method requires a vertical polariser, a horizontal polariser, a 45° polariser and a quarter waveplate [31][22]. Assuming a laser propagating into an intensity detector, various combinations of the above optical elements will be placed in the beam path and the resultant intensity will be recorded. Each individual Stokes parameter can then be calculated from the addition and subtraction of the different intensity values.

A notable drawback of this approach is that it requires manually removing and placing different optical elements into a beam path and then doing a measurement. While automation is possible it requires a complex mechanism that can slide different optical elements into and out of the beam path. Additionally, each optical element has a different amount of absorption, which affects the recorded intensity, thus introducing inaccuracies in the parameters. Finally, it only uses four recorded intensities, thus making it susceptible to errors.

An alternative approach, which is able to eliminate the first drawback is by using multiple beam splitters and detectors so the beam is split into four paths with the optical elements already in place. This allows for all four intensities to be measured at the same time. However, other drawbacks remain present, and such an approach requires precise alignment and the acquisition of multiple optical elements, which are not feasible within the scope of this project. Thus, an alternative approach has to be considered.

Appendix B

Intensity Measurement for Polarimeter

Initial measurement was performed by reading out the voltage recorded by a photodiode and displayed onto a picoscope, which is a 4 channel oscilloscope that connects directly to the computer for display and analysis. While it is possible to automate the recording and analysis process through using a library written by a fellow lab member and PhD student Markus Rademacher, there is considerable overhead involved in the process and it is not guaranteed that this method is fast enough for our purposes. Additionally, we wanted to use a device which can also serve as an output device, as this will become useful in later stages.

As a result, we opted to utilise a Red Pitaya, which is a four channel device with a FPGA programmable microcomputer [cite]. Designed as a replacement for expensive laboratory equipment, it can function as an oscilloscope, signal generator, spectrum analyser and other instuments. It also provides an ethernet connection, which allow devices connected to the network to communicate with the device. Most importantly, with the SCPI (Standard Commands and Programmable Instruments) protocol, Python code can be written to talk and listen to the device. Upon connecting the photodiode output to one of the Red Pitaya's two input ports, we can programmatically ask the Red Pitaya to read the photodiode voltage into its buffer and then perform numerical calculations on the recorded readings.

It was possible to record the voltage accurately when the QWP was rotated by the Thorlabs piezoelectric rotator in discrete steps. The Red

Pitaya fills its 16384 bits long buffer at a sampling frequency of 125MHz, and as there was a small delay after each step, this is enough time for the buffer to be filled with the same voltage reading with small fluctuations in noise. The average of the buffer would then give an accurate reading of the intensity when the QWP is at that orientation.

However, this method fails to work when the QWP is continuously rotated by the stepper motor. In this case, the objective is to measure the intensity variation over time. As the stepper motor rotates at a constant velocity, the time trace should be equivalent to if we measured the intensity at constant incremental change of the angle. We attempted to record the time trace in Red Pitaya's buffer as the stepper motor rotated. However, we encountered a lot of difficulty in consistently windowing the same part of the buffer to get the exact intensity variation at each measurement. Then, despite solving that problem, there is also a noise that the Red Pitaya picks up, in which after every few points in the buffer which records the reading from the photodiode, there would be a point at a value that is far from where the intensity should be. This causes the time trace to appear to have a 'noise floor' that is completely unphysical and destroys our ability to recover the Stokes parameters from the intensity trace.

Appendix C

Code for Polarimeter Control

Shown here are the code used for controlling the polarimeter. The stepper class uses SCPI commands to directly communicate with the Red Pitaya, whereas the calibration class is a higher level (more or less) API that contains helper functions to do more complex tasks like automating the mapping process.

C.1 Stepper Class

```
import serial
from time import sleep, time
import numpy as np
from scipy.optimize import curve_fit

# redpitaya imports
import redpitaya_scpi as scpi
# timing imports
from tqdm import tqdm
import datetime

class EOM_QWP:
    """Class to control the EOSpace and measurement"""

    def __init__(self, rp_url='128.40.2.122', port='/devttyACM0
    ', darknoise=0.2076):
        ### serial set up
        self.ser = serial.Serial('/dev/ttyACM0')
        sleep(2)

```

```
20          ### redpitaya setup
21          self.rp_s = scpi.scpi('128.40.2.122')
22          self.rp_s.tx_txt('GEN:RST')
23          # set output func to DC
24          self.rp_s.tx_txt('SOUR1:FUNC DC')
25          self.rp_s.tx_txt('SOUR2:FUNC DC')
26          self.rp_s.tx_txt('SOUR1:VOLT 0')
27          self.rp_s.tx_txt('SOUR2:VOLT 0')
28          self.darknoise = darknoise
29
30 ############ METHODS DEFINITIONS #############
31
32
33      def read(self):
34          self.ser.write(b'r\n')
35          a = []
36          for i in range(400):
37              try:
38                  x = float(self.ser.readline().strip(b'\r\n'))
39                  a.append(x)
40              except ValueError:
41                  pass
42          return a
43
44      def cont(self):
45          self.ser.write(b'cont\n')
46          s = input("Press any button to stop")
47          if len(s) > 0:
48              self.ser.write(b's\n')
49
50      def convert_to_volts(self, arr):
51          """ convert array to volts based on arduino specs"""
52          return 0.0049*np.array(arr) - self.darknoise
53
54      def intensity_func(self, theta, A,B,C,D,phase):
55          return 1/2*(A+B*np.sin(2*(theta+phase))+C*np.cos(4*(
    theta+phase))+D*np.sin(4*(theta+phase)))
56
57      def calculate_stokes(self, results):
58          N = len(results)
59          #thetas = np.linspace(0, np.pi, N)
60          #thetas = np.deg2rad(np.arange(0.9,180.9,0.9))
61          thetas = np.linspace(0, 2*np.pi, N, endpoint=False)
62          # when taking transmittance beam
```

```
63          # and fast axis horizontal
64          #A = (2/N)*np.sum(results)
65          #B = -(4/N)*np.dot(results,np.sin(2*thetas))
66          #C = (4/N)*np.dot(results,np.cos(4*thetas))
67          #D = (4/N)*np.dot(results,np.sin(4*thetas))
68
69          # when taking reflectance beam
70          # fast axis horizontal
71          # code for when polariser used is horizontal
72          popt, pcov = curve_fit(self.intensity_func, thetas,
    results,bounds=([-10,-10,-10,-10,0.8],[10,10,10,10,1.4]))
73          A,B,C,D,phase = popt
74          S0 = A-C
75          S1 = -2*C
76          S2 = -2*D
77          S3 = B
78          # intensity equation modified as we take the reflected
    beam
79          # in effect a vertical polariser (as reflected beam
    takes s pol)
80
81
82
83          stokes = np.array([S0,S1,S2,S3])
84          #nstokes = np.append(stokes/S0,phase)
85          nstokes = np.append(stokes,phase)
86          nstokes = np.concatenate([nstokes, [A,B,C,D]])
87          timestamp = time()
88          nstokes = np.append(nstokes, timestamp)
89          return nstokes, np.sqrt(np.diag(pcov))
90
91      def measurement_run(self):
92          self.ser.write(b'f\n')
93          sleep(0.4)
94          a = self.read()
95          v = self.convert_to_volts(a)
96          nst, st = self.calculate_stokes(v)
97          return nst, st, v
98
99      def test_volts(self, v1, v2, t=3, off=True):
100         '''Run volts on the two channels for testing'''
101         if v1 < 0:
102             self.rp_s.tx_txt('SOUR1:FUNC DC_NEG')
103         else:
```

```
104             self.rp_s.tx_txt('SOUR1:FUNC DC')
105
106         if v2 < 0:
107             self.rp_s.tx_txt('SOUR2:FUNC DC_NEG')
108         else:
109             self.rp_s.tx_txt('SOUR2:FUNC DC')
110
111         c1 = "SOUR1:VOLT %.4f"%(abs(v1))
112         c2 = "SOUR2:VOLT %.4f"%(abs(v2))
113         self.rp_s.tx_txt(c1)
114         self.rp_s.tx_txt(c2)
115         self.rp_s.tx_txt('OUTPUT1:STATE ON')
116         self.rp_s.tx_txt('OUTPUT2:STATE ON')
117         if off == True:
118             sleep(t)
119             self.rp_s.tx_txt('OUTPUT1:STATE OFF')
120             self.rp_s.tx_txt('OUTPUT2:STATE OFF')
121         else:
122             s = input("Type to turn off...\n")
123             if len(s) > 0:
124                 self.rp_s.tx_txt('OUTPUT1:STATE OFF')
125                 self.rp_s.tx_txt('OUTPUT2:STATE OFF')
126
127
128
129     def measurement_run_volts(self, v1, v2):
130         '''v1 refers to pin 1 voltage
131           v2 refers to pin 2 voltage'''
132         if v1 < 0:
133             self.rp_s.tx_txt('SOUR1:FUNC DC_NEG')
134         else:
135             self.rp_s.tx_txt('SOUR1:FUNC DC')
136
137         if v2 < 0:
138             self.rp_s.tx_txt('SOUR2:FUNC DC_NEG')
139         else:
140             self.rp_s.tx_txt('SOUR2:FUNC DC')
141
142         c1 = "SOUR1:VOLT %.4f"%(abs(v1))
143         c2 = "SOUR2:VOLT %.4f"%(abs(v2))
144         self.rp_s.tx_txt(c1)
145         self.rp_s.tx_txt(c2)
146         self.rp_s.tx_txt('OUTPUT1:STATE ON')
147         self.rp_s.tx_txt('OUTPUT2:STATE ON')
```

```
148         sleep(0.2)
149         nst, st, v = self.measurement_run()
150         self.rp_s.tx_txt('OUTPUT1:STATE OFF')
151         self.rp_s.tx_txt('OUTPUT2:STATE OFF')
152         return nst, st, v
153 
154 
155     def turnoff_volt(self):
156         """Turn off redpitaya and voltages"""
157         self.rp_s.tx_txt('SOUR1:VOLT 0')
158         self.rp_s.tx_txt('SOUR2:VOLT 0')
159         self.rp_s.tx_txt('OUTPUT1:STATE OFF')
160         self.rp_s.tx_txt('OUTPUT2:STATE OFF')
161 
162     def v_a(self, alpha, sigma=1/2, vo=10, vpi=10, bias=0):
163         """Formula given in EOSpace specsheet"""
164         return 2*vo*sigma*np.sin(alpha) - vpi*sigma*np.cos(
    alpha)+bias
165 
166     def v_b(self, alpha, sigma=1/2, vo=10, vpi=10, bias=0):
167         """Formula given in EOSpace specsheet"""
168         return 2*vo*sigma*np.sin(alpha) + vpi*sigma*np.cos(
    alpha)+bias
169 
170     def EOM_voltages(self, n, va_bias=0, vb_bias=0, vo=10, vpi
    =10):
171         """Returns 2 arrays of voltages to put into pin 1/A and
     pin 2/B
172         which takes arguments n1, n2 referring to number of
    voltages."""
173 
174         nsts = []
175         va = lambda alpha: self.v_a(alpha, vo=vo, vpi=vpi, bias
    =va_bias)
176         vb = lambda alpha: self.v_b(alpha, vo=vo, vpi=vpi, bias
    =vb_bias)
177         volts_a = list(map(va, np.linspace(0, 2*np.pi, n)))
178         volts_b = list(map(vb, np.linspace(0, 2*np.pi, n)))
179 
180         # ensures voltage demanded from RP is not too big
181         volts_a = np.where(volts_a > 50, 0, volts_a)
182         volts_b = np.where(volts_b > 50, 0, volts_b)
183 
184         for i in range(n):
```

```
185             v1 = volts_a[i]/50
186             v2 = volts_b[i]/50
187 
188             nst, _, _ = self.measurement_run_volts(v1, v2)
189             nsts.append(nst)
190 
191         return nsts
192     def pirotation(self, seconds):
193         """Outputs linearly polarised light over 180 degree
    orientation change"""
194         with np.load('v1v2_coeffs_17mar.npz') as data:
195             p_v1 = data['p_v1']
196             p_v2 = data['p_v2']
197 
198         wave_form = 'arbitrary'
199         freq = 10000 # 10 kHz
200         ampl = 1
201         N = 16384
202         t = np.linspace(0,1,N)
203 
204         fit_v1 = np.poly1d(p_v1)
205         fit_v2 = np.poly1d(p_v2)
206 
207         v1 = 0.02*(fit_v1(t))
208         v2 = 0.02*(fit_v2(t))
209 
210         #v1 = np.sin(t) + 1/3*np.sin(3*t)
211         #v2 = 1/2*np.sin(t) + 1/4*np.sin(4*t)
212 
213 
214         waveform_ch_10 = []
215         waveform_ch_20 = []
216 
217         for v in v1:
218             waveform_ch_10.append(f"{v:.5f}")
219         waveform_ch_1 = ", ".join(map(str, waveform_ch_10))
220 
221         for v in v2:
222             waveform_ch_20.append(f"{v:.5f}")
223         waveform_ch_2 = ", ".join(map(str, waveform_ch_20))
224 
225         #self.rp_s.tx_txt('GEN:RST')
226 
```

```
227         self.rp_s.tx_txt('SOUR1:FUNC ' + str(wave_form).upper()
    )
228         self.rp_s.tx_txt('SOUR2:FUNC ' + str(wave_form).upper()
    )
229 
230         self.rp_s.tx_txt('SOUR1:TRAC:DATA:DATA ' +
    waveform_ch_1)
231         self.rp_s.tx_txt('SOUR2:TRAC:DATA:DATA ' +
    waveform_ch_2)
232 
233         self.rp_s.tx_txt('SOUR1:FREQ:FIX ' + str(freq))
234         self.rp_s.tx_txt('SOUR2:FREQ:FIX ' + str(freq))
235 
236         self.rp_s.tx_txt('SOUR1:VOLT ' + str(ampl))
237         self.rp_s.tx_txt('SOUR2:VOLT ' + str(ampl))
238         self.rp_s.tx_txt('OUTPUT1:STATE ON')
239         self.rp_s.tx_txt('OUTPUT2:STATE ON')
240         self.rp_s.tx_txt('SOUR:TRIG:INT')
241 
242         sleep(seconds)
243 
244         self.rp_s.tx_txt('OUTPUT1:STATE OFF')
245         self.rp_s.tx_txt('OUTPUT2:STATE OFF')
246 
247     def unpolarisedlight(self, hlpv, vlpv, t, gapswitch=1e-6):
248         '''Create unpolarised light'''
249         for i in range(t//gapswitch):
250             self.test_vlpv(hlpv[0], hlpv[1])
251             delay(gapswitch)
252             self.test_vlpv(vlpv[0], vlpv[1])
253             delay(gapswitch)
254 
255     def voltflips(self, start_v, flip_v, t1, t2):
256         '''Run voltages for t1 then switch to another for t2
    time.'''
257         self.test_volts(start_v[0], start_v[1], t=t1, off=False
    )
258         self.test_volts(flip_v[0], flip_v[1], t=t2, off=False)
259         self.test_volts(start_v[0], start_v[1])
260         s = input("Type something to turn off EOSpace...\n")
261         if len(s) > 0:
262             self.turnoff_volt()
263 
264     def upvoltflpis(self, hlpv, vlpv, flip_v, t1, t2):
```

```
265         '''Same as voltflips except we do unpolarised light as
    default'''
266         self.unpolarisedlight(hlpv, vlpv, t1)
267         self.test_volts(flip_v[0], flip_v[1], t=t2, off=False)
268         self.unpolarisedlight(hlpv, vlpv, t1)
269         self.turnoff_volt()
270 
271     def squarewaveflip(self, hlpv, vlpv, freq):
272         '''Outputs square wave which flips between horizontally
     and
273         vertically polarised light (for example) at freq and
    over time t.'''
274         hlpv = np.array(hlpv)/50
275         vlpv = np.array(vlpv)/50
276         ampl1 = abs(hlpv[0] - vlpv[0])/2
277         #ampl1 /= 50
278         ampl2 = abs(hlpv[1] - vlpv[1])/2
279         #ampl2 /= 50
280         offset1 = (hlpv[0]+vlpv[0])/2
281         #offset1 /= 50
282         offset2 = (hlpv[1]+vlpv[1])/2
283         #offset2 /= 50
284 
285         self.rp_s.tx_txt('GEN:RST')
286 
287         self.rp_s.tx_txt('SOUR1:FUNC SQUARE')
288         self.rp_s.tx_txt('SOUR2:FUNC SQUARE')
289         self.rp_s.tx_txt(f'SOUR1:FREQ:FIX {freq}')
290         self.rp_s.tx_txt(f'SOUR2:FREQ:FIX {freq}')
291         self.rp_s.tx_txt(f'SOUR1:VOLT {ampl1:.6f}')
292         self.rp_s.tx_txt(f'SOUR2:VOLT {ampl2:.6f}')
293         self.rp_s.tx_txt(f'SOUR1:VOLT:OFFS {offset1:.6f}')
294         self.rp_s.tx_txt(f'SOUR2:VOLT:OFFS {offset2:.6f}')
295 
296         # Enable output
297         self.rp_s.tx_txt('OUTPUT1:STATE ON')
298         #self.rp_s.tx_txt('SOUR1:TRIG:INT')
299         self.rp_s.tx_txt('OUTPUT2:STATE ON')
300         #self.rp_s.tx_txt('SOUR2:TRIG:INT')
301         #self.rp_s.tx_txt('OUTPUT:STATE ON')
302         self.rp_s.tx_txt('SOUR:TRIG:INT')
303         s = input("Enter something to turn off... \n")
304         if len(s)>0:
305             self.turnoff_volt()
```

C.2 Calibration Class

```
from stepper_class import EOM_QWP
from time import time, sleep
import numpy as np
from datetime import timedelta, datetime
from scipy.optimize import curve_fit
from tqdm import tqdm

class EOM_Calibrator(EOM_QWP):

    def measure(self):
        nst, st, v = super().measurement_run()
        return nst, st, v

    def bias_meas(self, bias_A, bias_B):
        bias_A /= 50
        bias_B /= 50
        nst, err, v =  super().measurement_run_volts(bias_A, bias_B)
        return nst, err, v

    def hwp_measurement(self, existing_res):
        nst, _, _ = self.measure()
        existing_res = list(existing_res)
        existing_res.append(nst)
        return existing_res

    def mse_stokes(self, measured, optimal):
        """Calculate the MSE with the optimal Stokes vector"""
        return np.sum((np.array(measured[1:]) - np.array(optimal[1:]))**2)

    def volts_mapping(self, vlim1=[-20,20], vlim2=[-20,20], n=51):
        volts1 = np.linspace(vlim1[0], vlim1[1], n)
        volts2 = np.linspace(vlim2[0], vlim2[1], n)
        coords1 = []
        coords2 = []
        coords3 = []
        nsts = []
        start = time()
        for v1 in volts1:
            for v2 in volts2:
```

```
                nst = self.bias_meas(v1, v2)
                nsts.append(nst)
                xyz1 = [v1,v2,nst[1]]
                xyz2 = [v1,v2,nst[2]]
                xyz3 = [v1,v2,nst[3]]
                coords1.append(xyz1)
                coords2.append(xyz2)
                coords3.append(xyz3)
        np.savez('volts_mapping', nsts=nsts, coords1=coords1, coords2=coords2, coords3=coords3, volts1=volts1, volts2=volts2)
        end = time()
        print("Time taken: ", end-start)
        return np.array(coords1), np.array(coords2), np.array(coords3), volts1, volts2


    def voltlist_test(self, voltlist):
        """Test voltages given in a list"""
        start = time()
        nsts = []
        for volts in voltlist:
            nst = self.bias_meas(volts[0], volts[1])
            nsts.append(nst)
        end = time()
        print("elapsed time: ", (end - start)/60)
        np.save("voltlist_test_feb", nsts)
        return nsts


    #def map_c3(self, vlim1=[-30,30], vlim2=[-30,30], n=41):
    #    """Does a map and return the 3 coordinates"""
    #    coords1, coords2, coords3, volts1, volts2 = self.volts_mapping(vlim1=vlim1, vlim2=vlim2, n=n)
    #    np.savez("map_c3_feb09", coords1=coords1, coords2=coords2, coords3=coords3)
    #    return coords1, coords2, coords3

    def map_c3(self, vlim1=[-30,30], vlim2=[-30,30], n=41):
        timestamp = datetime.now().strftime('%Y%m%d_%H%M')
        volts1 = np.linspace(vlim1[0], vlim1[1], n)
        volts2 = np.linspace(vlim2[0], vlim2[1], n)

```

```
79          nsts = []
80          errs = []
81          vs = []
82          voltages = np.zeros([n*n, 2])
83          count = 0
84
85          for v1 in tqdm(volts1):
86              for v2 in tqdm(volts2):
87                  nst, err, v = self.bias_meas(v1,v2)
88                  voltages[count] = [v1, v2]
89
90                  nsts.append(nst)
91                  errs.append(err)
92                  vs.append(v)
93                  count += 1
94          np.savez('voltmaps/voltmap_'+timestamp, nsts=nsts, errs
    =errs, vs=vs, voltages=voltages, volts1=volts1, volts2=
    volts2)
95
96      def map_rp(self, vlim1=[-1,1], vlim2=[-1,1], n=21):
97          timestamp = datetime.now().strftime('%Y%m%d_%H%M')
98          volts1 = np.linspace(vlim1[0], vlim1[1], n)
99          volts2 = np.linspace(vlim2[0], vlim2[1], n)
100
101         nsts = []
102         errs = []
103         vs = []
104         voltages = np.zeros([n*n, 2])
105         count = 0
106
107         for v1 in tqdm(volts1):
108             for v2 in tqdm(volts2):
109                 nst, err, v =  super().measurement_run_volts(v1
    , v2)
110                 voltages[count] = [v1, v2]
111
112                 nsts.append(nst)
113                 errs.append(err)
114                 vs.append(v)
115                 count += 1
116         np.savez('voltmaps/voltmap_'+timestamp, nsts=nsts, errs
    =errs, vs=vs, voltages=voltages, volts1=volts1, volts2=
    volts2)
117
```

```
118
119
120     def filter_equator(self, coords3, tol=0.1, printmode=False)
        :
121         if printmode == True:
122             print(coords3)
123         """For list of format (v1,v2,z), find where z=0"""
124         indices = np.where(np.abs(coords3[:,-1])<tol)
125         equatorial = coords3[indices]
126         z = equatorial[:,-1]
127         voltlist = equatorial[:,:-1]
128         np.savez("equator", voltlist=voltlist, indices=indices)
129         return voltlist, indices
130
131
132
133     def sort_voltlist(self, upperlim=2*np.pi, lowerlim=np.pi):
134         """Sorts the voltlist"""
135         with np.load("equator.npz") as data:
136             voltlist = data["voltlist"]
137             indices = data["indices"]
138         with np.load("map_c3.npz") as data:
139             coords1 = data["coords1"]
140             coords2 = data["coords2"]
141             coords3 = data["coords3"]
142
143         coords1 = coords1[indices]
144         coords2 = coords2[indices]
145         coords3 = coords3[indices]
146
147         x = coords1[0,:, -1]
148         y = coords2[0,:, -1]
149         angles = np.arctan2(y,x)
150         angles = (angles + np.pi)%(2*np.pi)
151         voltlist = voltlist[np.argsort(angles)]
152         angles = np.sort(angles)
153         halfvoltlist = voltlist[np.logical_and(angles<=upperlim
        , angles >=lowerlim)]
154         halfangles = angles[np.logical_and(angles <=upperlim,
        angles >=lowerlim)]
155         np.save("sort_voltlist", voltlist)
156         np.savez("angles_halfvoltlist", halfvoltlist=
        halfvoltlist, halfangles=halfangles)
157         return voltlist, halfvoltlist
```

```
158
159
160     def equator_full_test(self, vlim1=[-30,30], vlim2=[-30,30],
         n=41):
161         start = time()
162         coords1, coords2, coords3 = self.map_c3(vlim1=vlim1,
    vlim2=vlim2, n=n)
163         voltlist, indices = self.filter_equator(coords3)
164         voltlist, halfvoltlist = self.sort_voltlist()
165         nsts = self.voltlist_test(voltlist)
166         end = time()
167         print("Elapsed time: ",(end-start)/60)
168         voltlist = self.sort_voltlist()
169         return nsts
170
171
172
173
174     def find_equator_points(self, vlim1=[-30,30], vlim2
    =[-30,30], n=41, tol=1e-1):
175         """Find the points on the equatorial plane given the x,
    y,z coordinates"""
176         volts1 = np.linspace(vlim1[0], vlim1[1], n)
177         volts2 = np.linspace(vlim2[0], vlim2[1], n)
178         coords1 = []
179         coords2 = []
180         coords3 = []
181         nsts = []
182         start = time()
183         for v1 in volts1:
184             for v2 in volts2:
185                 nst = self.bias_meas(v1, v2)
186                 nsts.append(nst)
187                 xyz1 = [v1,v2,nst[1]]
188                 xyz2 = [v1,v2,nst[2]]
189                 xyz3 = [v1,v2,nst[3]]
190                 coords1.append(xyz1)
191                 coords2.append(xyz2)
192                 coords3.append(xyz3)
193         end = time()
194         print("Time taken (mins): ", (end-start)/60)
195         coords1 = np.array(coords1)
196         coords2 = np.array(coords2)
197         coords3 = np.array(coords3)
```

```
198         packed = zip(coords1[:,2], coords2[:,2], coords3[:,2],
    volts1, volts2)
199
200         equatorial = []
201         for data in packed:
202             if np.abs(data[2]) < tol:
203                 equatorial.append(data)
204
205         points = [data[:3] for data in equatorial]
206         voltlist = [data[3:] for data in equatorial]
207
208         nsts = self.voltlist_test(voltlist)
209
210         np.savez("equatorial_find", coords1=coords1, coords2=
    coords2, coords3=coords3, equatorial=equatorial, voltlist=
    voltlist)
211         return voltlist
212
213
214
215     def calibrate(self):
216         self.ser.write(b'calibrate\n')
217         nst,_,v = self.measure()
218         return nst
219     def findmin(self):
220         self.ser.write(b'findmin\n')
221         nst,_,v = self.measure()
222         return nst
223
224     def EOM_voltages(self, n, end=2*np.pi, va_bias=0, vb_bias
    =0, vo=10, vpi=10):
225         """Returns 2 arrays of voltages to put into pin 1/A and
     pin 2/B
226         which takes arguments n1, n2 referring to number of
    voltages."""
227
228         nsts = []
229         va = lambda alpha: self.v_a(alpha, vo=vo, vpi=vpi, bias
    =va_bias)
230         vb = lambda alpha: self.v_b(alpha, vo=vo, vpi=vpi, bias
    =vb_bias)
231         volts_a = np.array(list(map(va, np.linspace(0, end, n))
    ))
232         volts_b = np.array(list(map(vb, np.linspace(0, end, n))
```

```
    ))
233
234         # ensures voltage demanded from RP is not too big
235         volts_a = np.where(volts_a > 50, 0, volts_a)
236         volts_b = np.where(volts_b > 50, 0, volts_b)
237
238         for i in range(n):
239             v1 = volts_a[i]/50
240             v2 = volts_b[i]/50
241
242             nst, _, _ = self.measurement_run_volts(v1, v2)
243             nsts.append(nst)
244
245         np.save('specsheet_hwp',nsts)
246
247         return nsts
248
249
250     def pso_cost_function_cn(self, va_bias=0, vb_bias=0, vo=10,
     vpi=10, n=20):
251         """Calculate cost function"""
252         # follow the format of va_bias, vb_bias, vo, vpi
253         nsts = self.EOM_voltages(n, va_bias=va_bias, vb_bias=
    vb_bias, vo=vo, vpi=vpi)
254
255         # calculate error
256         points = np.array(nsts).T[1:]
257         points1 = points[:,:int(n/2)]
258         points2 = points[:,int(n/2):]
259         return np.sum((points1-points2)**2)
260
261     def pso_cost_function(self, va_bias=0, vb_bias=0, vo=10,
    vpi=10, n=20):
262         """Cost function where ideal distribution is on
    equatorial plane"""
263         m = int(n/2)
264         thetas = np.linspace(0, 2*np.pi, m)
265         x = np.cos(thetas)
266         y = np.sin(thetas)
267         z = np.zeros(m)
268         coords = np.reshape(np.array([x,y,z]), [3,m])
269         # follow the format of va_bias, vb_bias, vo, vpi
270         nsts = self.EOM_voltages(n, va_bias=va_bias, vb_bias=
    vb_bias, vo=vo, vpi=vpi)
```

```
271
272         # calculate error
273         points = np.array(nsts).T[1:]
274         points1 = points[:,:m]
275         points2 = points[:,m:]
276         p1_error = np.sum((points1-coords)**2)
277         p2_error = np.sum((points2-coords)**2)
278         return p1_error + p2_error
279     def particle_swarm(self, n_particles=5, n=20, kv=1,
    iterations=10, w=0.2, cognitive=1, social=1):
280         """Run swarm algorithm. Could use a class approach but
    KISS."""
281         start = time()
282         particle_pos = np.zeros([n_particles, 4])
283         best_known_pos = np.zeros([n_particles, 4])
284         particle_vel = np.zeros([n_particles, 4])
285         global_best = np.array([0,0,10,10]) # global best
286         particle_best_error = np.zeros([n_particles,])
287         global_error = self.pso_cost_function(va_bias=
    global_best[0], vb_bias=global_best[0], vo=global_best[0],
    vpi=global_best[0], n=n)
288         for i in range(n_particles):
289             # initialise position
290             init_pos = np.concatenate([-5+10*np.random.random
    (2),-15+30*np.random.random(2)])
291             particle_error = self.pso_cost_function(va_bias=
    init_pos[0], vb_bias=init_pos[0], vo=init_pos[0], vpi=
    init_pos[0], n=n)
292             particle_best_error[i] = particle_error
293             if particle_error < global_error:
294                 global_best = init_pos
295             # initialise velocity
296             init_vel = kv*np.random.random([4])
297             # append into vectors that describe particle
298             particle_pos[i] = best_known_pos[i] = init_pos
299             particle_vel[i] = init_vel
300             # make array to store error of each iteration
301             error_over_iterations = np.zeros(iterations)
302         for i in range(iterations):
303             for j in range(n_particles):
304                 rp = np.random.random(4)
305                 rg = np.random.random(4)
306                 velocity =  w*particle_vel[j] + cognitive*rp*(
    best_known_pos[j]-particle_pos[j]) + social*rg*(global_best
```

```
-particle_pos[j])
307                 particle_pos[j] += velocity
308                 # check error
309                 pos = particle_pos[j]
310                 particle_error = self.pso_cost_function(va_bias
=pos[0], vb_bias=pos[1], vo=pos[2], vpi=pos[3], n=n)
311                 if particle_error < particle_best_error[j]:
312                     particle_best_error[j] = particle_error
313                     best_known_pos[j] = pos
314                     if particle_error < global_error:
315                         global_error = particle_error
316                         global_best = pos
317             error_over_iterations[i] = global_error
318             print(particle_pos)
319             print(best_known_pos)
320             print(particle_vel)
321             print(particle_best_error)
322 
323         end = time()
324 
325         print("Time elapsed: ", timedelta(seconds=end - start))
326         nsts = self.EOM_voltages(n, va_bias=global_best[0],
vb_bias=global_best[1], vo=global_best[2], vpi=global_best
[3])
327 
328         return global_best, error_over_iterations
329 
330 
331     def stability(self, repeats=100, wait=60, v1=-6.6, v2=0.4):
332         nsts = []
333         traces = []
334         error = []
335         for i in tqdm(range(repeats)):
336              self.test_volts(v1,v2, off=False)
337              nstp, pcov,  v = self.measure()
338              nsts.append(nstp)
339              traces.append(v)
340              error.append(pcov)
341              np.save('calibration_data/nsts4', np.array(nsts))
342              np.save('calibration_data/traces4', np.array(
traces))
343              np.save('calibration_data/error4', np.array(error)
)
344              sleep(wait)
```

```
345         super().turnoff_volt()
346 
347     ### functions for output
```

Bibliography

[1]J. Millen, T. S. Monteiro, R. Pettit, and A. N. Vamivakas, "Optomechanics with levitated particles", en, Reports on Progress in Physics **83**, Publisher: IOP Publishing, 026401 (2020).

[2]A. Belenchia, M. Carlesso, Ö. Bayraktar, D. Dequal, I. Derkach, G. Gasbarri, W. Herr, Y. L. Li, M. Rademacher, J. Sidhu, D. K. L. Oi, S. T. Seidel, R. Kaltenbaek, C. Marquardt, H. Ulbricht, V. C. Usenko, L. Wörner, A. Xuereb, M. Paternostro, and A. Bassi, "Quantum physics in space", en, Physics Reports, Quantum Physics in Space **951**, 1–70 (2022).

[3]M. Rademacher, M. Konopik, M. Debiossac, D. Grass, E. Lutz, and N. Kiesel, "Nonequilibrium Control of Thermal and Mechanical Changes in a Levitated System", Physical Review Letters **128**, Publisher: American Physical Society, 070601 (2022).

[4]M. Rademacher, J. Millen, and Y. L. Li, "Quantum sensing with nanoparticles for gravimetry: when bigger is better", en, Advanced Optical Technologies **9**, Publisher: De Gruyter, 227–239 (2020).

[5]J. Bell, "Against 'measurement'", en, Physics World **3**, Publisher: IOP Publishing, 33–41 (1990).

[6]E. Schrödinger, "Die gegenwärtige Situation in der Quantenmechanik", de, Naturwissenschaften **23**, 807–812 (1935).

[7]*Schroedinger: "The Present Situation in Quantum Mechanics"*, Dec. 2012.

[8]C. Gonzalez-Ballestero, M. Aspelmeyer, L. Novotny, R. Quidant, and O. Romero-Isart, "Levitodynamics: Levitation and control of microscopic objects in vacuum", Science **374**, Publisher: American Association for the Advancement of Science, eabg3027 (2021).

[9]U. Delić, M. Reisenbauer, K. Dare, D. Grass, V. Vuletić, N. Kiesel, and M. Aspelmeyer, "Cooling of a levitated nanoparticle to the motional quantum ground state", Science **367**, Publisher: American Association for the Advancement of Science, 892–895 (2020).

[10]D. Windey, C. Gonzalez-Ballestero, P. Maurer, L. Novotny, O. Romero-Isart, and R. Reimann, "Cavity-Based 3D Cooling of a Levitated Nanoparticle via Coherent Scattering", Physical Review Letters **122**, Publisher: American Physical Society, 123601 (2019).

[11]J. Millen, T. Deesuwan, P. Barker, and J. Anders, "Nanoscale temperature measurements using non-equilibrium Brownian dynamics of a levitated nanosphere", en, Nature Nanotechnology **9**, Number: 6 Publisher: Nature Publishing Group, 425–429 (2014).

[12]J. Bang, T. Seberson, P. Ju, J. Ahn, Z. Xu, X. Gao, F. Robicheaux, and T. Li, "Five-dimensional cooling and nonlinear dynamics of an optically levitated nanodumbbell", Physical Review Research **2**, Publisher: American Physical Society, 043054 (2020).

[13]B. A. Stickler, B. Papendell, S. Kuhn, B. Schrinski, J. Millen, M. Arndt, and K. Hornberger, "Probing macroscopic quantum superpositions with nanorotors", en, New Journal of Physics **20**, Publisher: IOP Publishing, 122001 (2018).

[14]B. A. Stickler, K. Hornberger, and M. S. Kim, "Quantum rotations of nanoparticles", en, Nature Reviews Physics **3**, Number: 8 Publisher: Nature Publishing Group, 589–597 (2021).

[15]R. A. Norte, J. P. Moura, and S. Gröblacher, "Mechanical Resonators for Quantum Optomechanics Experiments at Room Temperature", Physical Review Letters **116**, Publisher: American Physical Society, 147202 (2016).

[16]H. Shi and M. Bhattacharya, "Optomechanics based on angular momentum exchange between light and matter", en, Journal of Physics B: Atomic, Molecular and Optical Physics **49**, Publisher: IOP Publishing, 153001 (2016).

[17]S. Kuhn, P. Asenbaum, A. Kosloff, M. Sclafani, B. A. Stickler, S. Nimmrichter, K. Hornberger, O. Cheshnovsky, F. Patolsky, and M. Arndt, "Cavity-Assisted Manipulation of Freely Rotating Silicon Nanorods in High Vac-

uum", Nano Letters **15**, Publisher: American Chemical Society, 5604–5608 (2015).

[18] J. Ahn, Z. Xu, J. Bang, Y.-H. Deng, T. M. Hoang, Q. Han, R.-M. Ma, and T. Li, "Optically Levitated Nanodumbbell Torsion Balance and GHz Nanomechanical Rotor", Physical Review Letters **121**, Publisher: American Physical Society, 033603 (2018).

[19] *Optically Levitated Nanospheres for Cavity Quantum Optomechanics — Quantum Optomechanics and Nanomechanics: Lecture Notes of the Les Houches Summer School: Volume 105, August 2015 — Oxford Academic.*

[20] A. Pontin, H. Fu, M. Toroš, T. S. Monteiro, and P. F. Barker, "Simultaneous cavity cooling of all six degrees of freedom of a levitated nanoparticle", en, Nature Physics, Publisher: Nature Publishing Group, 1–6 (2023).

[21] M. Aspelmeyer, T. J. Kippenberg, and F. Marquardt, "Cavity optomechanics", en, Reviews of Modern Physics **86**, 1391–1452 (2014).

[22] D. H. Goldstein, *Polarized Light*, 3rd ed. (CRC Press, Boca Raton, Jan. 2017).

[23] J. Karczmarek, J. Wright, P. Corkum, and M. Ivanov, "Optical Centrifuge for Molecules", Physical Review Letters **82**, Publisher: American Physical Society, 3420–3423 (1999).

[24] M. Antia, "Spinning the Molecule", en, Physics **3**, Publisher: American Physical Society, 24 (1999).

[25] A. Korobenko, "Control of molecular rotation with an optical centrifuge", en, Journal of Physics B: Atomic, Molecular and Optical Physics **51**, Publisher: IOP Publishing, 203001 (2018).

[26] A. Owens, A. Yachmenev, and J. Küpper, "Coherent Control of the Rotation Axis of Molecular Superrotors", The Journal of Physical Chemistry Letters **9**, Publisher: American Chemical Society, 4206–4209 (2018).

[27] R. W. Minck, E. E. Hagenlocker, and W. G. Rado, "Stimulated Pure Rotational Raman Scattering in Deuterium", Physical Review Letters **17**, Publisher: American Physical Society, 229–231 (1966).

[28] *Microwave Spectroscopy - an overview — ScienceDirect Topics.*

[29] *Rotational-Raman Spectrum - an overview — ScienceDirect Topics.*

[30]H. Stapelfeldt and T. Seideman, "Colloquium: Aligning molecules with strong laser pulses", Reviews of Modern Physics **75**, Publisher: American Physical Society, 543–557 (2003).

[31]B. Schaefer, E. Collett, R. Smyth, D. Barrett, and B. Fraher, "Measuring the Stokes polarization parameters", American Journal of Physics **75**, Publisher: American Association of Physics Teachers, 163–168 (2007).

[32]*Rotation Mount with Resonant Piezoelectric Motors*, en.

[33]D. P. G. Nilsson, T. Dahlberg, and M. Andersson, "Step-by-step guide to 3D print motorized rotation mounts for optical applications", EN, Applied Optics **60**, Publisher: Optica Publishing Group, 3764–3771 (2021).

[34]*Red Pitaya - Swiss Army Knife For Engineers*, en-US, May 2021.

[35]*List of supported SCPI commands — Red Pitaya 0.97 documentation.*

[36]*Polarization Controllers*, en-US.

[37]M. Rademacher, J. Gosling, A. Pontin, M. Toroš, J. T. Mulder, A. J. Houtepen, and P. F. Barker, "Measurement of single nanoparticle anisotropy by laser induced optical alignment and Rayleigh scattering for determining particle morphology", en, Applied Physics Letters **121**, 221102 (2022).

[38]S. Kuhn, B. A. Stickler, A. Kosloff, F. Patolsky, K. Hornberger, M. Arndt, and J. Millen, "Optically driven ultra-stable nanomechanical rotor", en, Nature Communications **8**, Number: 1 Publisher: Nature Publishing Group, 1670 (2017).