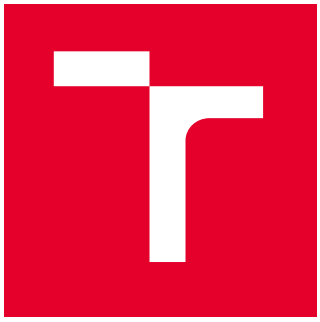


BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

STABILIZATION OF MACROSCOPIC PARTICLE IN OPTICAL TRAP

STABILIZACE MAKROSKOPICKÉ ČÁSTICE V OPTICKÉ PASTI

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Vojtěch Mlynář**

AUTOR PRÁCE

SUPERVISOR — **Ing. Martin Brablc**

VEDOUCÍ PRÁCE

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Master's Thesis

Institut:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Bc. Vojtěch Mlynář**
Degree programm:	Mechatronics
Branch:	no specialisation
Supervisor:	**Ing. Martin Brablc**
Academic year:	2021/22

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Stabilization of macroscopic particle in optical trap

Brief Description:

A macroscopic particle trapped in the vacuum using an optical tweezer constitutes a harmonic oscillator with natural frequency ranging from tens to hundreds of kHz and multiple degrees of freedom, which is well isolated from the environment and thus can serve as an extremely precise force sensor, or as an experimental setup for exploration of quantum mechanics. Feedback control of the particle is required to stabilize the particle and as a preparation for future experiments. The universal FPGA Red Pitaya is capable of such control. The main goal of the thesis is to evaluate the possibilities of the Red Pitaya hardware and to implement basic algorithms. The thesis assignment is based on cooperation with the Institute of Scientific Instruments of the Czech Academy of Sciences.

Master's Thesis goals:

1. Research the applications of the Red Pitaya hardware regarding the control of optical and mechatronic systems. Focus on the use of FPGAs and automatic HDL code generation.

2. Demonstrate basic functionality on Red Pitaya FPGA – LED blinking, sampling of an analog signal, saving data to a file, signal filtering and analog signal output.

3. Simulate an experiment on the FPGA – Stabilization via a standard feedback control algorithm for the target system with adequate loop frequency.

4. Simulate an experiment on the FPGA – Stabilization via feedback controller with state observer (e.g. locally optimal LQG).

5. Try to implement experiments specified in 4 and 5 on the real experimental setup and evaluate the performance of the control algorithms.

Recommended bibliography:

LABBE, Roger. Kalman and Bayesian Filters in Python, Dostupné z: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python

NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, 2011. ISBN 978-364-2086-748.

LJUNG, Lennart. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0136566953.

NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 80-722-50-0-2.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2021/22

In Brno,

L. S.

prof. Ing. Jindřich Petruška, CSc. Director of the Institute	doc. Ing. Jaroslav Katolický, Ph.D. FME dean

Summary

This thesis deals with the implementation and evaluation of the feedback stabilization algorithms for a particle in an optical trap. The algorithms are implemented in a field-programmable gate array Red Pitaya STEMlab 125-14 leveraging HDL code autogeneration from Simulink models. Theoretical background is laid out to highlight the connection of underlying physics to the Kalman filter theory, together with the details of FPGA implementation and parameter estimation. The implemented algorithms were successfully tested in a real experimental setup, and the results are presented in the last chapter.

Abstrakt

Cílem této práce je implementace a otestování algoritmů pro zpětnovazební stabilizaci částice v optické pasti. Tyto algoritmy jsou implementovány v programovatelném hradlovém poli Red Pitaya STEMlab 125-14 s využitím automatického generování kódu z modelů v programu Simulink. V práci je popsán fyzikální základ řízeného systému, jeho návaznost na teorii Kálmánova filtru, detaily implementovaných algoritmů v FPGA a procedury využité pro odhad parametrů. Implementované algoritmy byly úspěšně otestovány na reálné sestavě a výsledky jsou ukázány v závěru práce.

Keywords

Stochastic system, photonics, optical trap, optical tweezers, feedback stabilization, feedback cooling, Kalman filter, LQR, parameter estimation, FPGA, Red Pitaya

Klíčová slova

Stochastický systém, fotonika, optická past, optická pinzeta, zpětnovazební stabilizace, zpětnovazební chlazení, Kálmánův filtr, LQR, odhad parametrů, FPGA, Red Pitaya

Bibliographic citation

MLYNÁŘ, Vojtěch. *Stabilization of macroscopic particle in optical trap.* Brno: Brno University of Technology, Faculty of Mechanical Engineering, 2022. 82 pages, Master's thesis supervisor: Ing. Martin Brablc.

Rozšířený abstrakt

Optické pinzety jsou technologií aplikovanou již od 70. let minulého století k manipulaci s mikroskopickými objekty jako jsou například bakterie, buňky, ale také uměle vytvořené částice různých tvarů za pomocí světelného paprsku. V posledních letech byly tyto přístroje úspěšně využity pro studium chování mikroskopických částic blízko absolutní nule, kdy jsou zkoumány termodynamické a stochastické procesy, kvantově-mechanické jevy a další.

Samotná manipulace s pevnými částicemi pomocí světla je umožněna díky změně hybnosti světla. Pro ilustraci je uvažována kulová pevná částice a zaostřený paprsek světla s Gaussovským rozložením intenzity (nejsilnější uprostřed a klesající k okraji). Když se částice nachází v paprsku, tak dochází k ohybu světla, čímž se mění hybnost fotonů. Díky zákonu akce a reakce potom působí na částici síla, která částici udržuje v ohnisku čočky, která zaostřuje paprsek. Tato síla dokonce umožňuje překonat tíhové zrychlení a tím zachytit mikroskopické částice v prostoru. Pokud je částice elektricky nabitá, je možné na ni pomocí elektrod působit dodatečnou silou a vhodnou modulací této síly ji stabilizovat a minimalizovat tak její energii.

Jelikož se částice nachází ve vakuové komoře za sníženého tlaku, je stále v kontaktu se vzduchem a s ním v termodynamické rovnováze. Tento plyn částici budí náhodnými impulzy způsobenými kolizemi s dalšími částicemi, které se zároveň projevují také jako tlumení (viz Brownův pohyb). Síla vyvinutá paprskem světla působí proti výchylce částice z rovnovážné polohy, čímž vznikne slabě tlumený harmonický oscilátor buzený šumem. Pohybová energie tohoto oscilátoru je vázaná na teplotu okolního prostředí, nicméně vhodnou modulací zpětnovazební síly lze částici „brzdit“, snížit tím její pohybovou energii a tím ekvivalentně snížit její teplotu těžiště, nezávisle na reálné teplotě materiálu částice.

Vlastní frekvence tohoto oscilátoru se pohybuje řádově v desítkách až stovkách kHz, a navíc je buzen bílým šumem. Z těchto důvodů je nutné zajistit co nejvyšší frekvenci zpětné vazby zároveň s co nejmenším časovým zpožděním a možností ladit jejich parametry během experimentu. Tyto požadavky splňují programovatelná hradlová pole (Field-programmable gate array, FPGA), která umožňují aplikovat algoritmy s frekvencemi v řádu MHz až GHz díky možnosti paralelizace výpočetních operací.

Cílem této práce bylo implementovat algoritmy filtrace signálu a zpětnovazební stabilizace na FPGA Red Pitaya a následně je otestovat na reálné sestavě ve spolupráci s Ústavem přístrojové techniky Akademie věd České republiky. V úvodu práce je představen teoretický základ stochastických systémů, optické pinzety a algoritmů zpracování signálu, konkrétně Butterworthova a Kálmánova filtru. V další kapitole je uvedena konkrétní implementace zmíněných algoritmů, včetně popisu pracovního postupu s využitím automatického generování kódu z modelů vytvořených v Simulinku a rozhraní použitého ke konfiguraci FPGA během experimentu. V kapitole 4 je představena experimentální sestava a procedury použité pro odhad parametrů sestavy, které jsou v experimentu použity pro nastavení Kálmánova filtru. Následující kapitola 5 demonstruje dosažené výsledky z experimentů, konkrétně porovnání různých přístupů k filtrování

signálu a porovnání efektivity variant zpětné vazby.

V práci byly úspěšně implementovány Butterworthův filtr (spodní propusť) a Kálmánův filtr na FPGA s vzorkovací frekvencí 5 MHz a výpočetním zpožděním menším než 200 ns. Tyto algoritmy jsou konfigurovatelné v reálném čase pomocí Matlab aplikace s grafickým rozhraním a přidruženého TCP/IP serveru běžícího na procesorové části FPGA. Filtrovaný signál je možné vynásobit vhodným koeficient a pomocí digitálně-analogového převodníku s ním řídit částici v optické pasti. Díky tomu, že Kálmánův filtr obsahuje informace o systému v podobě stavového modelu dosáhl výrazně lepší výsledky než Butterworthův filtr a zároveň je implementačně podobně náročný. Zajímavostí je, že částice v optické pasti výborně odpovídá teoretickým předpokladům pro lineární Kálmánův filtr a tak jej lze naladit přesně podle odhadnutých parametrů (statistických charakteristik procesního šumu a šumu měření).

Odhad stavových veličin systému získaný z Kálmánova filtru je následně využit k stavovému zpětnovazebnímu řízení naladěnému metodou lineárně-kvadratického regulátoru (LQR). Touto metodou bylo dosaženo redukce rozptylu výchylky částice o 95 %, což lze přirovnat k snížení teploty z 23 na -257 stupňů Celsia (15,93 stupně nad absolutní nulou). Druhá metoda založené na zpětné vazbě odpovídající pouze rychlosti, získané buďto numerickou derivací nebo z Kálmánova filtru, dosáhla největší redukce 87 %.

Bylo úspěšně ověřeno, že lze automaticky generovaný kód ze Simulinku nasadit na reálnou sestavu a že tato metoda ulehčuje proces vývoje numericky složitých algoritmů. Dalšími možnostmi pokračování této práce může být rozšíření modelu v Kálmánovu filtru na všechny tři osy místo současné jedné, případně dokonce řídit dvě částice v pasti zároveň.

I declare that this thesis was composed solely by myself under the supervision of Ing. Martin Brablc, using the listed sources and clearly referencing the contributions of others.

Bc. Vojtěch Mlynář

Brno, May 18, 2022

I would like to express my gratitude to my supervisor, Ing. Martin Brablc, who has been a great advisor on many practical and theoretical problems. This endeavor would not have been possible without the Institute of Scientific Instruments of the Czech Academy of Sciences, where I extend my sincere thanks to Mgr. Oto Brzobohatý, Ph.D. and Mgr. Martin Duchaň for their cooperation during the first experiments, and mainly to Mgr. Vojtěch Svak for his dedicated effort and numerous fruitful consultations. Lastly, I would be remiss not to mention my girlfriend, who always supported me enthusiastically and unconditionally.

Bc. Vojtěch Mlynář

Contents

List of Abbreviations

API	Application programming interface
CM	Center of mass (temperature)
DMA	Direct memory access
DSP	Digital signal processing
FIR	Finite impulse response
FPGA	Field-programmable gate array
HDL	Hardware description language
IIR	Infinite impulse response
IP Core	Intellectual property core
LQR	Linear quadratic regulator
PSD	Power spectral density
SoC	System on chip

1 Introduction

The main task of this thesis is to implement and test algorithms for feedback cooling of the particle in the optical trap in Red Pitaya FPGA and stems from the cooperation with the Institute of Scientific Instruments of the Czech Academy of Sciences. Levitational photonics, specifically optical levitation of micro- and nanospheres, provide valuable insight into the interaction of particles with fluids, stochastic mechanics, and even quantum mechanics. The spherical particle with a diameter in a micrometer range is suspended in a gaseous environment by a focused light beam, which counteracts the gravitational pull on the particle and thus the particle is levitated without mechanical coupling [1]. The absence of such coupling enables the use of the particle for precise force sensing and the study of interactions between particles at the microscale [2].

The particle is in thermodynamical equilibrium with the surrounding medium, which constantly perturbs the particle. This motion can be characterized by its variance, and by applying a stabilization feedback loop to the system, the movement of the particle (precisely its center of mass) can be attenuated or the variance reduced. The reduced variance is then equivalent to some temperature (regardless of the actual temperature of the particle material), which is why this process is called *feedback cooling*. The physical limit for feedback cooling is absolute zero (0 Kelvin), and several teams have been able to cool microspheres very close to this limit [3, 4]. Hence, absolute zero (or ground state) represents the lowest energy the particle can attain. Feedback cooling is thus an invaluable tool for studying quantum physics at (relatively) large scales, with its importance in physics, precise sensing technology, and even emerging quantum computers.

In this particular experimental setup (see Figure 1.1), the particle is trapped by two focused counterpropagating Gaussian light beams that levitate the particle. Particle displacement scatters a portion of the light, which is in turn detected by the quadrant photodiode that transduces the incident photons into a voltage signal. According to this signal, a feedback force can be exerted on the (electrically charged) particle via modulation of the electric field in the vacuum chamber. The Red Pitaya FPGA in this case samples the displacement signal and produces voltage control signal applied to the electrodes directly through an operational amplifier. The experimental setup is introduced in greater detail in Section 4.1.

Although the underlying physical principles for a particle in the optical trap are quite complex, it can be reasonably approximated by a well-known mass-spring-damper system with a single degree of freedom driven by white noise. The driving force is the result of interaction with the medium, as the trapped particle collides with other particles. These collisions also inhibit the free motion of the particle, which manifests itself as weak damping in the system. The optical trap then adds the restoring force, which creates a harmonic oscillator with natural frequency ranging from tens to hundreds of kilohertz and a very weak damping. This system also conveniently maps very well to the Kalman filter model with process and measurement noise, which is implemented in the FPGA together

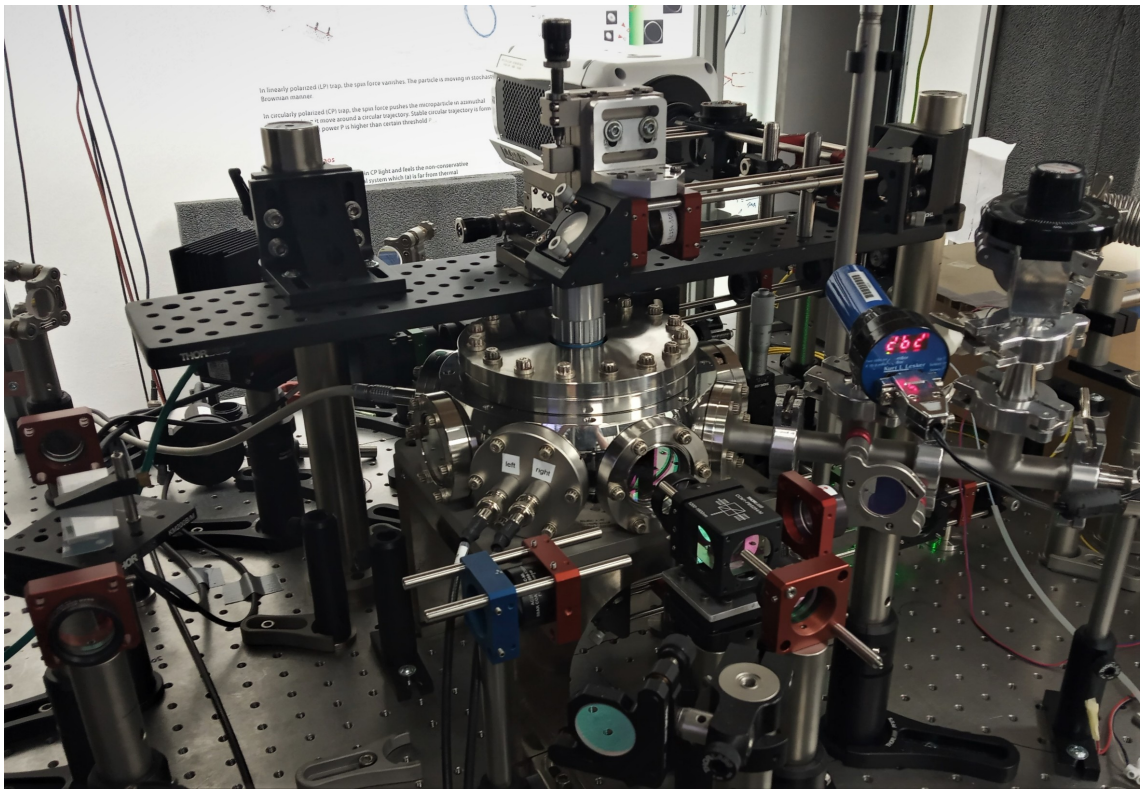


Figure 1.1: Photograph of the experimental setup, with the vacuum chamber (center) and part of the optical system (author V. Svak)

with the full-state regulator to enable feedback cooling.

Due to the stochastic nature of the system and its sensitivity to the surrounding environment, the parameters of the system vary from particle to particle, experiment to experiment, and even drift during the experiment itself. Therefore, it is inconvenient to work with hard-coded programs and quick and easy configuration of the signal processing algorithms is required instead. Because the particle is constantly driven by noise and its trajectory is unpredictable, signal processing must be performed at very high frequency (megahertz and higher) and also very low latency to be able to stabilize the particle efficiently. These requirements are well met by system-on-chip field-programmable gate arrays (SoC FPGA). These devices integrate the flexibility of the microprocessor and the massive throughput and parallelism of FPGAs, providing the ideal candidate for controlling such systems.

However, the implementation of digital signal processing algorithms on the FPGA is a complicated process because most of the FPGAs do not support floating-point arithmetic and timing of the blocks is critical. Fortunately, these difficulties can be aided by the hardware description language (HDL) autogeneration. In this thesis, the DSP algorithms are first developed in Simulink and then converted to fixed-point representation with the Simulink Fixed Point Tool and the HDL module is generated with the HDL Coder. The generated code is then integrated in the Vivado (Xilinx FPGA development tool). Configuration of the DSP algorithms in the FPGA ensures C server application running in the integrated microprocessor. This server communicates with the Matlab application providing a graphical user interface and other functions required for successful operation of the controller.

2 Theoretical Survey

This chapter presents the theoretical foundation for implemented algorithms. A brief overview of stochastic system theory is presented to clarify the used notation. The physical model of the particle in the optical trap is described, as well as the simplified model used to develop control schemes.

Signal filtering algorithms such as Kalman and infinite impulse response filters are displayed, further enabling various methods of feedback particle cooling. Finally, system-on-chip FPGAs and the implementation of digital signal processing algorithms are discussed.

2.1 Description of stochastic systems

When observing a free particle suspended in a fluid medium, its velocity and position appear to fluctuate randomly due to collisions with other particles in the medium - this random walk is called Brownian motion [5]. After the particle is caught in an optical trap, it still exhibits random fluctuations; however, the particle's movement is now additionally influenced by the deterministic interaction with the laser beam. Therefore, a particle in the optical trap remains a stochastic system, and this section lays out the notation and concepts used in this thesis.

2.1.1 Langevin equation

Usually, systems in engineering practice are deterministic, which means that probabilistic effects on the state evolution of the system are negligible. There exist a plethora of theories that describe random processes, but for the purposes of this thesis, a description in the form of stochastic differential equations is desirable. One such suitable description is the Langevin equation, which describes the evolution of system states under both deterministic and stochastic (random) effects [6].

The original Langevin equation describing the Brownian motion of a particle in fluid medium is as follows [5]:

$$m\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\gamma\mathbf{v} + \boldsymbol{\eta}(t) \tag{2.1}$$

The m is the mass of the particle, $\boldsymbol{v}$ its velocity, γ is the viscous damping due to collisions with other particles (Stokes law), and $\boldsymbol{\eta}(t)$ is the stochastic term representing random force impulses. It is important to remember that $\boldsymbol{\eta}(t)$ is not a function of time in the deterministic sense, but a white noise sampled at time t specified by mean (Eq. 2.2) and correlation function (Eq. 2.3) [7]:

$$Mean(\eta(t)) = \mu_\eta = \mathbf{E}[\eta(t)] = \langle \eta(t) \rangle = 0 \tag{2.2}$$

$$R(\tau)_{\eta\eta} = \langle \eta(t) \cdot \eta(t+\tau) \rangle = 2\gamma k_B T \delta(\tau) \tag{2.3}$$

From equation (2.3), γ is the Stokes damping coefficient, k_B the Boltzmann constant, and T is the bath (surrounding medium) temperature. $\delta(\tau)$ is Dirac delta, meaning that the correlation function $R(\tau)_{\eta\eta}$ is uncorrelated for time difference τ greater than zero. Delta-correlation requires the assumption that the sampling time is significantly longer than the time of individual collisions. Molecules in water experience approximately 10^{14} collisions per second [8], so for most of the realizable sampling frequencies, this assumption holds well.

In the next chapter, this equation will be further modified by adding terms representing the optical trap restoring force as well as feedback force.

2.1.2 Power spectral density

Although the random processes are realized in time domain, their time trace (e.g. position, temperature, etc.) usually offers very little information that is intuitively useful. Apart from the usual statistical measures such as mean, variance, covariance, and others, transforming the signal into the frequency domain provides a clearer view detached from the time domain. One representation of this transformation is *Power spectral density* (PSD) or, as sometimes called, simply *Power spectrum*.

PSD is a measure of the average power contained in the signal component with frequency ω. Units generally correspond to signal variance per Hertz ($(physical\ unit)^2/Hz$), so PSD of displacement would be in m^2/Hz and voltage would be in V^2/Hz, etc.

For a deterministic function $x(t)$, the first definition of PSD is using the autocorrelation function $R_x(\tau)$ as [9, 10]:

$$S_x(\omega) = \int_{-\infty}^{\infty} R_X(\tau) e^{-i\omega t} d\tau, \quad R_x(\tau) = \mathbf{E}[x(t), x(t+\tau)] \tag{2.4}$$

The other definition is based on the frequency domain function $X(i\omega)$, which can be obtained, for example, by the Fourier or Laplace transform from the time domain function $x(t)$ [9, p. 3]:

$$S_x(\omega) = |X(i\omega)|^2 \tag{2.5}$$

Obtaining PSD estimate from a discrete-time measurement is less straightforward, with the easiest option being *Periodogram* [9]:

$$\hat{S}_x(\omega) = \frac{1}{N} \left| \sum_{t=0}^{N} x(t) e^{-i\omega t} \right|^2 \tag{2.6}$$

Periodogram (Eq. 2.6) can be computed using the Fast Fourier Transform (FFT)

algorithm implemented in most signal processing tools. Unfortunately, a periodogram is not the best PSD estimator because increasing the number of samples N does not reduce the variance of the periodogram; it only increases frequency resolution.

PSD estimate can be improved by various techniques, which usually utilize signal windowing, ensemble averaging, or both [9]. This thesis uses mainly the Bartlett method [11], which divides the time trace with N samples into $L = N/M$ subsets each containing M samples. Each sub-periodogram $\hat{S}_j(\omega)$ is computed using equation 2.6 on M samples and then averaged:

$$\hat{S}_x(\omega) = \frac{1}{L} \sum_{j=1}^{L} \hat{S}_j(\omega) \tag{2.7}$$

The Bartlett method apparently trades off frequency resolution for variance reduction because subsets cannot overlap. It is up to every user to balance this trade-off for the specific application.

2.1.3 Spectral analysis

Usually, obtaining analytical frequency-domain representation $X(\omega)$ of an arbitrary deterministic system $x(t)$ is straightforward. However, stochastic systems (such as Eq. 2.1) do not have a defined time-domain solution (which is dependent on sampling of a random variable). If the stochastic term can be expressed as a system's input, its statistical characteristics are known, and the deterministic part is linear and time-invariant, static frequency domain analysis is still possible [10].

Then, for a signal with a spectrum defined by $S_u(i\omega)$ passing through a linear system with frequency response $H(i\omega)$ (see Fig. 2.1), the output's power spectral density is obtained as:

$$S_y(i\omega) = |H(i\omega)|^2 \, S_u(i\omega) \tag{2.8}$$

The input's spectral density function can be arbritrarily complex, as long as there exists an analytical description. The simplest case is, of course, when the input is white noise which is usually defined as [12]:

- Random sequence has zero mean.
- Contains all frequencies at equal power, thus producing constant PSD up to infinity.
- Every sample is uncorrelated with any other sample: $R(\tau) = k \cdot \delta(\tau)$.

The problem with the previous definition is that such a random process would contain infinite energy. The spectrum is theoretically infinite, and signal's energy can be determined by integrating the power spectral density; hence is the integral (and the energy) infinite. However, real-world systems are typically band-limited and pure white noise cannot be observed.

It is possible to mathematically construct a band-limited white noise (e.g. [13]), but this autocorrelation function is significantly more complicated. Fortunately, when analyzing band-limited systems, pure white noise can be used without band limitation, because

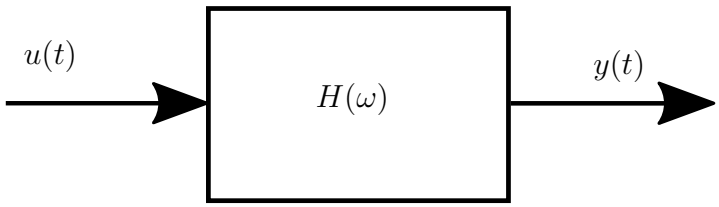


Figure 2.1: Noise shaping filter

the contribution of high frequencies (which are attenuated) quickly becomes negligible [10].

2.2 Particle in the optical trap

First in this section, a brief overview of optical tweezers is introduced to give the reader a general understanding of the technology. Then, a physical model of the trapped particle is laid out to clarify the physical principles associated with particle trapping, which will later serve as a foundation for the Kalman filter and associated control strategies.

Note that this section only skims the surface of the particle-trapping physical foundation and serves only as a brief introduction necessary from the control theory point of view. For a detailed explanation and derivation, see some of the cited sources [2, 14, 15].

2.2.1 Optical tweezers

The core principle of optical trapping is that light carries momentum and thus can exert force on other objects, however, the scale of this phenomenon was originally deemed too insignificant to overcome other forces, such as gravity and friction [14]. With the advent of laser technology, A. Ashkin and others have been able to demonstrate that a tightly focused beam of light can be used to trap single cells and other microscopic particles [1, 16], for which Ashkin was later awarded the Nobel prize.

Optical tweezers today are used to trap and manipulate various objects and particles ranging in scale from tens of nanometers (particles, viruses, etc.) to hundreds of micrometers (e.g. cells) and are used in many fields including experimental physics, biology, and others.

A typical optical tweezer consists of a light source (laser), an optical system to guide, focus, and steer the light beam, and some sort of position detector. Figure 2.2 shows this tweezer setup, with a quadrant photodiode for position sensing and an additional CCD camera with an independent light source. Although a single beam can trap a spherical particle in all three transversal degrees of freedom, for feedback cooling purposes the trapping beam can be split into three beams which are guided orthogonally into the trapping chamber to constrain the particle effectively. Feedback for particle cooling can be realized by modulating the power of the laser, creating an electric field via electrodes embedded into the chamber (applicable only for electrically charged particles), or other methods.

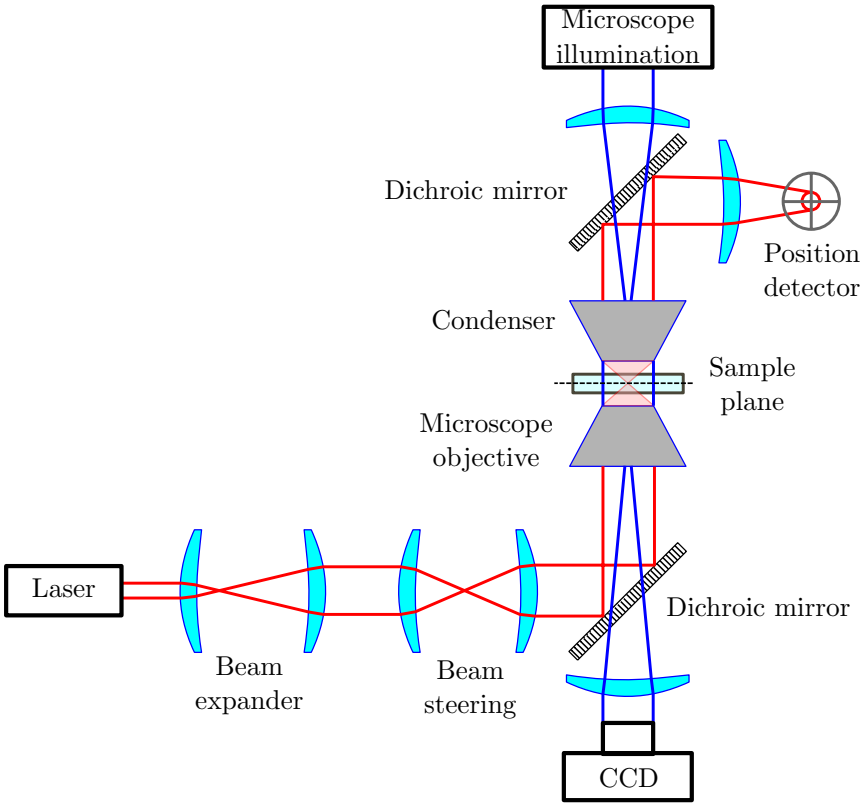


Figure 2.2: Generic optical tweezer schematic (adapted from [17])

2.2.2 Qualitative explanation of particle trapping

The forces acting on the particle by the light beam arise from the interaction between photons and the particle, particularly by transferring momentum from one to the other. This interaction is properly described by complex electromagnetic theory, but reasonably accurate approximations can be made depending on the relative scale of the particle compared to the light beam wavelength. For particles significantly smaller than the wavelength, the particle can be approximated as a dipole in an electromagnetic field created by the light beam. On the other side of the spectrum, for particles significantly larger than the wavelength, the geometric optics approach can be used. Although the latter method is less accurate than the electromagnetic one due to the particle scale [15, 14], it is conveniently intuitive and thus will be used in this subsection.

There are two main forces caused by the light beam with a Gaussian intensity profile that act on an object in an optical trap: radiation pressure (sometimes called the scattering force) and gradient force. The former acts in the direction of propagation of the beam, trying to push the particle downstream, while the latter creates a restoring force, which guides the particle towards the light beam's highest intensity.

The explanation of the scattering force is straightforward after accepting that light carries momentum; the light is being scattered from the surface of the particle and thus its momentum is changing. As a result of the law of action and reaction, the change in momentum exerts a force on the particle. The gradient force can be explained in a similar manner, although it requires the light beam to have an appropriate light intensity

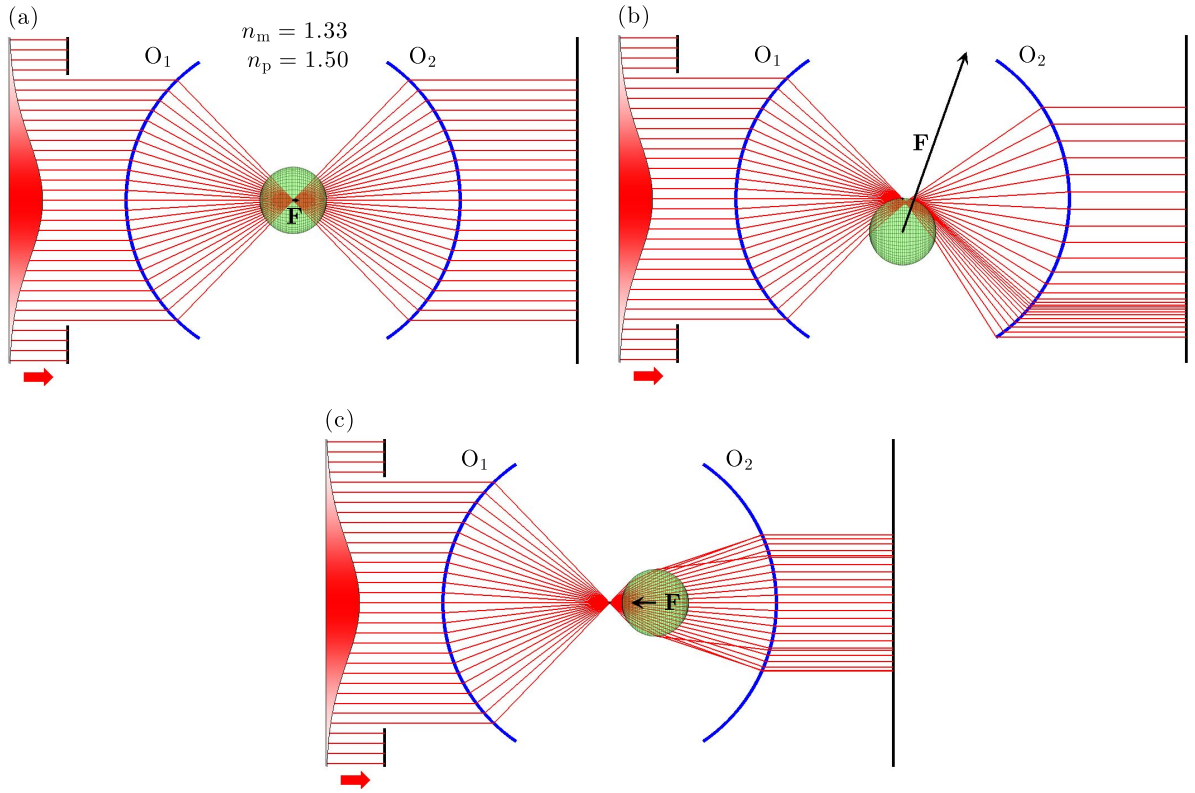


Figure 2.3: Geometrical approximation of particle in gaussian light beam (adapted from [18])

profile (e.g. Gaussian) and to be convergent (see Figure 2.3a). A spherical particle will be assumed, which can be viewed as a condenser lens. If the particle makes the convergent beam more convergent, the beam loses momentum in the direction of propagation, and this momentum is transferred to the particle, pushing it forward. Conversely, if the particle is past the focal point, it makes the divergent beam more convergent, that is, the beam has a higher momentum, and the reaction effect pulls the particle back to the focal point (see Figure 2.3c). Note that the gradient effect in the longitudinal direction can be significant enough to overcome the scattering force.

Transversal trapping (perpendicular to the direction of beam propagation) is explained analogously and is possible due to the Gaussian light intensity profile. If the particle is deflected from the center line, it again makes the beam more convergent. However, the beam has the highest intensity (and highest momentum) near the centerline, and thus the sum of sideways light momentum is pointing away from the centerline. Then, the reaction force on the particle pulls the particle back toward the centerline (see 2.3b).

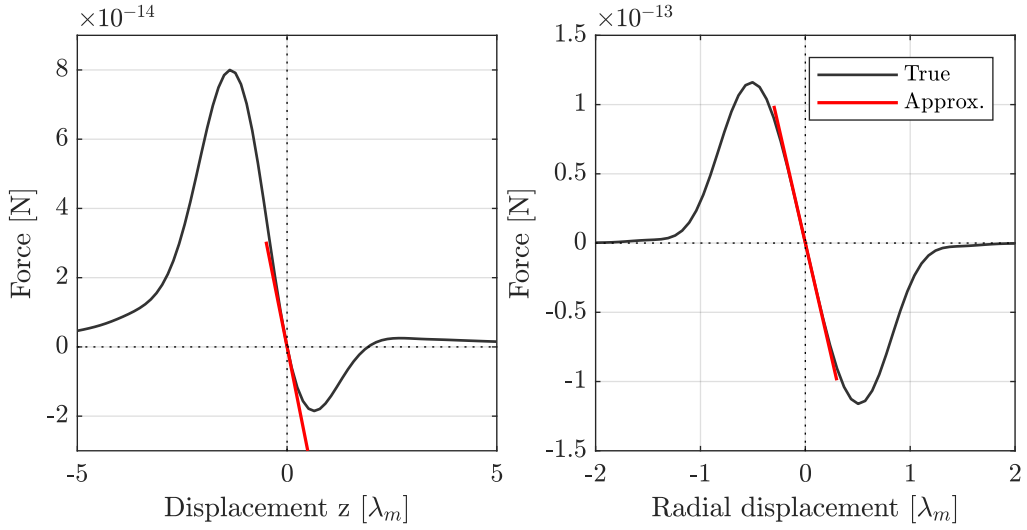


Figure 2.4: Non-linear gradient forces example with linear approximation (Created using [20])

2.2.3 Equation of motion

Leaving the momentum explanation used in the previous subsection, the gradient force will be expressed using the electric field of the Gaussian beam of light[2]:

$$\langle \mathbf{F}_{grad} \rangle = \frac{\alpha'}{2} \langle \nabla \mathbf{E}^2 \rangle \tag{2.9}$$

From equation 2.9, α' is the optical potential (real part of the complex polarizability[1]), $\mathbf{E}$ is the electric field, and ∇ is a gradient operator. Because the trapping beam has the Gaussian intensity profile, the resulting gradient force is non-linear with the displacement of the particle (see Figure 2.4) and its magnitude is also dependent on laser power (which in turn allows modulation of the "spring stiffness"). For small displacements q, the gradient force can be linearly approximated with a spring constant k_{grad} for each degree of freedom:

$$\langle F_{grad,q} \rangle = -k_{grad,q} \cdot q \tag{2.10}$$

It is evident from Figure 2.4, that the stiffness of the trap weakens after exceeding a certain threshold. In coordination with stochastic force impulses from the collisions with other particles in the medium, an impulse (or series of impulses) can occur which cannot be compensated by the trap, and then the particle is ejected from the trap and lost. This phenomenon is called Kramers escape [19].

As hinted earlier in Section 2.1.1, Equation 2.1, the next important effect arising from the interaction with the surrounding medium is the dissipative force that counteracts particle motion. For a spherical particle and Brownian motion in gas, the damping ratio γ_0 is expressed as follows [21, 2]:

[1]Polarizability is used to describe particle coupling to light and surrounding environment

$$\gamma_0 = 6\pi\eta R \cdot \frac{0.619}{0.619 + \mathrm{K_n}}(1 + c_K) \tag{2.11}$$

$$c_K = \frac{0.31\mathrm{K_n}}{0.785 + 1.152\mathrm{K_n} + {\mathrm{K_n}}^2} \tag{2.12}$$

Where η is the viscosity coefficient of the gas, R is the radius of the sphere, and $\mathrm{K_n}$ is the Knudsen number. The important quality is that the damping coefficient γ_0 of the equation 2.11 is proportional to the pressure, and thus it is possible to adjust the quality factor of the mechanical oscillator from overdamped to underdamped regime. Therefore, for constant pressure, the damping force is proportional only to the velocity of the particle:

$$\boldsymbol{F}_{damp} = -\gamma_0 \cdot \dot{\boldsymbol{q}} \tag{2.13}$$

Adding the restoring gradient force (Eq. 2.10) and the damping force (Eq. 2.13) to the Brownian motion equation 2.1, the Langevin equation describing the particle in the optical trap is obtained [15, 2, 21]:

$$\underbrace{m \cdot \ddot{\boldsymbol{q}}(t)}_{\text{Inertia}} + \underbrace{\gamma_0 \cdot \dot{\boldsymbol{q}}(t)}_{\text{Damping/dissipation}} + \underbrace{k_{grad}(q,t) \cdot \boldsymbol{q}(t)}_{\text{Restoring force}} = \underbrace{\sqrt{2k_\mathrm{B}T\gamma_0} \cdot \xi(t)}_{\text{Thermal fluctuation}} \tag{2.14}$$

In equation 2.14, m is the particle mass, γ_0 is the viscous damping coefficient from equation 2.11, k_{grad} is the effect of the light beam's gradient force (equation 2.9), k_B is the Boltzmann constant, T absolute temperature of the environment (i.e. surrounding medium) and $\xi(t)$ is a unitary white noise as described earlier. Notice that the constant γ_0 is contained in both the dissipation term and the driving fluctuating force - this is a result of the Fluctuation-Dissipation Theorem. The spring constant k_{grad} is written as a function of time t and displacement q as a reminder that it is nonlinear and can be modulated by laser power.

2.2.4 Feedback

Controlling particle's position and velocity is crucial for various experiments, e.g. cooling particles down to only a couple milli-Kelvins, or limiting the particle's amplitude and thus avoiding the nonlinear stiffness region [2]. Langevin equation 2.14 which describes the particle in the optical trap can be further modified by adding another term denoting the feedback force:

$$m \cdot \ddot{\boldsymbol{q}}(t) + \gamma_0 \cdot \dot{\boldsymbol{q}}(t) + k_{grad}(q,t) \cdot \boldsymbol{q}(t) = \sqrt{2k_\mathrm{B}T\gamma_0} \cdot \boldsymbol{\xi}(t) + \boldsymbol{u}_{fb}(t) \tag{2.15}$$

One of the methods to achieve the feedback force $\boldsymbol{u}_{fb}$ in an experiment is by inserting electrodes into the chamber and then controlling the intensity of the electric field, exerting the Coulomb force on the charged particle. The control signal is usually derived from the measured position and / or velocity of the particle, and some of the strategies used for feedback cooling are:

- **Frequency doubling** $u(t) = g_{fb} \cdot x \cdot \dot{x}$ [4]
- **Cold damping** $u(t) = g_{fb} \cdot \dot{x}$
- **Full state feedback** $u(t) = g_x \cdot x + g_{dx} \cdot \dot{x}$ [3]

The idea behind the frequency-doubling algorithm is straightforward: As the particle moves towards equilibrium, trap stiffness is weakened so as not to further accelerate the particle, and vice versa, stiffness is increased to pull the particle back after it moves away from the equilibrium. As mentioned before, stiffness modulation can be achieved by modulating the laser's power. The double-frequency signal can be generated by either a phase-locked loop (PLL) or product of position and velocity phase shifted by an appropriate amount. This method also allows feedback cooling in three axes with a single control input (the laser intensity). In that case, the weighed product $g_{fb} \cdot q \cdot \dot{q}, \quad q \in (x, y, z)$ for each axis is summed together, producing the control signal [4].

The next algorithm links the control signal to the velocity of the particle $u_{fb} = g_{fb} \cdot \dot{q}$, effectively lowering or increasing the damping γ_0. Methods of obtaining the velocity of the particle will be discussed later. Contrary to the damping due to collisions with the medium, this additional damping alters only the dissipation term, not the fluctuation term (stochastic force impulses), hence the *cold* damping [2]:

$$m \cdot \ddot{\boldsymbol{q}}(t) + (\gamma_0 + \gamma_{fb}) \cdot \dot{\boldsymbol{q}}(t) + k_{grad}(q, t) \cdot \boldsymbol{q}(t) = \sqrt{2k_{\mathrm{B}}T\gamma_0} \cdot \boldsymbol{\xi}(t) \tag{2.16}$$

Figure 2.5 demonstrates the effect of cold damping with stationary analysis. It is clear that increasing the feedback gain decreases the height of the resonance peak as a result of increased damping. Therefore, this method achieves limited attenuation for off-resonance frequencies.

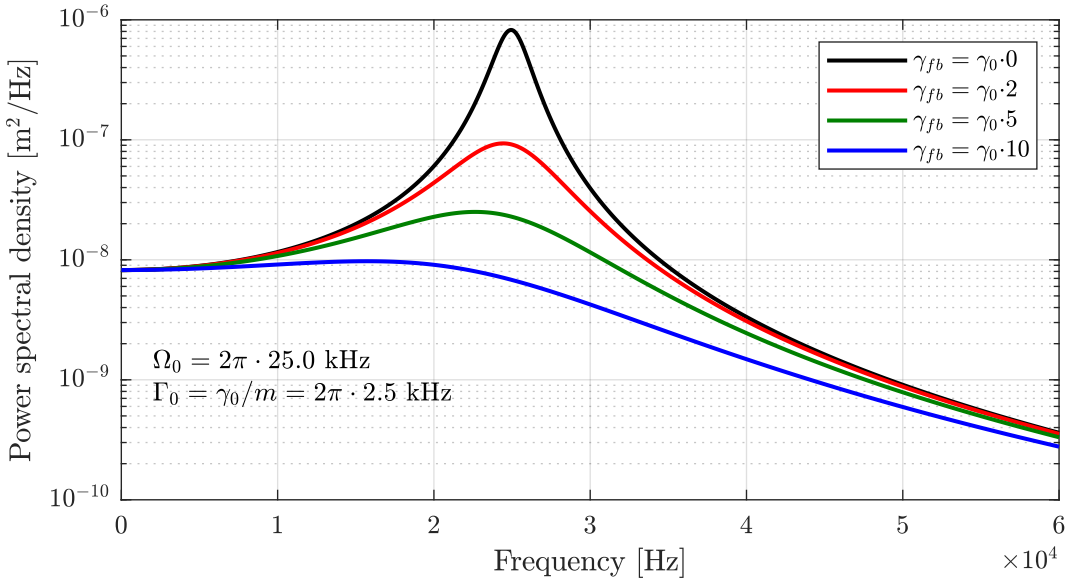


Figure 2.5: Cold damping effect on power spectral density

If the experimental setup (or subsequent signal processing) provides measurements

or estimates of both displacement and velocity, full-state feedback can be employed for particle cooling. There exists a wide array of methods for calculating the feedback gain, such as pole placement, or optimal control theory (e.g. the linear quadratic regulator). In particular, LQR can be used efficiently [3, 22] and will be presented in detail later. For the control signal calculated as a linear combination of the position and velocity of the particle $u_{fb} = g_q \cdot q + g_{dq} \cdot \dot{q}$, the Langevin equation can be modified as:

$$m \cdot \ddot{\boldsymbol{q}}(t) + (\gamma_0 + \gamma_{fb}) \cdot \dot{\boldsymbol{q}}(t) + (k_{grad}(q,t) + k_{fb}) \cdot \boldsymbol{q}(t) = \sqrt{2k_{\mathrm{B}}T\gamma_0} \cdot \boldsymbol{\xi}(t) \tag{2.17}$$

The intuition of the parameters γ_{fb} and k_{fb} is apparent from equation 2.17: the first parameter alters the damping of the system and the latter alters the natural frequency. The stationary analysis of a system with full state feedback is demonstrated in Figure 2.6.

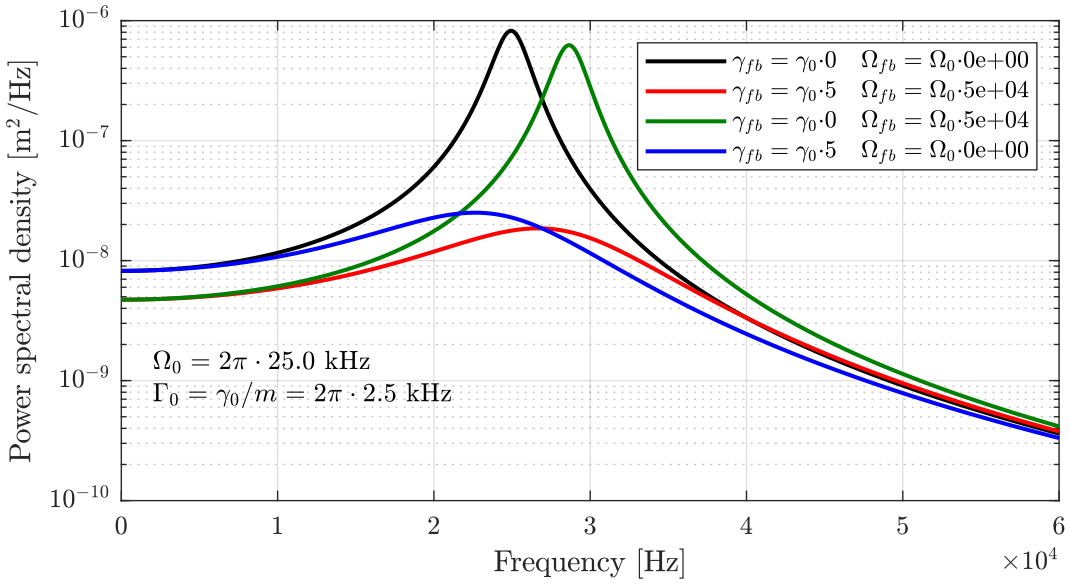


Figure 2.6: Full-state feedback effect on power spectral density

The energy of the particle (and consequently the effect of the feedback) is usually characterized by its center-of-mass (CM) temperature. Note that the CM temperature refers only to the motion of the particle, regardless of the actual temperature of the material. Without any feedback force, the particle temperature is equal to the surrounding medium (heat bath); however, by applying an appropriate feedback force, the CM temperature can be raised or lowered [4]:

$$T_{cm} = T_0 \frac{\Gamma_0}{\Gamma_0 + \Gamma_{fb}} \tag{2.18}$$

T_{cm} is new equivalent temperature after applying the feedback, T_0 is the temperature of the heat bath, $\Gamma_0 = \gamma_0/m$ is the damping coefficient divided by the mass of the particle, and Γ_{fb} is the damping change due to the feedback. From another perspective, if the

reduced displacement variance would have been solely a result of a heat bath, T_{cm} would be its temperature. This ratio can be obtained similarly as a ratio of integrals of the power spectral densities with and without feedback.

2.3 Approximate model of the particle

Following the previous section, the approximate linearized model of the particle in the optical trap is presented, and its limitations are discussed. Frequency domain and state-space representations with relevant transformations used throughout this thesis are also demonstrated. Finally, simulation of the particle in the optical trap is discussed.

2.3.1 Linearization

The linearization procedure was hinted in the previous subsection 2.2.3 and is summarized in this section. Several assumptions are important for the linearization:

- Particle is spherical and thus its orientation is irrelevant.
- Density, dimensions and electrical charge of the particle remain constant.
- Translational degrees of freedom are independent.
- Feedback force is linearly proportional to the control signal and independent of the position of the particle.
- Particle stays near the equilibrium in the linear stiffness region.
- Pressure and temperature of the surrounding gas remain constant.

The Langevin equation 2.15 linearized under mentioned assumptions conveniently corresponds to the well-known mass-spring-damper system driven by white noise:

$$m \cdot \ddot{q}(t) + \gamma_0 \cdot \dot{q}(t) + k_0 \cdot q(t) = \sqrt{2k_\mathrm{B}T\gamma_0} \cdot \xi(t) + c_{fb} \cdot u_{fb}(t) \tag{2.19}$$

Dividing the equation 2.19 by mass of the particle m produces:

$$\ddot{q}(t) + \Gamma_0 \cdot \dot{q}(t) + \Omega_0^2 \cdot q(t) = 1/m \cdot \sqrt{2k_\mathrm{B}T\gamma_0} \cdot \xi(t) + C_{fb} \cdot u_{fb}(t) \tag{2.20}$$

Here, the damping constant $\Gamma_0 = \gamma_0/m$, the natural frequency $\Omega_0^2 = k_0/m$ (k_0 is the linear spring constant), and $C_{fb} = c_{fb}/m$ is the feedback constant ($\frac{\mathrm{N}}{\mathrm{V \cdot kg}}$ for $u_{fb}(t)$ in Volts). The frequency domain representation can be obtained by splitting the Langevin equation into deterministic and stochastic parts and applying the Fourier transform.

The frequency representation of white noise $\xi(t)$ is unity independent of frequency $\xi(i\omega) = 1$ [7] scaled by the constant term, resulting in power spectral density $S_\xi(\omega) = 2k_\mathrm{B}T\gamma_0$. The square of the Fourier transform of the deterministic part of equation 2.20 S_0, combined with the PSD of driving noise S_ξ, results in the power spectral density for position q [21]:

$$S_q(\omega) = S_\xi(\omega) \cdot S_0(\omega) = \underbrace{2k_\mathrm{B}T\gamma_0}_{S_\xi(\omega)} \underbrace{\frac{1/m}{(\Omega_0 - \omega^2)^2 + \omega^2\Gamma_0^2}}_{S_0(\omega)} \tag{2.21}$$

2.3.2 State space and transformations

State space form is a common method for describing a system modeled by a set of ordinary differential equations (ODE) by transforming them into a set of first-order ODEs [23]. The common notation for continuous-time state-space form is as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{2.22}$$

Where $\mathbf{x}$ and $\dot{\mathbf{x}}$ are the vector of n states and the vector of their derivatives, respectively, $\mathbf{u}$ is the vector of m inputs, $\mathbf{y}$ is the vector of r outputs, $\mathbf{A}(n \times n)$ is *state matrix*, $\mathbf{B}(n \times m)$ is *input matrix*, $\mathbf{C}(r \times n)$ is *output matrix*, and $\mathbf{D}(r \times m)$ is *direct transmission matrix*.

The deterministic part of the equation 2.20 can be converted into the state space representation for state vector $\mathbf{x} = [q, \dot{q}]^T$ consisting of displacement and velocity, respectively:

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{\gamma_0}{m} \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ C_{fb} \end{bmatrix} u_{fb} = \begin{bmatrix} 0 & 1 \\ -\Omega_0^2 & -\Gamma_0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ C_{fb} \end{bmatrix} u_{fb} \tag{2.23}$$

The state space form also allows for straightforward linear transformation of state variables and relevant matrices, which can be utilized to improve numerical properties of the model or to transform states to a more convenient representation for the specific application. Starting with the relation of the transformed and original state vectors $\mathbf{x} = \mathbf{T}\bar{\mathbf{x}}$, where $\mathbf{T}$ is a non-singular matrix with dimensions $n \times n$, the relevant transformed matrices are [23]:

$$\dot{\bar{x}}(t) = \bar{\mathbf{A}}\bar{\mathbf{x}}(t) + \bar{\mathbf{B}}\mathbf{u}(t), \quad \mathbf{y}(t) = \bar{\mathbf{C}}\bar{\mathbf{x}}(t) + \mathbf{D}\mathbf{u}(t) \tag{2.24}$$

$$\bar{\mathbf{A}} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \quad \bar{\mathbf{B}} = \mathbf{T}^{-1}\mathbf{B}, \quad \bar{\mathbf{C}} = \mathbf{C}\mathbf{T}$$

Note that the input $\mathbf{u}(t)$ and the output $\mathbf{y}(t)$ remain the same after transformation, but the state variables are different. Transformation in equation 2.24 allows *normalization* of the harmonic oscillator state-space representation (eq. 2.23) such as:

$$\bar{\mathbf{q}} = \mathbf{T}^{-1}\mathbf{q} = \begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\Omega_0} \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} q \\ \frac{\dot{q}}{\Omega_0} \end{bmatrix} \tag{2.25}$$

This transformation normalizes the states (formerly displacement and velocity) by dividing the velocity by natural frequency Ω_0, which makes its magnitude similar to displacement. Advantages of this transformation will be evident later in the implementation of the Kalman filter. The rest of the state space model is for transformation matrices 2.26 as follows:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & \Omega_0 \end{bmatrix}, \quad \mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\Omega_0} \end{bmatrix} \tag{2.26}$$

$$\begin{bmatrix} \dot{\bar{q}}_1 \\ \dot{\bar{q}}_2 \end{bmatrix} = \begin{bmatrix} 0 & \Omega \\ -\Omega & -\Gamma_0 \end{bmatrix} \begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{C_{fb}}{\Omega_0} \end{bmatrix} u_{fb} \tag{2.27}$$

Although continuous-time state-space representation is useful on its own, discrete-time representation is often required for the simulation, derivation, and implementation of signal processing and control algorithms. The state-space model sampled at $k, k+1, ...$ separated by the time interval ΔT can be written as:

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k, \quad \mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{E}\mathbf{u}_k \tag{2.28}$$

Discrete-time matrices $\mathbf{F}$ and $\mathbf{G}$ are then obtained through [24]:

$$\mathbf{F} = e^{\mathbf{A}\cdot\Delta T}, \qquad \mathbf{G} = \left(\int_{\tau=0}^{\Delta T} e^{\mathbf{A}\cdot\tau}\, d\tau \right) \mathbf{B} = \mathbf{A}^{-1}(\mathbf{F} - \mathbf{I})\mathbf{B} \tag{2.29}$$

The matrices $\mathbf{H} = \mathbf{C}$ and $\mathbf{E} = \mathbf{D}$ remain the same as in the continuous-time state-space model.

2.3.3 Simulation

The difficulty of simulating a particle in the optical trap depends on the degree of complexity required, as any other simulation. This thesis does not cover particle-to-particle interaction or complex fluid dynamics (e.g. [25]), so the simulations used focused only on the macroscopic effects of the surrounding medium on the particle itself, that is, the damping and fluctuation of random force.

$$m \cdot \ddot{q}(t) + \gamma_0 \cdot \dot{q}(t) + k_0 \cdot q(t) = \sqrt{2k_\mathrm{B}T\gamma_0} \cdot \xi(t) + c_{fb} \cdot u_{fb}(t) \tag{2.30}$$

This simplified scenario is described by the Langevin equation 2.30, which describes a damped harmonic oscillator driven by white noise and force feedback. There are two important time scales: $\tau_h = \gamma_0/k_0$ (damping coefficient divided by trap stiffness) related to the oscillatory behavior due to the restoring force and $\tau_p = m/\gamma_0$ (particle mass divided by damping coefficient) describing the transition from diffusive Brownian motion to ballistic Brownian motion [26, 21]. These time scales are comparatively very different; the effect of the restoring force is "slower" than random fluctuations ($\tau_h >> \tau_p$).

The main difference between the diffusive and ballistic regimes is that, for diffusive motion, the inertia of the particle can be neglected due to the high number of collisions between observations. This effectively eliminates any correlation of successive displacement observations ($R_q(\tau) = \delta(\tau)$ for the sample rate $\Delta t >> \tau_p$). However, for a sample rate comparable to or less than the inertia relaxation time τ_p, the inertial term cannot be neglected. Intuitively, the inertia of the particle impedes an instantaneous change of momentum; see Figure 2.7 for visual explanation.

Thermal white noise $\xi(t)$ is generated as a sequence of random numbers with Gaussian distribution, which is a feature available in most simulation software. However, to simulate the system in discrete time and provide power to the system accurately according to

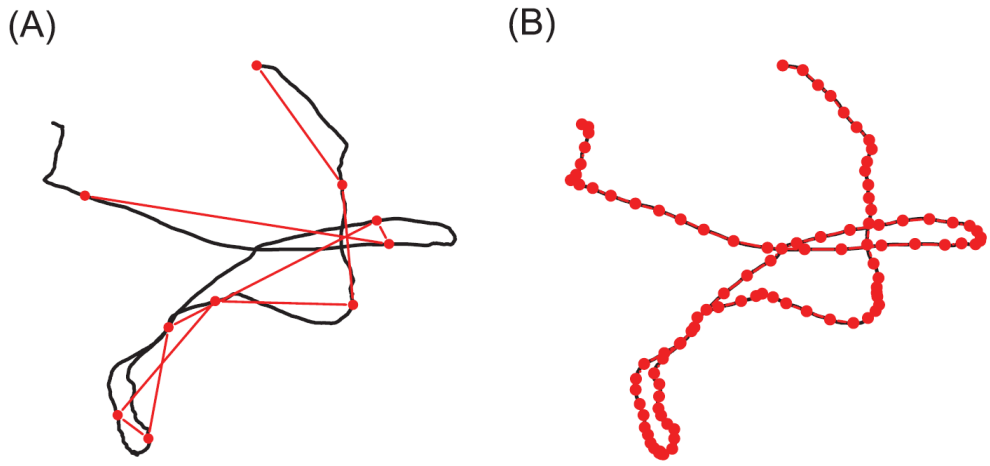


Figure 2.7: A 2D trajectory of Brownian particle: The black trace is true trajectory of the particle, the red dots represent observations at different sample rates (Δt A = $10\Delta t$ B) [21]

equation 2.30, $\xi(t)$ must be scaled by $1/\sqrt{\Delta t}$. The finite difference equation is then as follows [26]:

$$m\frac{q_i - 2q_{i-1} + q_{i-2}}{(\Delta t)^2} = -\gamma_0\frac{q_i - q_{i-1}}{\Delta t} - k_0 \cdot q_i + \sqrt{2k_\mathrm{B}T\gamma_0}\frac{1}{\sqrt{\Delta t}} \cdot \xi_i \tag{2.31}$$

2.4 Signal processing

Digital signal processing constitutes an important element of measurement and control systems and is used to extract important information from real-world signals that are often noisy. This section primarily covers the implementation of filters with infinite impulse response (IIR), the intricacies of discrete downsampling, and the Kalman filter.

Kalman filter provides full-state estimation, which can be readily utilized for feedback control. One of such controllers is the linear-quadratic regulator (LQR), which is presented in the last subsection.

2.4.1 Filters with infinite impulse response

Contrary to filters with finite impulse response (FIR), filters with an infinite impulse response (IIR, sometimes called *recursive filters*) provide better performance with comparable computational power, although FIR filters can achieve overall better time and frequency responses [27]. Due to the requirements for particle stabilization, mainly the high sampling frequency (in the megahertz range) and low latency, IIR filters are more suitable.

The Direct form I of the nth-order IIR filter sampled at time $k \cdot \Delta T$ is calculated as:

$$\begin{aligned} y(k) = b_0 \cdot x(k) + b_1 \cdot x(k-1) + ... + b_n \cdot x(k-n) + \\ + a_1 \cdot y(k-1) + a_2 \cdot y(k-2) + ... + a_n \cdot y(k-n) \end{aligned} \tag{2.32}$$

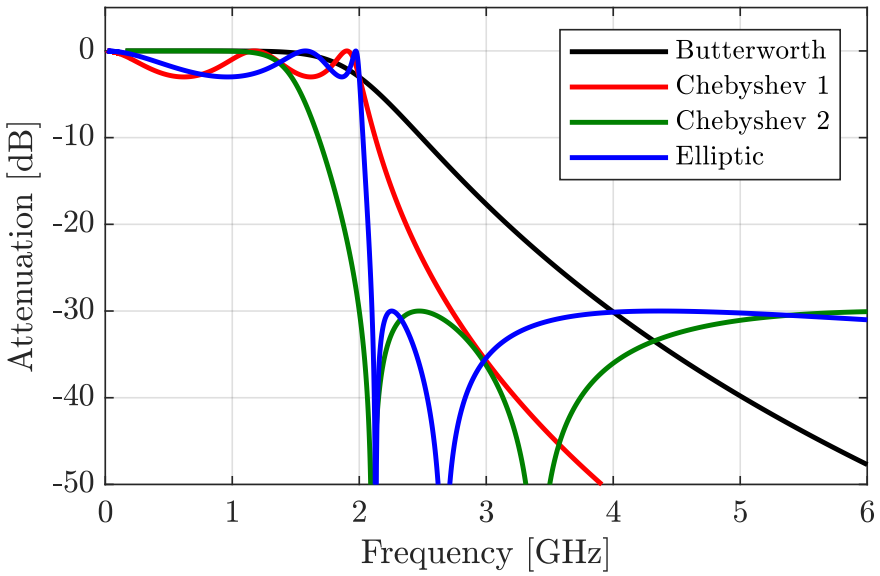


Figure 2.8: Comparison of IIR lowpass filters (cutoff frequency 2GHz) designed by various methods in Matlab [30]

Where b_i are *feedforward* coefficients, a_i are *feedback*[2] coefficients, $x(k-i)$ is the input value delayed by i samples and $y(k-i)$ is the previous output value delayed by i samples. Equation 2.32 can be expressed in the form of a transfer function in the s-domain [28]:

$$H(s) = \frac{B(s)}{A(s)} = \frac{b_0 s^n + b_1 s^{n-1} + ... + b_n}{a_0 s^n + a_1 s^{n-1} + ... + a_n} \tag{2.33}$$

The frequency response of the IIR filter can be tuned to arbitrary configurations, and there exists a wide array of methods and precalculated coefficients that yield filters with specified characteristics such as stopband attenuation, passband width, ripple in both stopband and passband, etc.

Butterworth filter [29] offers a maximally flat passband and attenuation of 20 decibels per decade per filter order. This means that the amplitude of the signal that contains frequencies that should pass through the filter is minimally distorted, and the frequency components outside of this passband are attenuated (see Figure 2.8).

After designing appropriate filter coefficients, the implementation of high-order IIR filters in fixed-point hardware can pose problems due to the large numerical range, leading to overflows, frequency response distortion, and even instability [28]. The numerical range of coefficients can be improved by dividing the nth-order filter into s = n/2 second-order filters that are cascaded in series, producing coefficients that are relatively closer in magnitude and thus require less computational resources.

Direct form I of the IIR filter according to 2.32 can perform in an unsatisfactory way on fixed-point arithmetic hardware, which can be improved by using the transposed Direct form II, depicted in Figure 2.9.

[2]Beware that various literature and software occasionally swap the b/a notation

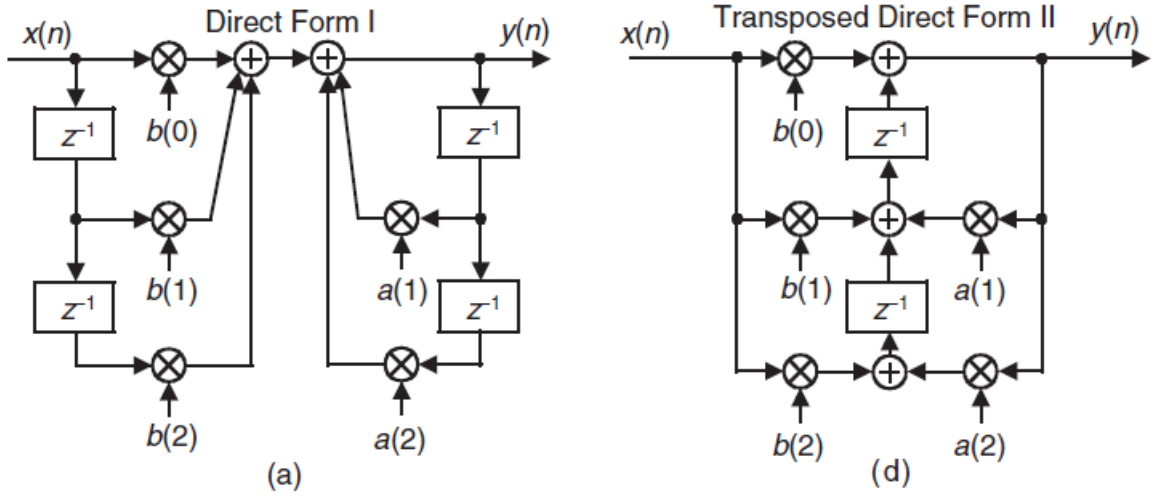


Figure 2.9: IIR filter in Direct form I and Transposed Direct form II (adapted from [28])

2.4.2 Downsampling

Issues related to downsampling are closely related to the aliasing phenomenon (Nyquist-Shannon theorem), which establishes a necessary condition for discrete sampling of continuous signals - the sampling frequency must be at least double of the highest frequency component of the sampled signal [27]. The analog anti-aliasing filter is an inseparable component of any reliable analog-digital converter (ADC), which should sufficiently attenuate any frequency components over the Nyquist frequency.

Therefore, additional anti-aliasing filtering should be performed before downsampling digital signals. This process can be conveniently combined using Cascaded Integrator-Comb (CIC) filters, which are a form of *recursive running sum* filters [28].

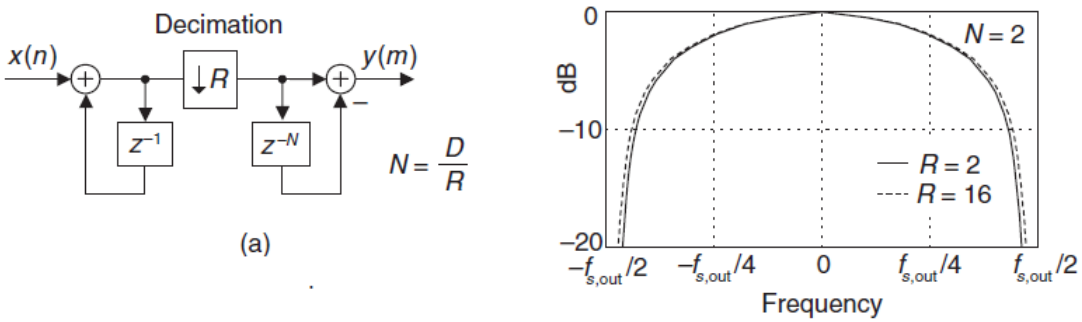


Figure 2.10: Single-stage CIC decimation filter block scheme and frequency response (adapted from [28])

As is evident from Figure 2.10, this filter consists of the integrator stage (running sum) and the delayed feedforward (*comb*) stage with decimation stage in between. Decimation means that only the n-th sample is passed through (sampled), and the rest is discarded, lowering the sampling frequency by a factor of n. Although the frequency response of

CIC filters is inferior to either FIR or IIR filters, implementation is very cheap in terms of computational time and/or resources (only two summations per stage). Multiple stages can be stacked in series to improve stopband attenuation.

2.4.3 Kalman filter

Kalman filter is a state estimation algorithm derived for the estimation of hidden[3] states based on the measurement of some of the states and the relevant state-space *process model*. A typical example can be the estimation of motor angular velocity from position readings (encoder signal). Knowledge of these states is crucial for many applications, including full-state feedback control. The standard Kalman filter assumes a linear time-invariant system with additive Gaussian *process noise* $\mathbf{w}$ affecting states and linear measurement with additive Gaussian *measurement noise* $\mathbf{v}$:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \end{aligned} \tag{2.34}$$

The algorithm alternates two steps, prediction and update. The prediction for the next timestep is based on the defined state-space model, while the update step corrects the prediction with the current measured values of the observed states. The mathematical description of this process is described in Table 2.1.

Predict	
$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u}$	Evolve states for one timestep, obtaining state prediction $\bar{\mathbf{x}}$ (also called *prior*)
$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^\mathbf{T} + \mathbf{Q}$	Predict state covariance $\bar{\mathbf{P}}$ through the state matrix $\mathbf{F}$ and *process noise covariance* matrix $\mathbf{Q}$
Update	
$\mathbf{e} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}}$	Calculate residual $\mathbf{e}$ (also called *innovation*) of measurement $\mathbf{z}$ and observation of internal state $\bar{\mathbf{x}}$ through the observation matrix $\mathbf{H}$
$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^\mathbf{T}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathbf{T} + \mathbf{R})^{-\mathbf{1}}$	Update Kalman gain $\mathbf{K}$ using state covariance matrix $\bar{\mathbf{P}}$, observation matrix $\mathbf{H}$ and *measurement noise covariance* matrix $\mathbf{R}$
$\mathbf{x} = \bar{\mathbf{x}} - \mathbf{K}\mathbf{e}$	Update state $\mathbf{x}$ (also called *posterior*) with product of Kalman gain $\mathbf{K}$ and residual $\mathbf{e}$
$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$	Update state covariance matrix $\mathbf{P}$ of measurement $\mathbf{z}$ and observation of state $\bar{\mathbf{x}}$ through the observation matrix $\mathbf{H}$

Table 2.1: Kalman filter formulas and description (adapted from [31])

To design the Kalman filter, the state-space model of the observed system is required,

[3]States that are not measured directly

as well as the covariance matrix $\mathbf{Q}$ of the process noise $\mathbf{w}$ and the covariance matrix $\mathbf{R}$ of the measurement noise $\mathbf{v}$. The design of the $\mathbf{R}$ matrix is usually straightforward because its evaluation can be easily made from a series of static measurements. The other matrix $\mathbf{Q}$, representing additional excitation of states or general uncertainty in the system, is often a tuning engineering factor [31]. The relative magnitude of the matrices $\mathbf{Q}$ and $\mathbf{R}$ determines the degree of confidence in the model or the measurement (the smaller $\mathbf{Q}$, the greater confidence in the model and vice versa).

If there is a known source of process noise, such as thermal fluctuations of gas in the optical trap, the process noise covariance matrix can be determined analytically. In this thesis, two methods for computing the $\mathbf{Q}$ matrix were explored, differing in the assumption whether noise $\mathbf{w}$ stays constant during the sampling period (*discrete* or *piecewise noise*), or can vary during the sampling period (*continuous noise*).

Using the first option in the form of discrete noise, the process noise matrix is obtained by [31]:

$$\mathbf{Q} = \mathbf{\Gamma}\sigma_w^2\mathbf{\Gamma}^T \quad \text{for system } \mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{w}_k \tag{2.35}$$

The matrix $\mathbf{\Gamma}$ is the discrete-time projection of process noise $\mathbf{w}$ into the state space and can be (but might not be) the same as the input matrix $\mathbf{G}$.

The other approach (*continuous noise*) does not have any additional assumptions about the process noise apart from Gaussianity, decorrelation in time, etc. Van Loan's method of matrix exponential [32] represents a convenient way to discretize the state-space model and obtain the matrix $\mathbf{Q}$ at the same time. First, the matrix $\mathbf{\Lambda}$ must be constructed using a continuous-time state matrix $\mathbf{A}$, a continuous-time projection of process noise $\mathbf{W}$, a spectral density[4] of process noise Φ_w (or S_w) and discrete sample time ΔT, resulting in a matrix with dimensions $2n \times 2n$ (for n states) [10]:

$$\mathbf{\Lambda} = \Delta T \left[\begin{array}{c|c} -\mathbf{A} & \mathbf{W}\mathbf{\Phi_w}\mathbf{W^T} \\ \hline \mathbf{0} & \mathbf{A^T} \end{array}\right] \tag{2.36}$$

The exponential matrix of the $\mathbf{\Lambda}$ matrix is taken and the discrete-time matrices $\mathbf{F}$ and $\mathbf{Q}$ are finally obtained by extracting the submatrices and manipulating them as follows:

$$\exp(\mathbf{\Lambda}) = \left[\begin{array}{c|c} \varnothing & \varnothing \\ \hline \mathbf{\Theta} & \mathbf{F^T} \end{array}\right], \quad \mathbf{Q} = \mathbf{F} \times \mathbf{\Theta} \tag{2.37}$$

According to available sources [31, 10, 33], there is no easy answer to the question of which noise model to use. Best practice is to test both, if possible, and evaluate their performance in the specific problem.

Evaluation of a real-world implemented Kalman filter is difficult due to the obvious lack of perfect knowledge of the underlying process, with only imperfect measurements available. However, because the Kalman filter is an optimal mean-square error estimator, it is possible to leverage the innovation sequence (the difference between the state estimate and the actual measurements) to assess the consistency and performance of the filter [34].

[4]For white noise this is the *height* of the power spectral density, which is constant

Ideally, the Kalman filter should eliminate only additive measurement noise, and thus provide a perfect estimate of the underlying state affected by the process noise. If measurement noise is white (Gaussian), then the innovation sequence should also be Gaussian, and its covariance should be exactly the same as the initial design estimate, no more, no less.

Compared to other filtering algorithms, the greatest strength of the Kalman filter lies in the possibility of retrieving hidden states, which is superior to other methods, such as numerical differentiation. The frequency response and phase delay are also generally superior to model-free methods such as FIR or IIR filters, although it is not an entirely fair comparison. The main drawbacks are the requirement of a well-defined state-space model and noise analysis, and the implementation is not as straightforward as with the other methods.

2.4.4 Linear-quadratic regulator

Linear-quadratic regulator (LQR) is a common representative of optimal control theory and offers a method of tuning a full-state regulator according to a specified cost function. The user then tunes the closed-loop dynamics intuitively by manipulating the cost of states and control effort, contrary to the pole placement method, where the user achieves desired dynamics by specifying the position of poles in a complex plane. The state-space system with full-state feedback regulator $\mathbf{K}$ is written as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad \Rightarrow \quad \mathbf{u} = -\mathbf{K}\mathbf{x} \quad \Rightarrow \quad \dot{\mathbf{x}} = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} \tag{2.38}$$

LQR solves the problem of finding the feedback matrix $\mathbf{K}$ by optimizing the quadratic cost function J (eq. 2.39) on an infinite time horizon for the system specified by equation 2.38 [35]. The matrix $\mathbf{Q}_x$ represents the cost (or penalty) of each state's deviation from zero, and $\mathbf{R}_u$ is the cost of the control input.

$$J = \int_0^\infty \left(\mathbf{x}^\mathbf{T}\mathbf{Q}_x\mathbf{x} + \mathbf{u}^\mathbf{T}\mathbf{R}_u\mathbf{u}\right)\, dt \tag{2.39}$$

Equation 2.39 is solved using the algebraic Riccatti equation, which is implemented in many software tools used in control engineering. For example, Matlab offers a simple command *lqr()*, which produces the optimal gain according to the specified state-space model and the cost matrices [36].

The intuition behind the matrix $\mathbf{Q}_x$ is obvious; the higher the cost, the greater the effort that should be exerted to bring the state to zero as soon as possible. The cost of control input $\mathbf{R}_u$ can be explained using a popular example of a rocket with limited fuel supply. The goal is to reach a desired state and use as little fuel as possible; this would be the scenario in which the relative magnitude of the matrix $\mathbf{R}_u$ is greater than $\mathbf{Q}_x$, which puts greater emphasis on fuel conservation. This analogy breaks down for systems with practically unlimited *fuel supply*, then $\mathbf{R}_u$ becomes a tuning factor that is set as low as possible while respecting practical limits of the system, such as input saturation, rate of change, etc.

2.5 System on chip FPGA

This section covers the use of field programmable gate arrays (FPGA) with respect to digital signal processing (DSP), mainly the special case of system-on-chip (SOC) FPGAs. The parameters of STEMlab Red Pitaya are presented, since this is the hardware used for the experiments, which are covered in later chapters. In the last subsection, remarks about implementation of DSP algorithms are discussed together with tools which aid such development.

2.5.1 Overview

The general architecture of field programmable gate arrays is that each chip offers multitudes of discrete logic operators, lookup tables, memories, multipliers, and others in a single package, which are later non-permanently connected to serve the specified function. This is in direct contrast to the popular microcontroller architecture, which executes instructions sequentially on highly optimized hardware and offers only limited configuration [37].

The architecture of FPGAs thus allows massive parallelization of algorithms, leading to high throughput and low latency. It also supports custom width arithmetic, further enhancing the optimization possibilities. These advantages are tainted by the difficulty of configuring (programming) FPGAs, which is more involved than, for example, microcontroller programming in C.

System on chip FPGAs (SoC) combine strengths of both FPGAs and microprocessors by integrating both into a single device. Close integration allows offloading some of the heavy computations to the programmable logic (PL) and leaving the rest, such as monitoring, communication, and other time-noncritical tasks, to the processor (PS).

2.5.2 Red Pitaya STEMlab 125-14

STEMlab 125-14 is a versatile SoC FPGA-based development kit designed and manufactured by Red Pitaya[5]. The heart of this kit is a Zynq-7010 by Xilinx [38], a SoC that combines an ARM Cortex-A9 dual core processor and an Artix-7 FPGA that are interconnected by AXI bus. Both the FPGA and the microprocessor are programmable separately, but only their synergy unlocks the full potential of SoC FPGAs.

AXI bus is a flexible bus with multiple protocol variants for different applications, e.g., AXI-Stream for continuous stream of data and simple flow control, AXI4 for address-based data transfer, or its simpler variant AXI4-Lite. Any of these variants allows the exchange of data between the FPGA (programmable logic, PL) and the ARM processor (PS), extending the capabilities of the entire system [39].

The Zynq SoC is supported by two analog-digital converters (ADC) and two digital-analog converters (DAC), both with resolution of 14 bits and clocked at 125 MHz. There are other onboard peripherals such as Ethernet port, Micro-USB connectors for power supply and serial communication, GPIO, etc. See the hardware documentation at [40] for more information. Some of the main parameters are summarized in Table 2.2.

The great advantage of Red Pitaya hardware is a formidable community working with the hardware, which pleasantly lowers the initial learning curve, similarly to Arduino in the world of microcontrollers. Notable mentions are Anton Potočnik [41], who wrote a

[5]Although Red Pitaya is the name of the manufacturer, it will be reffered to the STEMlab 125-14 simply as Red Pitaya (RP)

Zynq-7010 - Processor ARM Cortex A9	
Max. frequency	667 Hz
External memory support	DDR3, DDR3L, DDR2, LPDDR2
Interface w/ PL	2x AXI 32bit Master, 2x AXI 32 bit slave
Zynq-7010 - FPGA Artix-7	
Lookup tables	17600
Flip-Flops	35200
Block RAM	2.1 Mb (60 36Kb blocks)
DSP Slices	80 of 18×25 bit integer Multiplier-Accumulator blocks
Peripherals	
Fast ADC	2× 14 bit, 125 MSps, ±1 V or ±20 V (switchable via jumpers)
Fast DAC	2× 14 bit, 125 MSps, ±1 V
RAM capacity	512 MB DDR3
Ethernet bandwidth	1 Gbit
GPIO	16× input, output, or tristate
Slow ADC	4× 12bit, 0-3.5 V, 100 kSps
Slow DAC	4× 12bit, 0-1.8 V, 100 kSps

Table 2.2: STEMlab 125-14 parameters [40, 38]

comprehensive introduction to HDL programming for Red Pitaya, and Pavel Demin, who maintains a great library of Red Pitaya compatible HDL IP cores [42].

Red Pitaya developers have prepared the entire open-source ecosystem on top of the Arm processor (based on Ubuntu 18.04), which contains ready-made applications such as oscilloscope and signal generator, and provides simple API for developers to write more of such applications, without programming the FPGA itself.

2.5.3 Implementation of DSP algorithms

As briefly mentioned above, the implementation of digital signal processing algorithms (DSPs) in FPGAs is accompanied by many intricacies. While some are shared with microprocessor programming, such as fixed point arithmetic, some are unique to FPGAs, such as timing closure.

The clock and consequently timing are a critical concept in FPGA programming when implementing sequential logic, that is, logic that is time-critical and requires controllable propagation of signals. Contrary to combinatorial logic (mostly logical operations), sequential logic is driven by a clock signal that controls the data flow (sampling) between flip flops. The propagation of the signal through, for example, multiplier is not instantaneous and takes up to couple of nanoseconds. If the signal needs to propagate through several multipliers, the entire delay would be too great, and if the clock is too fast (too

high frequency), flip-flop would sample entirely wrong signal on the next clock edge.

Pipelining is a technique designed to alleviate such situations and is achieved by inserting additional flip-flops (or pipeline registers) into the signal path, serving as an intermediate data storage. This reduces the time until the signal becomes stable and also distributes the delay in physical paths (net delay). It also increases the latency because the signal can take several clock cycles until it propagates through the entire structure, but the high clock frequency and throughput can be maintained without compromising reliability. It depends on every situation to evaluate the trade-off between frequency and latency of the algorithm.

Most FPGAs do not support hardware floating point arithmetic, and although it is realizable from other logic blocks, it requires more resources and is slower [43]. Thus, to fully leverage the parallel architecture, the DSP algorithms must be implemented in fixed point. This means that the user must choose the appropriate bit width to avoid integer overflows and satisfy required precision while respecting timing and available resources.

Fortunately, modern software tools provide great assistance in these tasks. In this thesis, the Simulink HDL Coder and the Simulink Fixed Point Tool were used. The former provides automated Hardware Description Language (HDL) code generation from Simulink models, including automated pipelining and AXI interface generation. Pipeline stages can be inserted into signal paths to satisfy a specified target frequency, and the coder will also match the delays created by these additional registers with other parallel paths. Fixed point tool assists in conversion of Simulink models from floating point to fixed point arithmetic according to the ranges obtained during simulation. The resulting model can then be imported as an IP core into the development tool for FPGA programming, such as Xilinx Vivado.

3 FPGA implementation

As described in Section 2.5, the heart of Red Pitaya is the Zynq 7010 SoC, which integrates both the FPGA (programmable logic, PL) and the dual-core ARM processor (PS). Both sections are programmable separately and thus each can be used for the most suitable task. For example, the FPGA can perform specific time-critical tasks, such as feedback control, and the processor can communicate with the outside world simultaneously. One of the presented implementation workflows focuses on implementation of DSP algorithms only on the processor, using the FPGA only as an interconnection of the peripherals. The other workflows primarily utilize the FPGA for time-critical operations and the processor is used only as an interface for configuration.

The overview of the Red Pitaya implementation will begin with the usual first program - LED blinking, the *hello world* of embedded programming, followed by some implementation details regarding signal routing and processing in the FPGA. In the next sections, implementation of Butterworth and Kalman filters is discussed, with the last section describing the configuration interface.

3.1 Blinking LED

This program was used to evaluate the accessibility and performance of the selected workflows, which are the following.

- Red Pitaya C API
- Hardware description language (HDL) programming and IP Cores integration
- HDL code generation from Matlab Simulink

The first method uses the Red Pitaya-specific API for C programming, coupled with the provided FPGA configuration file (bitstream). Thus, the FPGA serves only as an interconnection and all signal processing must be performed by the ARM processor. Simple functions such as `rp_DpinSetState(RP_DIO0_N, RP_HIGH)` are then used to retrieve input signals and set output peripherals. See Appendix A.1 for the bare minimum code for LED blinking, which achieved a maximum switching frequency of approximately 2.3 MHz for a single digital output in an infinite loop.

HDL programming unlocks the full potential of the FPGA, but is also more difficult and requires a specific development tool for the synthesis, implementation, and bitstream generation steps. For the Red Pitaya with Xilinx FPGA, the development tool is Xilinx Vivado. It provides an intuitive graphical block design workflow, where the user places vendor-provided or third-party IP cores[1], custom HDL modules, and can graphically route connections between modules and the hardware interface.

[1]Intellectual property cores - Complete functional modules verified on the hardware which usually provide only limited configuration

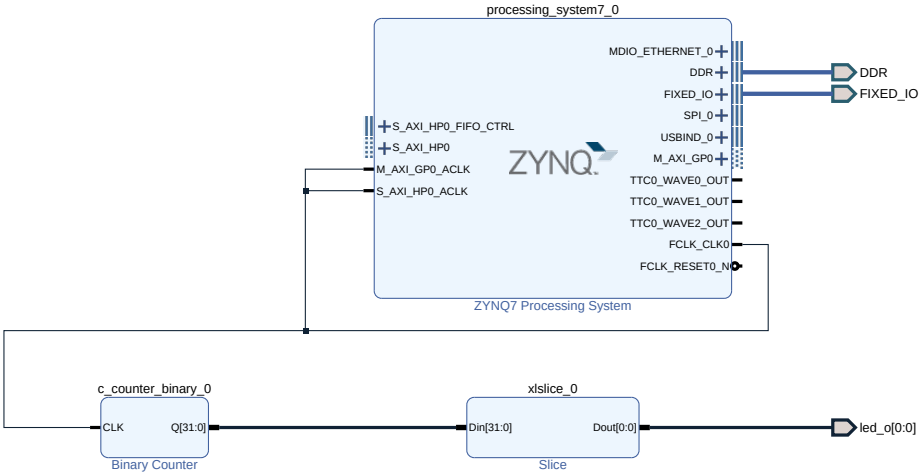


Figure 3.1: Block design for blinking LED example in Xilinx Vivado

The blinking LED block design can be easily realized using only *binary counter* and *bit slice* IP cores (see Figure 3.1). The counter increments by one on each clock cycle, and by extracting and connecting any bit of the counter value to the digital output, a blinking LED with a specific frequency can be constructed. Using the least significant bit allows the LED to blink at the clock frequency of the FPGA, which is typically 125 MHz for the Red Pitaya. This example is also the first one in the FPGA programming guide by Anton Potočnik [41].

Generating HDL (Verilog) code from the Simulink model yields similar results. The blinking LED can be simply modeled as a unit delay with bit inversion in the feedback loop (see Figure 3.2). The HDL Coder then converts the model into an intermediate Simulink model (which is identical to the original in this simple case) with added pipelining, delay balancing, and other operations to allow a successful conversion to HDL. From this altered model, the Verilog module is finally generated, which can be placed in the block design in Vivado. See Appendix A.2 for the generated Verilog code.

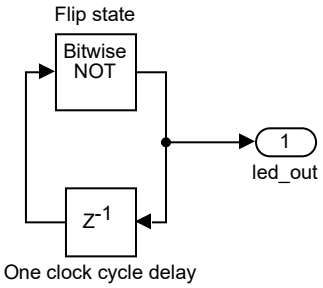


Figure 3.2: Simulink model for blinking LED at clock frequency

As expected, workflows leveraging FPGA programming achieve superior performance compared to the Red Pitaya API. In this case, the HDL-based workflows support LED switching frequency of 125 MHz, while the C-API only 2.3 MHz. This difference is further magnified in applications that allow for parallelization of some of the operations. Although C-API offers a comfortable way to interact with the Red Pitaya peripherals, it would pose a significant bottleneck in performance, and thus the HDL Coder workflow was used to

implement DSP algorithms themselves. These generated cores are supported by other custom Verilog modules for various data handling.

3.2 Data interface

This sections covers methods used to read data from the analog-digital converter, process it, and also provide necessary signals to the digital-analog converter. Modules required for recording of long time-traces are presented, as well as methods used to resolve clock-domain crossing.

3.2.1 A/D and D/A converter

Red Pitaya is equipped with Analog Devices LTC2145CUP-14 analog-digital converter clocked at 125 Mhz, featuring dual channel simultaneous sampling, 14-bit resolution and parallel digital data output [44]. Before sampling the analog signal, it passes through a hardware anti-aliasing filter with bandwidth approx. 50 Mhz. ADC interface in the FPGA then appears as two 14-bit wide vectors (one for each channel) with an associated differential clock input. The raw ADC input must be calibrated for correction of DC offset and also converted to the 2's complement format used for further signal processing (see Figure 3.3). Pavel Demin's AXIS ADC IP Core [42] was used as a basis and modified with offset and overflow saturation. The offset value for both channels is provided to the module input ports and is configurable from the processor through an AXI memory-mapped interface. ADC module also provides a clock signal for DSP algorithms to avoid clock domain crossing.

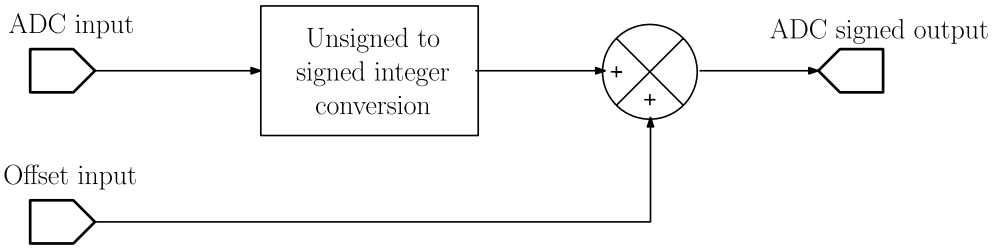


Figure 3.3: ADC calibration and data type conversion module schematic

The output of analog signals is facilitated by the Renesas DAC1401D125 digital analog converter [45], which features a sample rate of 125 Msps and an output resolution of 14 bits. DAC part is fed with unsigned interleaved dual-channel digital data, which means that the desired output value for channels A and B is alternating on a single signal path along with an appropriate signal bit. Again, Pavel Demin [42] provides a tested core that performs the necessary data conversions and routing.

3.2.2 Memory interface

There are several options for recording a relatively large amount of data at fast rate: constructing registers directly in the FPGA fabric from logic resources, using Block random access memory (BRAM), or using double data rate (DDR) RAM connected to Zynq. The first option is very ineffective and consumes the logic resources required for the implementation of DSP algorithms. The second option is more viable, but offers only up to 2.1 Mb of storage space. Red Pitaya is equipped with 4 Gb of RAM, with a 32bit wide

interface. This means that values of both ADC channels and some additional bits of data can be saved on each clock cycle.

Xilinx provides a direct memory access (DMA) IP core [46] that, in conjunction with the Zynq7 Processing system (PS7) IP core, controls most of the required signalling at the low level and greatly simplifies the RAM interface. See Figure 3.4 for a schematic overview of the interconnection of the mentioned modules.

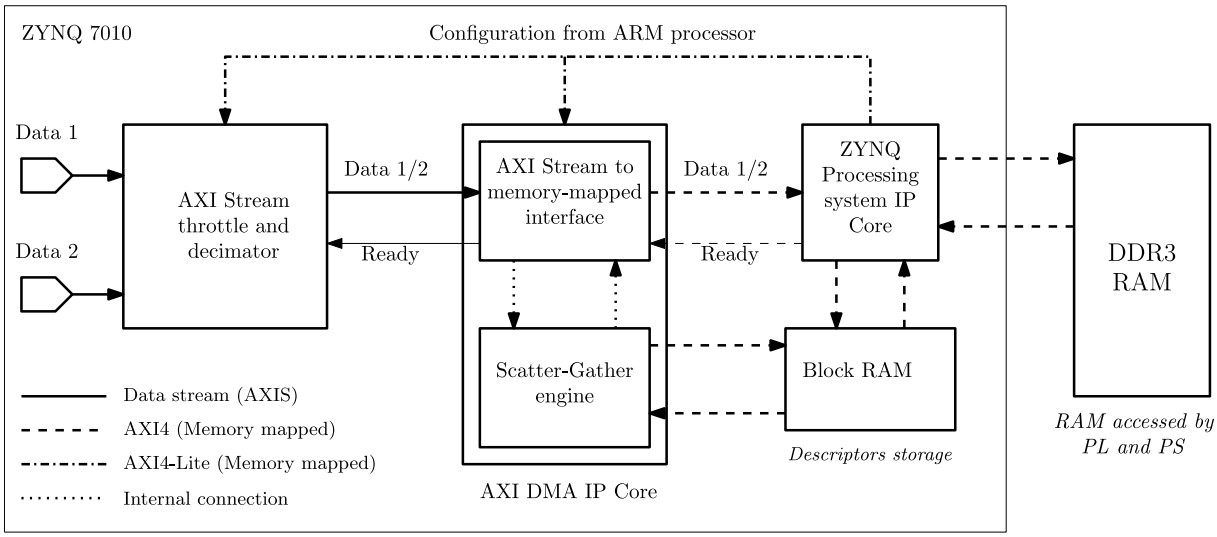


Figure 3.4: Schematic connection of modules required to save continuous data stream (e.g. ADC, or filter output) to RAM

DMA module requires data input through the AXI Stream (AXIS) interface (signal definition in Table 3.1). DMA allows a maximum packet length of 2^{26} bytes ($\doteq$ 67 MB), which is approximately 1/8 of the available RAM, hence the Scatter-Gather Engine (SGE) must be set up, which supports DMA to achieve longer uninterrupted packet lengths. SGE provides the DMA information about requested data transfers in the form of descriptors, data structures saved in BRAM (directly in the FPGA fabric). The descriptors contain the start address (first data frame location), the packet length, the address of the next descriptor, and control bits. These are configured prior to the start of the data transfer according to the information sent from the Matlab app, which specifies the sample rate and the amount of samples to record.

The throttle module (see Fig. 3.4) was developed to convert the continuous data stream from the ADC and DSP algorithms to the packet-based AXI Stream, which means that it must assert `TVALID` and `TLAST` and monitor the `TREADY` from DMA. This module is also used to interleave two 32-bit wide signals, so ADC channels and other data can be recorded simultaneously, albeit on half of the original frequency (125/2 = 62.5 MHz). Another feature is the data decimation used to record only every n-th sample, effectively dividing the sampling frequency by n but increasing the length of the record in time.

3.2.3 Clock domain crossing

Clock domain crossing (CDC) occurs whenever a signal transitions from one clock region to another. In this case, CDC occurs before the DMA throttle (ADC clock to FPGA clock) and on the boundary of the lower rate DSP algorithm (see Fig. 3.5). HDL Coder is fortunately able to resolve multirate designs and automatically generates necessary logic

32 bit AXI Stream (Master-Slave)		
Signal name [Bit width]	**Description**	**Direction**
TDATA [32]	Vector of data/payload	M→S
TVALID [1]	Bit indicating data validity	M→S
TLAST [1]	Bit indicating last data frame	M→S
TKEEP [4]	Bits indicating which bytes are valid in the particular data frame	M→S
TREADY [1]	Bit indicating if slave is ready to receive data	S→M
M = Master, S = Slave		

Table 3.1: Description of AXI Stream signals required by the DMA IP core [39]

to ensure reliable transmission from the ADC clock to the DSP clock.

Crossing clock domains with differing phase is solved by 16 words deep Asynchronous FIFO (First in, first out) buffer (Xilinx IP Core), which is able to bridge this crossing reliably, avoiding signal metastability. Because this FIFO can introduce up to 16 clock cycles of latency, clock domains are crossed only for recording the signal into RAM, where the latency is irrelevant. Thus, all latency-critical signal processing and DAC outputs are coupled to the ADC clock.

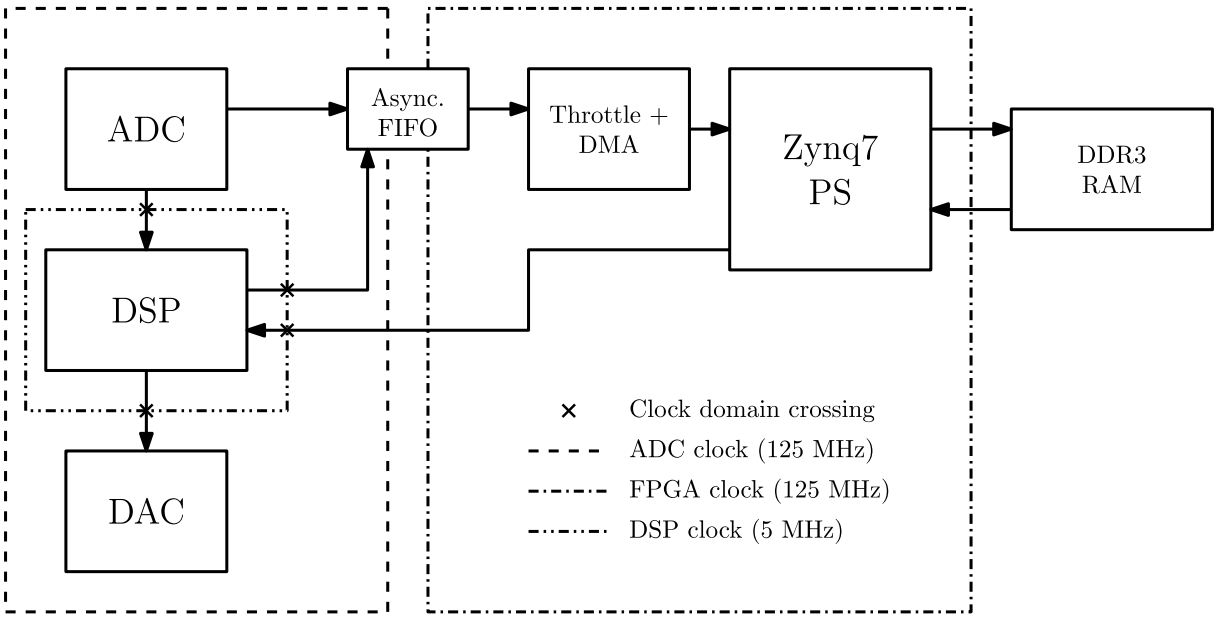


Figure 3.5: Clock domains in the implemented design. DSP clock is derived from the ADC clock and although FPGA and ADC clocks have (theoretically) the same frequency, they differ in phase.

3.3 Butterworth filter

The function of the Butterworth low-pass filter (BF) in the implemented design is twofold: first as an additional stage of the anti-aliasing filter and second as one of the DSP algorithms. The main filter, with which the displacement signal of the particle is filtered and the feedback control signal is obtained, will be discussed first.

As discussed in section 2.4.1, multiple IIR filter architectures are available. Because the Red Pitaya does not contain any hardware floating-point arithmetic resources, the filter must be implemented in fixed-point arithmetic, and the transposed direct form II (see Fig. 2.9 for schematic) should provide better numerical stability than other forms. Two second-order sections are cascaded to improve attenuation (see Fig. 3.6). The output of the second filter stage (displacement of the particle) is then multiplied by the correction gain, which is calculated along with the second-order section transformation (Matlab function `[sos,gain]=tf2sos(b,a)`). The velocity of the particle is obtained by a simple numerical differentiation and adjustable output gain.

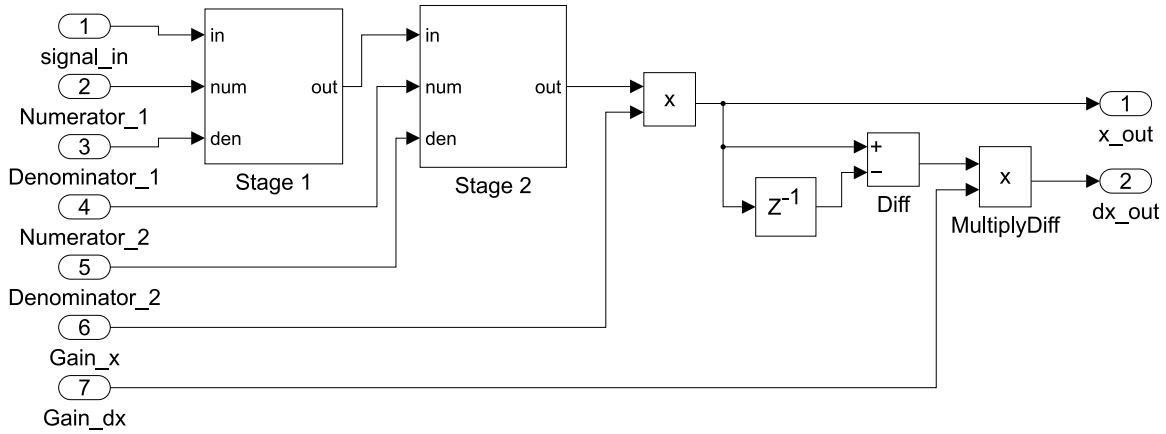


Figure 3.6: 4th order Butterworth filter Simulink model

Simulink Fixed Point tool was used to convert the model to fixed-point arithmetic to maintain numerical integrity. Red Pitaya's Artix-7 FPGA contains 80 DSP slices (multiplier-adder) that support 18×25 bit multiplication, creating a constraint for the Fixed Point tool. Therefore, the bit width for most of the coefficients and intermediate results is 18 bits, adjusted to 25 bits for some of the accumulation blocks. The output of the module is truncated to 14 bits to match the DAC resolution. Due to the requirement for configurability of the filter cutoff frequency, reference data for scaling of internal signals were collected for several design points of the cutoff frequency ranging from 20 to 350 kHz at 5 MHz sampling rate.

IP core with the AXI4-Lite interface is generated from the fixed-point model. This AXI interface then allows for the configuration of the cut-off frequency from the control application during the experiment. The HDL Coder also automatically resolves the conversion of the data rate from the fast clock domain (125 MHz) to slower clock domain of the filter (5 MHz), achieving very low latency. This means that after sampling a new value, the module output is updated in less than one clock cycle of the slow DSP clock. Some of the parameters of the implemented design are available in Table 3.2.

Sampling rate	5 MHz
Cut-off frequency	20-350 kHz
Oversampling	25 (125/5 MHz)
X phase delay	14 clock cycles (112 ns)
dX phase delay	14 clock cycles (112 ns)
Lookup tables	678 (17600 available)
Flip-Flops	880 (35200 available)
DSP slices	12 (80 Multipliers available)

Table 3.2: Parameters of the implemented 4th order Butterworth filter

3.3.1 Anti aliasing filter

The anti-aliasing filter is implemented to prevent aliasing of high-frequency components after the decimation step (lowering the sampling rate from 125 to 5 MHz). The filter is constructed in two stages(see Figure 3.7): The CIC decimator reduces the sampling rate by a factor of 5 (to 25 MHz) and feeds this signal into a second-order Butterworth low-pass filter with cut-off frequency of 2.5 MHz to improve attenuation of high frequencies (see Figure 3.8).

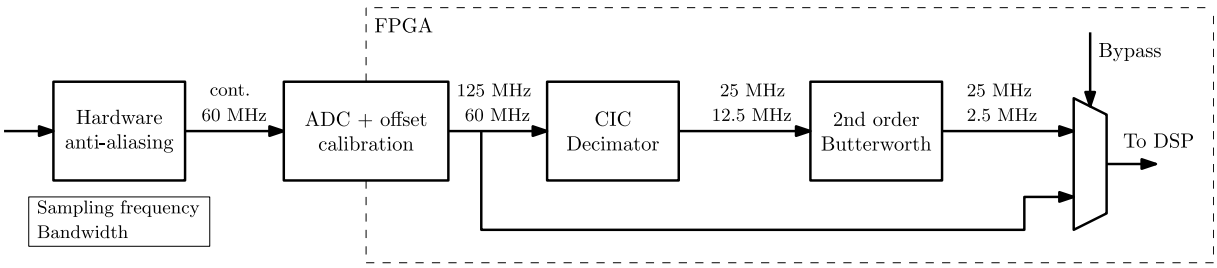


Figure 3.7: Block scheme of anti-aliasing chain. The numbers between blocks describe sampling rate and bandwidth after passing each stage.

The CIC Decimator is implemented using the Simulink block *CIC Decimation HDL Optimized* (found in the DSP System Toolbox) with a decimation factor of 5, differential delay of 1 and 2 sections. The low-pass filter is implemented as one stage of the transposed direct-form II IIR filter sampled at 25 MHz. The entire anti-aliasing digital filter is not configurable (it can only be bypassed) and adds approx. 48 ns of latency. The parameters of the generated IP cores are given in Table 3.3.

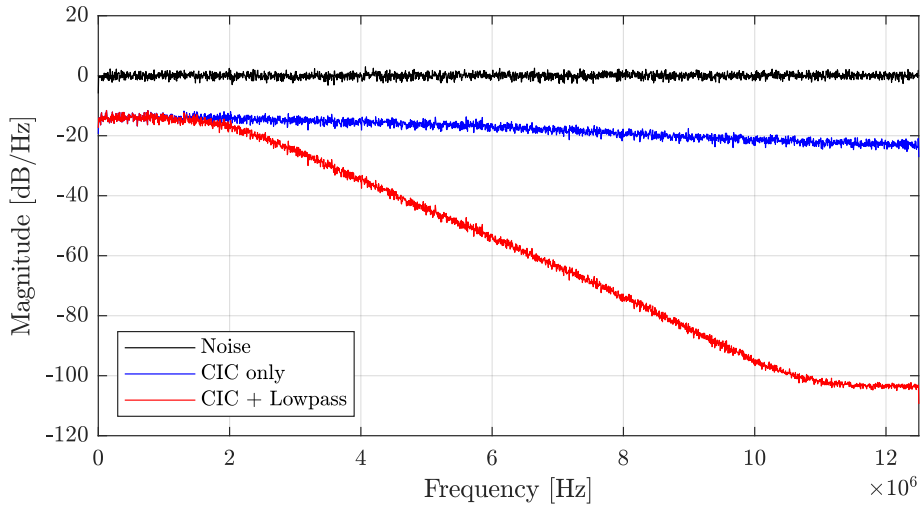


Figure 3.8: Power spectral density of original and filtered white noise signal obtained by simulation. Note that unfiltered noise is higher in magnitude due to aliasing of higher frequencies.

Parameter	**CIC Decimator**	**Low-pass**
Sampling rate	125 MHz	25 MHz
Cut-off frequency	12.5 MHz	2.5 MHz
Oversampling	1	5
Output phase delay	∅	4 clock cycles (32 ns)
Pipeline delay	2 clock cycles (16 ns)	∅
Lookup tables	119	223
Flip-Flops	178	152
DSP slices	1	6

Table 3.3: Parameters of the digital anti-aliasing filter

3.4 Linear systems

This section presents the FPGA implementation of a basic linear state-space model and primarily the extension towards the Kalman filter and the full state controller (for theoretical background, see Sections 2.4.3 and 2.4.4). Few remarks are made regarding the intricacies of numerical implementation, and the final LQG structure used in the experiments is laid out.

3.4.1 State-space model

Since only one-axis feedback is available for particle stabilization, second-order state-space representation is sufficient to describe the system. Due to the discrete-time nature of the FPGA, the original continuous-time model must be converted to discrete time (as described in Section 2.3.2). The Simulink model then follows the simple structure shown in Figure 3.9.

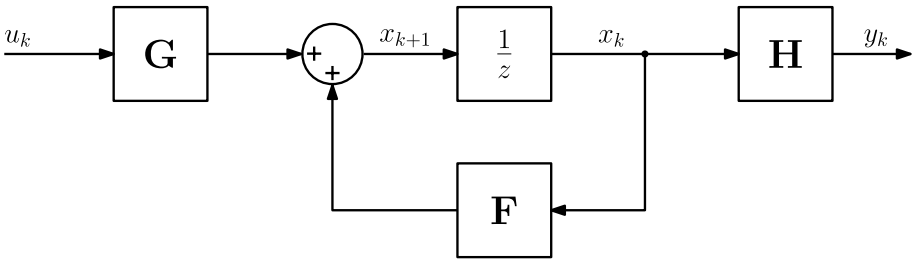


Figure 3.9: Discrete-time state-space model

To fully utilize the FPGA potential, matrix multiplications can be parallelized. The HDL Coder is fortunately able to automatically optimize such operations to ensure numerical integrity. The conversion to fixed-point arithmetic is again assisted with the Fixed-Point Tool; however, a standard state-space representation of the mass-spring-damper system (such as Equation 2.23) can pose problems in this conversion while attempting to limit bit-width of the coefficients to 18 bits (to fit multipliers supported width).

One of the ways to alleviate this problem can be through a defined transformation, such as 2.27 defined earlier. This transformation makes the non-diagonal elements of the state matrix **F** antisymmetric, but mainly changes the velocity scale to a magnitude similar to displacement. This means that bit-width originally required to capture the numerical range of velocity can now be used to improve precision of both states, improving stability and avoiding underflows.

3.4.2 Kalman filter

The implementation of the Kalman filter is derived directly from the linear state-space model described in the previous section. Another feedback loop is added, which corrects the state estimate x_{k+1} based on the current available measurement of the real system z_k (see Figure 3.10). This feedback loop (Eq. 3.1) also requires more operations during one update cycle of the model, which limits the maximum frequency of the system.

$$\mathrm{x}_{k+1} = \bar{\mathrm{x}}_k - \mathbf{K}(\mathbf{H}\bar{\mathrm{x}}_k - \mathrm{z}_k), \quad \bar{\mathrm{x}}_k = \mathbf{F}\mathrm{x}_k + \mathbf{G}\mathrm{u}_k \tag{3.1}$$

It can be understood from Figure 3.10 that for each update cycle (signal loop originating from unit delay), three matrix multiplications in series are required (**F**, **H**, and **K**). These operations cannot be easily parallelized without extensive manipulation of the equations, because they depend on previous results. However, the usual natural frequency of the particle lies in the range of tens to hundreds of kHz; hence lowering the maximum sampling rate of 125 MHz to 5 MHz still means the filter operates at a much shorter time scale than the effects to be observed. Observation of the ballistic Brownian regime is not the goal of this work.

By lowering the sampling rate, the original structure as described in Figure 3.10 can be maintained, which is easier to work with. Models with a high sampling rate also require better precision of the fixed-point representation to maintain numerical integrity and similarity to a floating-point reference model. The HDL Coder can use the sampling rate ratio (25 *fast* clock cycles to 1 update cycle) to produce an IP core that can reach timing closure.

In the linear Kalman filter, the Kalman gain calculation is independent of state (see

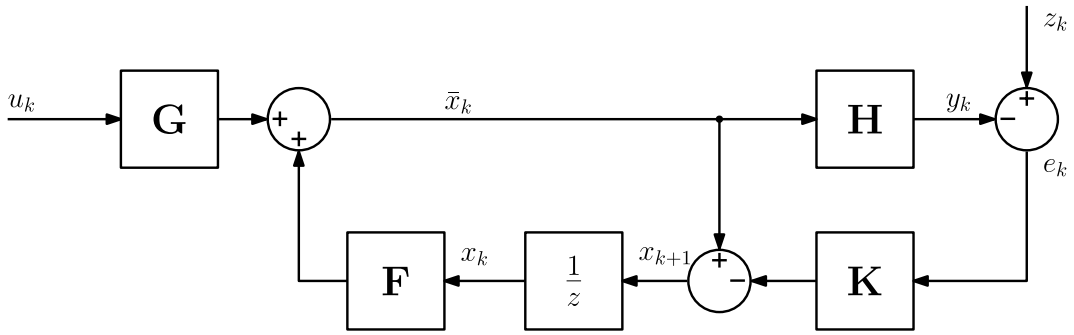


Figure 3.10: Discrete-time Kalman filter model

Table 2.1). This is leveraged to pre-calculate the Kalman gain in the Matlab app and provide the resulting matrix to the FPGA, also conveniently avoiding matrix inversion and multiple additional operations, which conserves limited DSP slices.

The analog input of the Red Pitaya is connected to the quadrant photo diode, which transduces light corresponding to the movement of the particle (typically in hundreds of nanometers) to the voltage signal between ± 1 Volt. Thus, to further simplify the implementation, the system is assumed as *voltage oscillator* by setting the observation matrix **H** to identity (ones on diagonal), neglecting the transduction coefficient C_{el}. The omega and $1/C_{el}$ state-space transformation is then written as follows:

$$\mathbf{T} = \begin{bmatrix} 1/C_{el} & 0 \\ 0 & \Omega_0/C_{el} \end{bmatrix}, \quad \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \bar{q}_1/C_{el} \\ \Omega_0 \cdot \bar{q}_2/C_{el} \end{bmatrix} \tag{3.2}$$

$$y = \begin{bmatrix} C_{el} & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \quad \Rightarrow y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \end{bmatrix} \tag{3.3}$$

$$\begin{aligned} \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\Omega_0^2 & -\Gamma_0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ C_{fb} \end{bmatrix} u_{fb} \quad \Rightarrow \\ \begin{bmatrix} \dot{\bar{q}}_1 \\ \dot{\bar{q}}_2 \end{bmatrix} &= \begin{bmatrix} 0 & \Omega_0 \\ -\Omega_0 & -\Gamma_0 \end{bmatrix} \begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ C_{el} \cdot C_{fb}/\Omega_0 \end{bmatrix} u_{fb} \end{aligned} \tag{3.4}$$

To obtain the feedback signal, each state is multiplied by the adjustable gain. The digital analog converter (which controls the feedback force) can be connected to either of the states (relative to the displacement and velocity of the particle) or to their linear combination. The third option (linear combination of states) enables full-state control with gain coefficients obtained by the Matlab function `lqr(sys,Q,R,N)` according to the linear quadratic regulator theory (see Section 2.4.4).

These gain coefficients, as well as state matrices, are configurable via the AXI4-Lite interface in real time from the Matlab application. The IP core also outputs both states to the DMA controller for recording, which are not modified by the feedback coefficients. The parameters of the HDL Coder generated IP core are presented in Table 3.4.

Sampling rate	5 MHz
Oversampling	25 (125/5 MHz)
X and dX phase delay	15 clock cycles (120 ns)
$X \cdot g_x, dX \cdot g_{dx}$ **phase delay**	20 clock cycles (160 ns)
$X \cdot g_x + dX \cdot g_{dx}$ **phase delay**	20 clock cycles (160 ns)
Lookup tables	743 (17600 available)
Flip-Flops	1503 (35200 available)
DSP slices	15 (80 Multipliers available)

Table 3.4: Parameters of the implemented Kalman filter

3.5 Configuration interface

It was briefly mentioned, that implemented DSP algorithms are configurable via the AXI4-Lite interface from the Matlab app in real time during the experiment. This section will delve into the details regarding the configuration interface consisting of the FPGA algorithm, TCP/IP server, and Matlab app.

Figure 3.11 shows the schematic of the configuration chain: An arbitrary command originates from the Matlab app on the PC and then is transferred over a TCP/IP socket to the server application running on the ARM processor in the Red Pitaya. The processor resolves the incoming command and either configures relevant peripherals in the FPGA or sends data back to the Matlab app.

3.5.1 FPGA support functions

Apart of the DMA throttle (described in Section 3.2.2) and the DSP algorithm, other modules related to data routing and signal generation are used. One multiplexer allows for the configuration of which signal is fed into the DSP algorithm, switching between two ADC channels and internal sine generator. Another multiplexer connects the selected signal to the DAC output, switching between various outputs of the DSP algorithm or the sine generator.

The sine generator is implemented with the Xilinx DDS Compiler IP core, with adjustable amplitude and frequency. It can be used either to readily test the DSP algorithm functionality without a hardware signal generator or to calibrate the feedback coefficient of the controlled system (particle in the trap), which will be explained in greater detail in Section (4.2).

3.5.2 TCP/IP server

The server is a C program running on the ARM processor and communicates with the Matlab application over the Ethernet and Linux-implemented TCP/IP socket. Upon starting the server, it configures the FPGA with the appropriate bitstream, maps the AXI configuration interfaces as memory areas, and then waits for the connection from the Matlab app. After successful connection, all of the functionality is event-based on the incoming commands from the TCP/IP interface.

The communication protocol is very simple: After receiving the command string (e.g.

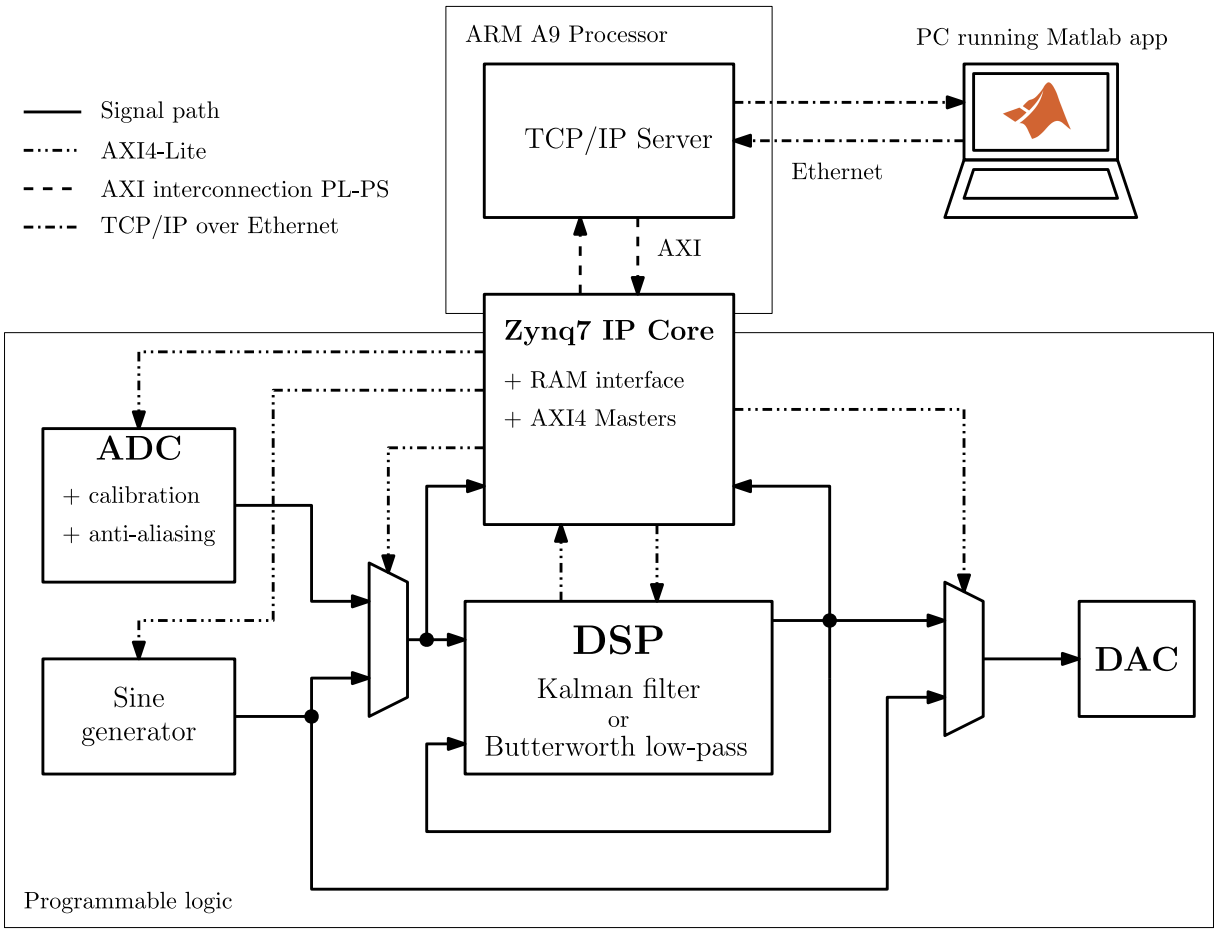


Figure 3.11: Discrete-time Kalman filter model

`MEASURE`), it is compared with the programmed functions. If a matching string is found, the requested function is executed (mostly register configuration), and the acknowledge string (`ACK`) is sent after successful execution. If an error is encountered during the execution of the function, an error message is sent back to the application, notifying the user and terminating the application in the event of severe errors.

Crucially, the server program can access the DDR3 RAM memory shared with the programmable logic where long time-traces are recorded. Note that the memory is shared only in the sense that it is accessible from both sections. The 512MB RAM is divided into halves, with the first half reserved only for the ARM processor and the other for the DMA. Before recording, the server sets the DMA descriptors according to the number of samples and the decimation factor requested from the application and initiates the throttle module that handles packetization of the data stream (described in Section 3.2.2). After reaching the requested number of samples, it can send a portion of the recorded data back to the application or dump the trace into a CSV file.

3.5.3 Matlab application

The user interface (TCP/IP client) was developed using Matlab App Designer. Note that the configuration chain is not firmly bound to Matlab because the user application communicates with the server via TCP/IP and a simple communication protocol, which could be easily implemented in LabView, Python, or any other software.

The main features of the application are:

- Recording and displaying input signal in the time or frequency domain (PSD).
- Configuring Kalman and Butterworth filters.
- Estimating model parameters from power spectral density.
- Evaluating the performance of the Kalman filter based on the innovation statistic.
- Calibrating feedback coefficient of the system.
- Calibrating ADC offset and managing signal routing.

Configuration messages from the client application are sent over the Ethernet in the form of byte array (8 bit vector). To simplify the configuration of a large number of registers, a single command is used to configure more registers at once, which requires data to be arranged in a specific order according to the register map generated by the HDL Coder.

The application supports the manipulation of up to 10^6 samples (100 ms trace of 4 channels sampled at 5 MHz); in case a larger amount of data is required, it can be remotely saved to a CSV file directly in the Red Pitaya storage and transferred over the SCP file manager.

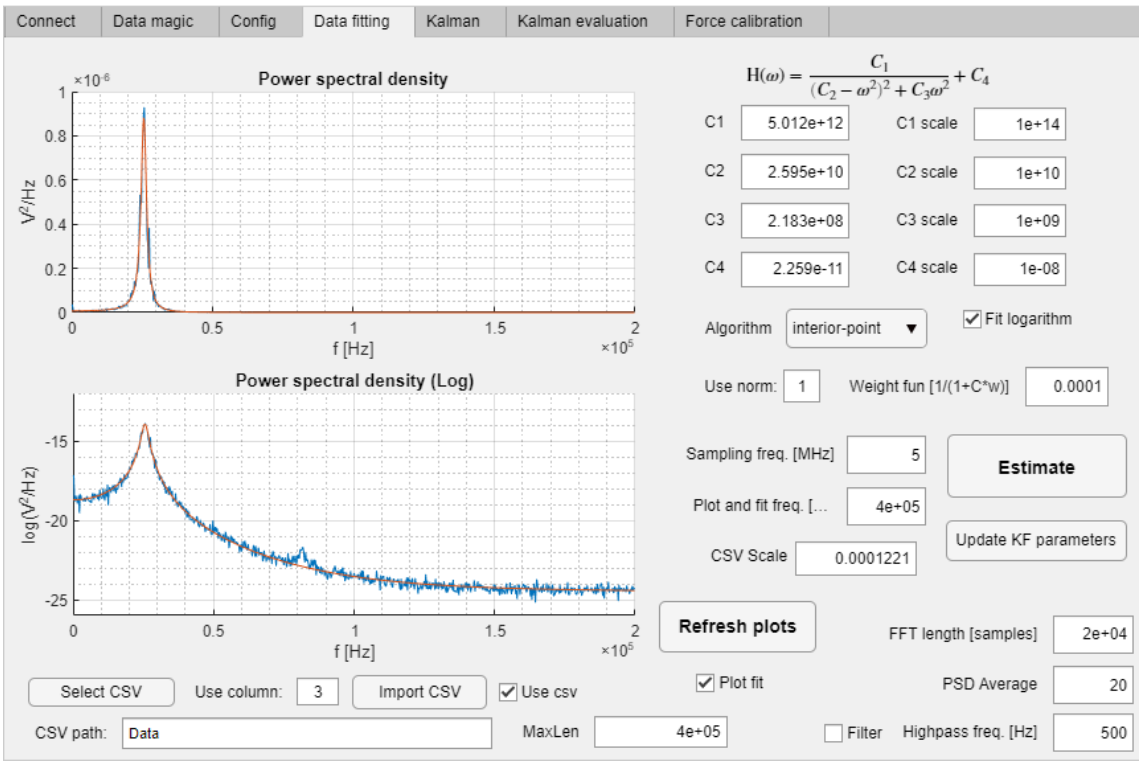


Figure 3.12: Data fitting feature of the Matlab application

4 Implementation of experiments

This chapters covers the implementation of experiments, starting with explanation of the experimental setup in the first section. In the next section, procedures used for parameter estimation during the experiments are discussed, together with effects arising from state-space transformations. Verification of the parameter estimation in simulation is also presented.

4.1 Experimental setup

Here, the experimental setup of optical tweezers is explained and the connection to the Red Pitaya and implemented algorithms is highlighted. For a theoretical overview of optical tweezers, see Section 2.2.1. The explanation will loosely follow the Figure 4.1 and will require basic knowledge of photonics, that is, light polarization and interference.

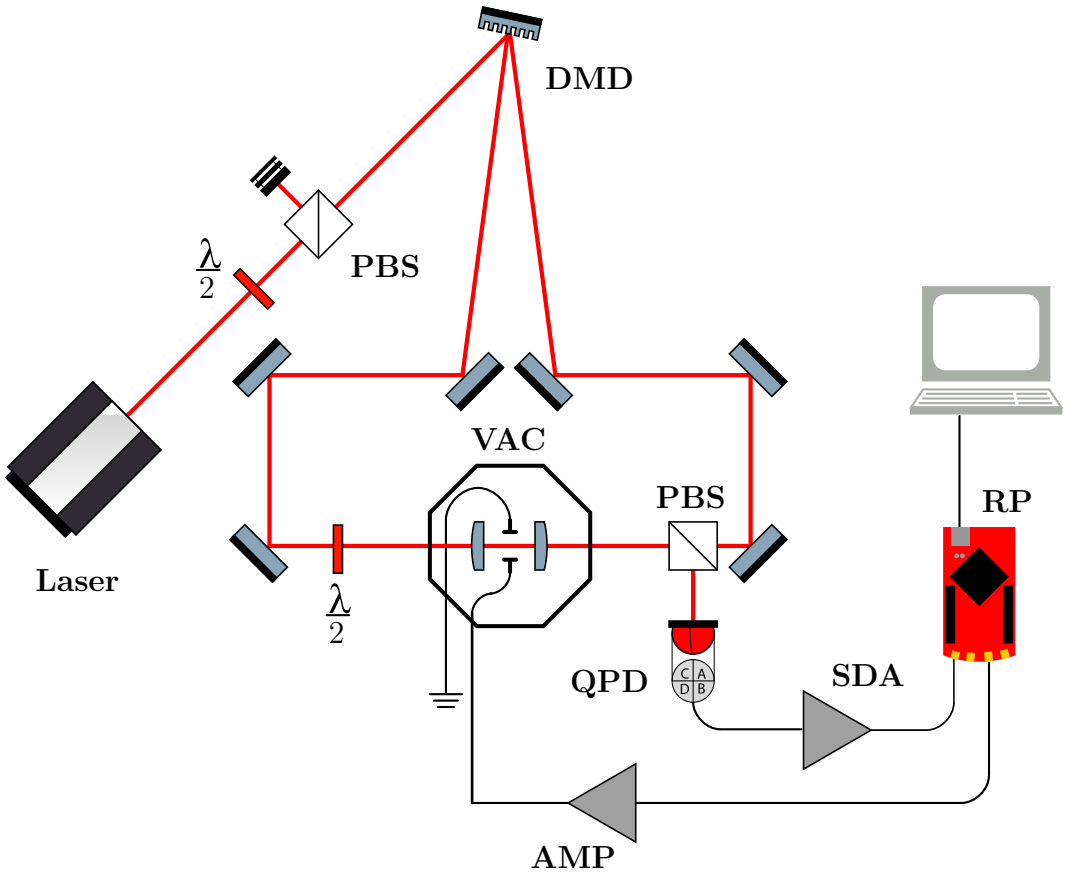
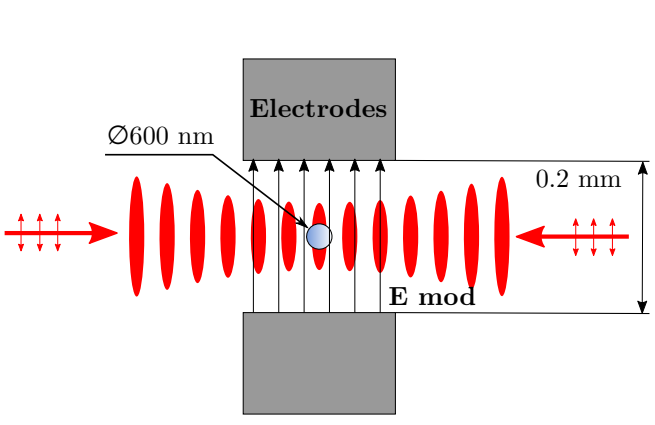


Figure 4.1: Experimental setup of optical tweezers (author V. Svak); PBS - Polarizing Beam Splitter; DMD - Digital Micromirror Device; QPD - Quadrant Photodiode; $\lambda/2$ - Half-wave plate; VAC - Vacuum chamber; RP - Red Pitaya; AMP - operational amplifier; SDA - sum-and-difference amplifier

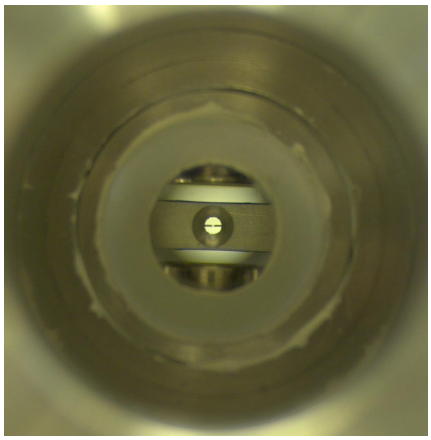
The source of the trapping beam is the laser with a 1550 nm wavelength (infrared spectrum), which passes through a half-wave plate ($\lambda/2$) and a polarizing beam splitter (PBS). The half-wave plate rotates the linear polarization of the incident light beam, which is then split into two with the PBS. According to the orientation of the polarization (determined by the wave plate), some light is passed through, and the rest is deflected and ends up in a beam dump. By rotating the half-wave plate, the power of the trapping beam can be adjusted. The light beam passed through the PBS is then split into two beams with the digital micromirror device (DMD), which are then guided into the vacuum chamber (VAC) in direct opposition using the optical system (note that most of the optical setup is omitted in Figure 4.1).

Counterpropagating light beams are tightly focused and create a standing wave in the vacuum chamber (VAC), where the particle is trapped. The particle is made of SiO2 (also called *silica*), has a diameter of approximately 600 nanometers, and its mass should be approximately $3 \cdot 10^{-13}$ grams. Its displacement is observed by the quadrant photodiode (QPD), which converts incident light on each quadrant into voltage signals, which are processed in the sum-and-difference amplifier (SDA). This amplifier then outputs three analog signals (± 1 V) proportional to left-right, top-down, and forward-back displacements of the particle relative to a plane perpendicular to the direction of propagation. Polarization of one of the counterpropagating beams must be rotated (with another half-wave plate) so that a portion of the light is deflected in the PBS and subsequently detected by the QPD. A high-speed camera is added to the setup, providing a visual feed of inside the vacuum chamber.

The particle in the trap can be influenced by the electric field created between the electrodes inserted into the chamber (see Figure 4.2). Clearly, the particle must be electrically charged for the electric field to exert any force on the particle, but it is irrelevant whether the charge is positive or negative. The electrodes are connected to an operational amplifier that amplifies the ± 1 Volt output of the Red Pitaya to ± 15 Volts. The appropriate output channel of the SDA (displacement parallel to the electric field) is connected to the analog input of the Red Pitaya, completing the feedback control loop.



(a) Standing wave and electrodes schematic



(b) Photo of the vacuum chamber and electrodes

Figure 4.2: Detail of the vacuum chamber, orientation of the schematic and the photo is the same - light travels horizontally between the electrodes. (author V. Svak)

4.2 Parameter estimation

Contrary to the Butterworth filter, the Kalman filter requires precise knowledge of the system parameters to correctly estimate hidden states from measurements. Due to the inaccessibility of good knowledge of system perturbations due to thermal fluctuation, frequency-domain parameter estimation is better suited to the problem. The procedure and model used for most of the parameter estimation are presented in the first subsection, followed by two methods for estimating the feedback coefficient.

4.2.1 Estimation model and procedure

The model used for parameter estimation (without feedback force) is the second-order linear model described in Section 2.3.1 (Eq. 4.1). The state-space representation with the thermal fluctuations perturbations and feedback force is described by Equation 4.2. The displacement of the particle is observed using the chain of optical system, quadrant photodiode, and amplifier, which ultimately converts light scattered by particle displacement into voltage signal. The conversion ratio of displacement to voltage is denoted by the transduction coefficient C_{el}. The measurement of the system (Eq. 4.3) is also influenced by additive noise v, which is assumed to be white (that is, Gaussian, zero mean).

$$\ddot{q} + \Gamma_0 \cdot \dot{q} + \Omega_0^2 \cdot q = 1/m \cdot \sqrt{2k_\mathrm{B}T\gamma_0} \cdot \xi + C_{fb} \cdot u_{fb} \tag{4.1}$$

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{A}\cdot\mathbf{q} + \mathbf{W}\cdot\xi + \mathbf{B}\cdot u_{fb} \\ \begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix} &= \begin{bmatrix}0 & 1\\ -\Omega_0^2 & -\Gamma_0\end{bmatrix}\begin{bmatrix}q\\ \dot{q}\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}\cdot\xi + \begin{bmatrix}0\\ C_{fb}\end{bmatrix} u_{fb} \end{aligned} \tag{4.2}$$

$$y = \mathbf{C}\cdot\mathbf{q} + v \qquad y = \begin{bmatrix}C_{el} & 0\end{bmatrix}\cdot\begin{bmatrix}q\\ \dot{q}\end{bmatrix} + v \tag{4.3}$$

The matrix $\mathbf{W}$ denotes the projection of stochastic momentum impulses[1] ξ (white noise) in the system, with variance $\sigma_\xi^2 = \langle\xi,\xi\rangle = 2k_\mathrm{B}T\gamma_0/m^2$. Because the stochastic term ξ is white noise that influences the states of the system, it maps directly to the *process noise* w defined in Kalman's filter theory (Section 2.4.3) [3].

Before proceeding to the model used for the parameter estimation, state-space model (Eqs. 4.2, 4.3) must be transformed by the Ω-transformation according to the Kalman filter implemented in the FPGA (Sect. 3.4.2). The system will be viewed as a voltage oscillator ($C_{el} = 1$), so this transformation produces the following model:

$$\begin{bmatrix}\dot{\bar{q}}_1\\ \dot{\bar{q}}_2\end{bmatrix} = \begin{bmatrix}0 & \Omega_0\\ -\Omega_0 & -\Gamma_0\end{bmatrix}\begin{bmatrix}\bar{q}_1\\ \bar{q}_2\end{bmatrix} + \begin{bmatrix}0\\ 1/\Omega_0\end{bmatrix}\cdot\xi + \begin{bmatrix}0\\ C_{fb}/\Omega_0\end{bmatrix} u_{fb}, \quad y = \begin{bmatrix}1 & 0\end{bmatrix}\cdot\begin{bmatrix}\bar{q}_1\\ \bar{q}_2\end{bmatrix} + v \tag{4.4}$$

The transfer function $G(s)$ of equation 4.4 is then obtained directly from the state space matrices [35, Chapter 9]:

[1] They are called *momentum* impulses here due to the reorganization of the terms: division by particle mass m is absorbed into the stochastic term ξ to simplify future transformations

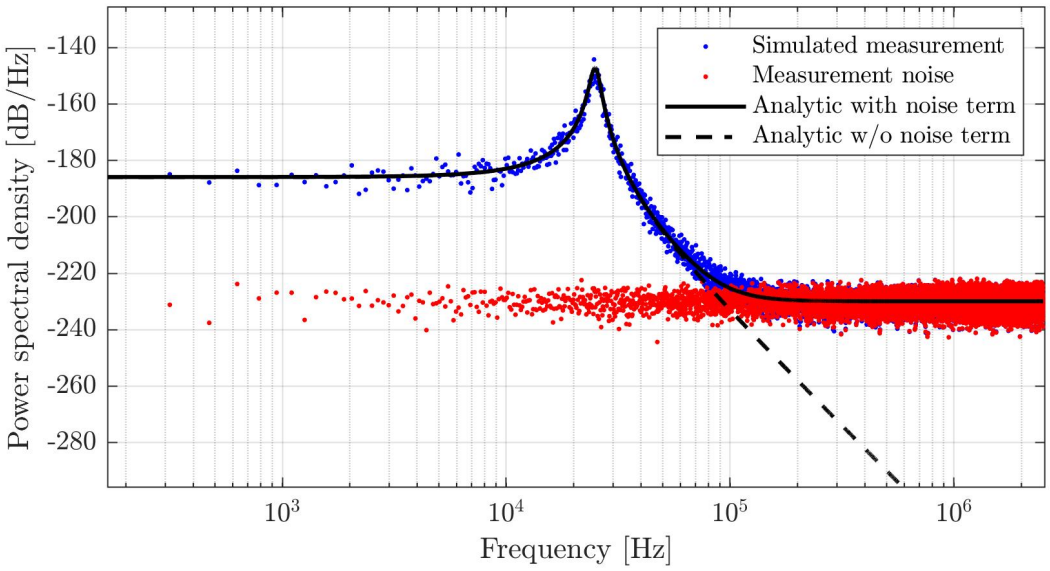


Figure 4.3: PSD estimate from simulation compared to analytical expression

$$G(s) = \frac{y}{\xi} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{W} = \frac{1}{\Omega_0^2 + \Gamma_0 s + s^2} \tag{4.5}$$

The power spectral density of the measured displacement S_y is the result of multiplying the transfer function $G(s)$ (Eq. 4.5) by ξ, substituting the $s = i\omega$ and squaring the complex magnitude. Because measurement y consists of a signal proportional to displacement q and additive noise v, the spectral density of measurement noise S_v is added:

$$S_y = |\xi \cdot G(s)|^2 + S_v = \left| \frac{\sqrt{2k_\mathrm{B}T\gamma_0}}{m} \cdot \frac{1}{\Omega_0^2 + \Gamma_0 s + s^2} \right|^2 + S_v = \frac{S_w}{(\Omega_0^2 - \omega^2)^2 + \Gamma_0^2 \cdot \omega^2} + S_v \tag{4.6}$$

where $S_w = 2k_\mathrm{B}T\gamma_0/m^2$. Therefore, for a proper configuration of the Kalman filter, four parameters are required: natural frequency Ω_0, damping coefficient Γ_0, process noise variance σ_W^2, and measurement noise variance σ_V^2. As is evident from Equation 4.6, these parameters are recovered by fitting the power spectral density of the measured displacement of the particle with this function (see Figure 4.3).

The first step of the data fitting procedure is to record a sufficiently long time trace of the particle displacement (without any feedback applied), from which a power spectral density estimate is obtained using the Bartlett method (see Section 2.1.2). Then, the Matlab function `fmincon` is used to fit the function 4.7 to the data:

$$S_{fit}(\omega) = \log\left(\frac{C_w}{(C_\Omega - \omega^2)^2 + C_\Gamma \cdot \omega^2} + C_v \right) \tag{4.7}$$

Where $C_w = S_w$, $C_\Omega = \Omega_0^2$, $C_\Gamma = \Gamma_0^2$, and $C_v = S_v$. However, fitting this function is not straightforward due to the disappearing gradient outside of the resonance peak. Therefore, a good initial estimate is required which can be easily tuned visually. The

intuition of the parameters C_Ω and C_Γ is obvious, with the first shifting the resonance frequency and the second flattening and broadening the main peak. Then C_w is a scaling factor that multiplies the PSD by a constant. The last C_v then determines the noise floor, which manifests itself in the logarithmic axes by curving and leveling the high-frequency tail, where the displacement spectral density is lower than that of the measurement noise.

In addition to the initial guess, fitting the logarithm of the spectral density improves the reliability, as well as replacing the quadratic parameters by linear ones. `fmincon` solver also allows the specification of constraints, so that the parameters cannot diverge in a non-realistic direction (e.g., negative measurement noise power). The estimated parameters are then finally used to determine the Kalman filter matrices and noise parameters, which are used to configure the FPGA. For the sampling rate F_s, the parameters are recovered, such as:

$$\begin{aligned} \Omega_0 &= \sqrt{C_\Omega} & \Gamma_0 &= \sqrt{C_\Gamma} \\ \sigma_v^2 &= C_v \cdot \frac{F_s}{2} & \sigma_w^2 &= C_w \cdot \frac{F_s}{2} \\ \mathbf{R} &= \sigma_v^2 & \mathbf{W} &= \begin{bmatrix} 0 \\ 1/\Omega_0 \end{bmatrix} \\ \mathbf{Q}_{discrete} &= \mathbf{W_d} \sigma_w^2 \mathbf{W_d^T} & \mathbf{Q}_c &= \mathbf{W} C_w \mathbf{W^T} \end{aligned}$$

Where $\mathbf{R}$ is the measurement noise matrix, $\mathbf{Q}_{discrete}$ the process noise matrix for the discrete noise model, $\mathbf{W_d}$ is the discretized[2] matrix $\mathbf{W}$ (see Eq. 4.4). $\mathbf{Q}_c$ is the upper right quadrant of the $\mathbf{\Lambda}$ matrix used in continuous noise model (see Eq. 2.36).

4.2.2 Feedback calibration - harmonic

To complete the Kalman filter model, the feedback coefficient C_{fb} must be known to properly correct the state estimation based on the control input. This coefficient determines the effect of the voltage applied to the operational amplifier (AMP in Figure 4.1) on the particle in the optical trap, combining the mass and charge of the particle, as well as the electric field created by the electrodes. When the state-space model is transformed into the voltage oscillator as described in the previous section, the transduction coefficient is also included in the estimation.

The first calibration method presented here was introduced in the article by Magrini et al. [3]. It requires an adjustable sine-wave source that is connected to the feedback input of the experimental setup. A series of sine waves varying in frequency and amplitude are then introduced to the electrodes, and the response of the particle is recorded. For convenience, the sine-wave generator is also implemented directly in the Red Pitaya, allowing for quick calibration of the system.

The coefficient is simply obtained by comparing the variances of the control signal and the displacement of the particle. For off-resonant frequencies and lightly damaged oscillator, the displacement variance corresponds to:

$$\sigma_q^2 = \frac{1}{2\pi} \int_{\Omega_d - \epsilon}^{\Omega_d + \epsilon} \left(S_y(\omega) - S_v \right) d\omega = \frac{0.5 F_d^2}{\left(\Omega_d^2 - \Omega_0^2 \right)^2} \tag{4.8}$$

[2]The discretization procedure is the same as with the input matrix $\mathbf{B} \rightarrow \mathbf{G}$

Where σ_q^2 is the displacement variance estimated by integrating the power spectral density S_y (corrected by subtracting the PSD of the measurement noise S_v) near the sine-wave frequency Ω_d. By rearranging the terms, the standard deviation of the applied force is obtained:

$$\sqrt{F_d^2} = \sqrt{\sigma_q^2} \cdot \left(\Omega_d^2 - \Omega_0^2\right) \tag{4.9}$$

The *force* F_d is linearly proportional to the voltage provided to the control input $C_{fb,lump} \cdot u_{fb}$, therefore the lumped feedback coefficient $C_{fb,lump} = C_{el} \cdot C_{fb}$ (containing more parameters discussed earlier) is recovered by dividing the standard deviation of the F_d by the standard deviation of the applied signal (sine wave). Calibration is improved by linear regression of multiple measurement points with varying frequency and amplitude; the coefficient is then given by the slope of the regressed line, as in Figure 4.4.

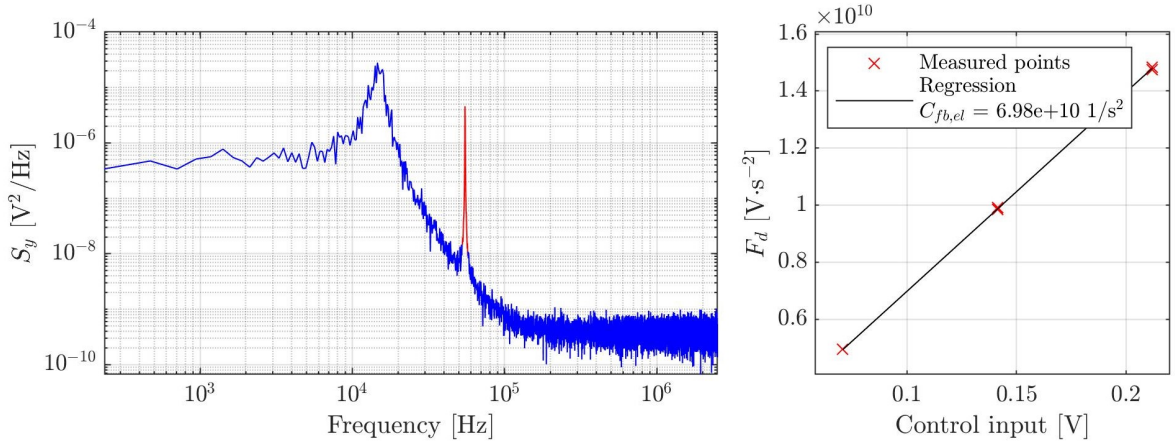


Figure 4.4: Simulated calibration of the feedback coefficient. Left - power spectral density with highlighted peak due to harmonic force. Right - Regression of multiple measurement points. (Inspired by [3])

4.2.3 Feedback calibration - damping

The other method of calibrating the feedback coefficient utilizes cold damping. The signal proportional to the velocity of the particles is connected to the electrodes, and a change in the damping of the harmonic oscillator is observed. This change is then directly proportional to the coupling of the particle with the electrodes C_{fb}, as well as the transduction coefficient C_{el}.

The control signal is obtained as a basic numerical difference from the signal from the Red Pitaya's analog input, multiplied by the adjustable gain g_{cd}. Therefore, the input signal is proportional to the difference of the measured displacement (voltage signal), but at the same time feeds a portion of the measurement noise v_d (with PSD S_d) back to the system. It is also affected by the transport delay τ, which has an adverse effect on the feedback loop. Substituting $u_{fb} = C_{el}\, g_{cd} \cdot \dot{q}(t-\tau)) + g_{cd} \cdot v_d(t)$ into 4.1 produces the following differential equation 4.10 and power spectral density 4.11:

$$\ddot{q}(t) + \Gamma_0 \cdot \dot{q}(t) + \Omega_0^2 \cdot q(t) = \frac{\sqrt{2k_\mathrm{B}T\gamma_0}}{m} \cdot \xi(t) + C_{fb} \cdot (C_{el}\, g_{cd} \cdot \dot{q}(t-\tau) + g_{cd} \cdot v_d(t)) \quad (4.10)$$

$$S_y(\omega) = \frac{S_w}{S_n} + \frac{G_{fb}^2 \cdot S_d}{S_n} + S_v, \quad G_{fb} = g_{cd} \cdot C_{el} \cdot C_{fb} \quad (4.11)$$

$$S_n(\omega) = \left(\Omega_0^2 - \omega^2 + G_{fb}\,\omega \sin(\tau\,\omega)\right)^2 + \left(\Gamma_0 \cdot \omega - G_{fb}\,\omega \cos(\tau\,\omega)\right)^2 \quad (4.12)$$

For very small transport delay $\tau \to 0$ and $S_d \ll S_w$, the equation 4.11 can be simplified as:

$$S_y(\omega) = \frac{S_w}{\left(\Omega_0^2 - \omega^2\right)^2 + \left(\Gamma_0 - G_{fb}\right)^2 \cdot \omega^2} + S_v, \quad G_{fb} = g_{cd} \cdot C_{el} \cdot C_{fb} \quad (4.13)$$

Evidently, by applying a signal proportional to the derivative of the displacement to the control input, the damping of the system is altered. By fitting either equation 4.11 or 4.13 to the recorded data, the new damping coefficient is recovered. The feedback coefficient is then easily calculated with knowledge of the internal signal gain g_{cd}:

$$C_{fb} \cdot C_{el} = G_{fb}/g_{cd}(\cdot F_s) \quad (4.14)$$

If the control signal is produced as a numerical difference $u_{fb} = g_{cd} \cdot (y_k - y_{k-1})$, the calculation must be corrected for with the sampling rate $F_s = 1/\Delta t$ ($\dot{q} \approx \frac{\Delta q}{\Delta t}$). Note that the product $C_{fb} \cdot C_{el}$ is required for the input matrix of the transformed model (voltage oscillator); otherwise, to be able to separate only the feedback coefficient, the transduction coefficient C_{el} would also have to be determined by some other method.

4.2.4 Verification

Estimation procedure was verified first in a simulated experiment in Simulink. The reference is the continuous linear state-space model described previously, driven by band-limited white noise (see Figure 4.5). The parameters of the model are presented in Table 4.1. Note that the credibility of this simulation is limited because its main purpose is to evaluate the estimation procedures, not to simulate accurate behavior of a particle in the optical trap. Part of the time trace generated by this simulation is shown in Figure 4.6.

The first step is to estimate the main parameters of the system, specifically viewed as the voltage oscillator as described in Section 4.2.1. As a result, the spectral density of the process noise $\tilde{S}_w = S_w \cdot C_{el}^2$ and the feedback coefficient $\tilde{C}_{fb} = C_{fb} \cdot C_{el}$ contain the transduction coefficient C_{el}. To be able to compare these parameters directly, they are corrected with perfect knowledge of C_{el} (otherwise unknown) back to the *real* parameters.

After estimating the main parameters, the feedback coefficient is estimated with the presented methods: harmonic excitation and cold damping. Harmonic excitation used six estimation points ranging from 40 to 65 kHz with amplitudes 1-3 Volts (as in Fig. 4.4). Cold damping was achieved with a simple discrete difference, multiplied by gain $g_{cd} = -3$. Both methods achieve comparable precision, but damping estimation requires

a more involved process of fitting the power spectrum (Figure 4.7), whereas harmonic estimation can be easily automated. The results of the estimation are compared in Table 4.1. Overall, the parameters were estimated with sufficient precision in this ideal scenario, and any imprecision should be lower than general uncertainty in the model of the real system, e.g. neglected nonlinearities.

Parameter	Ground truth	Estimated	Error
Natural frequency $\Omega_0/2\pi$ [kHz]	15.00	14.98	0.13 %
Damping coefficient $\Gamma_0/2\pi$ [kHz]	1.94	1.81	6.7 %
Process noise PSD S_w $[\frac{\mathrm{N}^2}{\mathrm{Hz}\cdot\mathrm{kg}^2}]$	$1.770 \cdot 10^{-7}$	$1.734 \cdot 10^{-7}$	2.03 %
Measurement noise variance σ_v^2 $[\mathrm{V}^2]$	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	0.14 %
Transduction coefficient C_{el} [V/nm]	10	∅	∅
Feedback coefficient C_{fb} **(damp.)** $[\frac{\mathrm{N}}{\mathrm{kg}\cdot\mathrm{V}}]$	7.00	6.95	0.68 %
Feedback coefficient C_{fb} **(harm.)** $[\frac{\mathrm{N}}{\mathrm{kg}\cdot\mathrm{V}}]$	7.00	7.00	0.06 %
Transport delay τ_{fb} $[\mu\mathrm{s}]$	4.00	3.99	0.23 %

Table 4.1: Comparison of estimated parameters

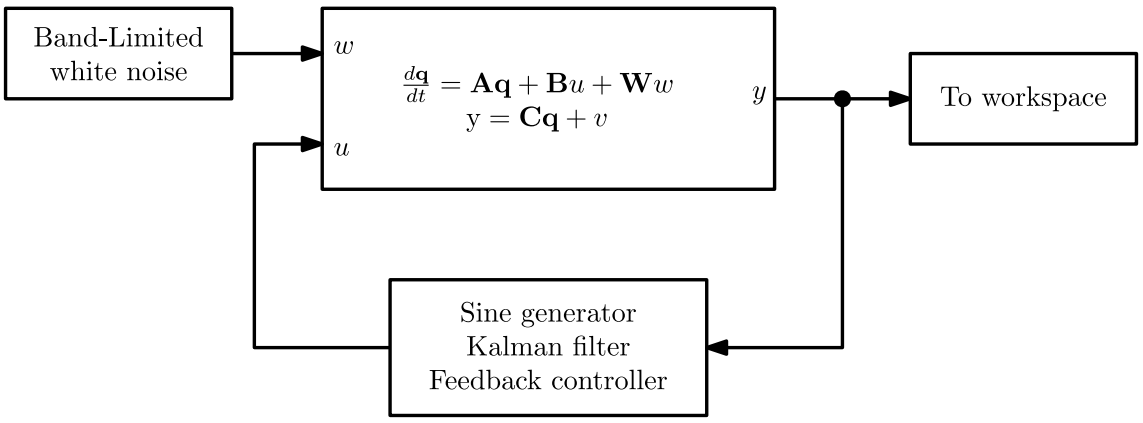


Figure 4.5: Block scheme of Simulink simulation

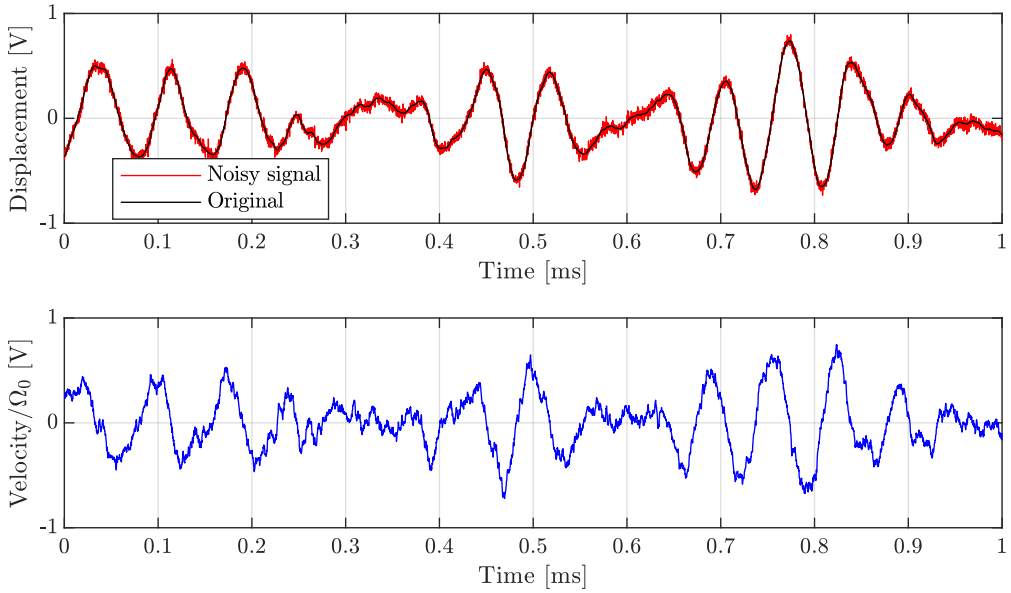


Figure 4.6: Results of a simulation: Top - particle displacement (true and noisy); Bottom - normalized particle velocity (only true)

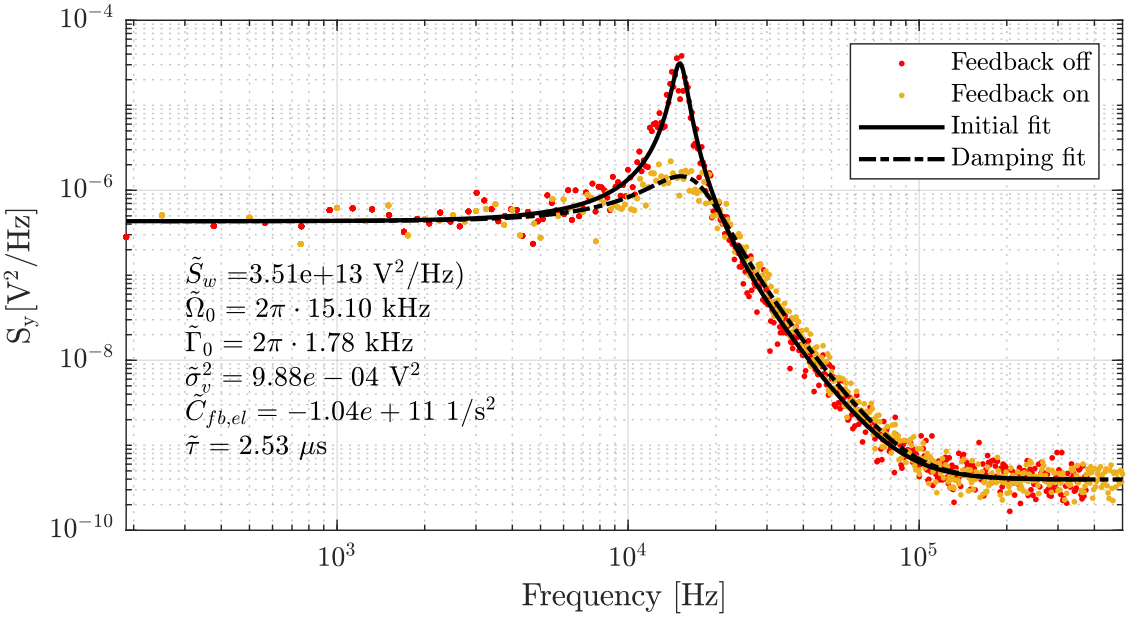


Figure 4.7: Estimated parameters of the particle via damping

5 Experimental results

The main motivation of this thesis is to implement feedback cooling / stabilization algorithms in Red Pitaya and to test the algorithms in a real experimental setting located at the Institute of Scientific Instruments of the Czech Academy of Sciences. The experimental setup is described in Section 4.1, but the important parameters are summarized in Table 5.1. The particle is an electrically charged silica nanoball (⌀600nm) and is trapped in a standing wave created by two counterpropagating Gaussian light beams ($\lambda = 1550$ nm) at a pressure of approximately 10 mbar.

The part of the light incident on the particle is scattered and is converted to a voltage signal by a quadrant photodiode and amplifier. The output of the amplifier is connected to Red Pitaya for signal processing, which generates a feedback signal applied to the electrodes in the vacuum chamber, which exerts a force proportional to voltage on the particle.

First, measurement noise is analyzed with respect to the assumptions stated previously, which is followed by parameter estimation. With the estimated model, the Butterworth and Kalman filters are evaluated and compared. Finally, feedback cooling of the particle is presented, the methods used are compared, and the results are assessed to determine whether they correspond to expectations.

Parameters of the particle	
Diameter	600 nm
Mass	$3 \cdot 10^{-13}$ g
Natural frequency	22-25 kHz
Damping ratio Ω_0/Γ_0	8-12
Typical displacement amplitude	200-300 nm
Parameters of the setup	
Bath temperature	23°C
Pressure in vacuum chamber	10 mbar
Red Pitaya sampling rate	5 MHz
Displacement signal range	±1 V
Measurement noise variance σ_v^2	$100 \cdot 10^{-6}$ V^2
Feedback input range	±1 V

Table 5.1: Overview of the experimental parameters

5.1 Measurement noise

The Kalman filter theory (Sect. 2.4.3) assumes measurement noise that is additive and Gaussian/White. For this reason, its probability density function (PDF) should be Gaussian, its mean zero, and the power spectral density constant over the entire frequency range. So, for this evaluation, two one-second-long traces of the output signal without particle in the vacuum chamber were recorded. One with an insulated setup and camera turned off, and another without complete setup insulation and camera turned on. The recorded signal contains only noise components of the laser, the photodiode with amplifier, the analog-digital converter, and other environmental sources.

As can be seen in Figures 5.1 and 5.2, the white-noise assumptions are not met. Both PSDs of noise are clearly not constant in the frequency range. There is apparent low-frequency 1/f noise reaching up to 10 kHz and also a prominent peak from 200 kHz up to 1.5 MHz. The sources of 1/f noise are probably low-frequency electrical interference and thermal fluctuations of air, which can be partially limited by completely insulating the setup.

The high-frequency peak is caused by a laser modulation (hence is visible even without a particle), and thus cannot be eliminated. However, 200 kHz is well beyond the frequencies of interest (as visible in Figure 5.1) and the peak is several orders weaker than the particle signal.

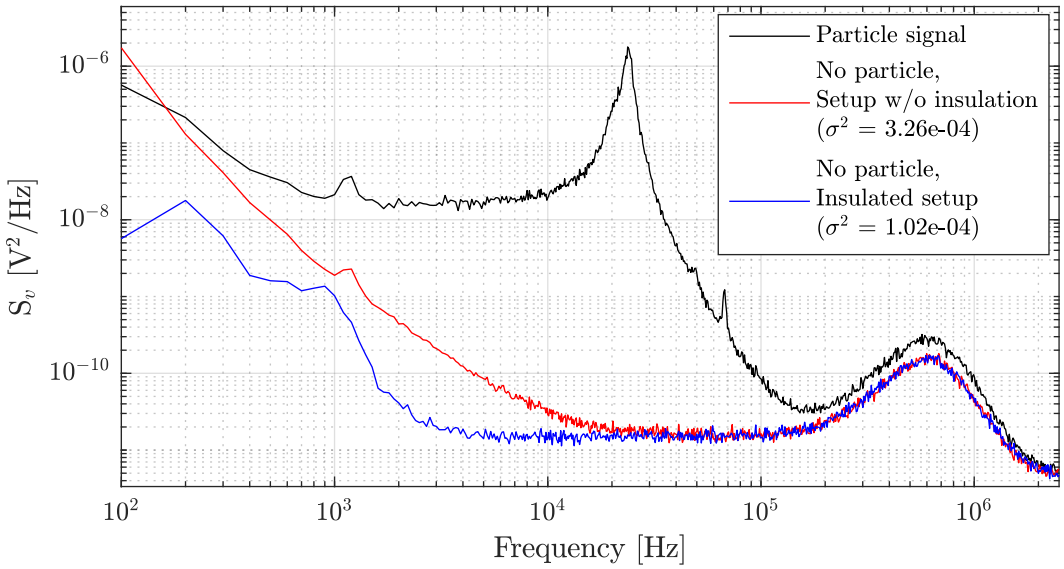


Figure 5.1: Power spectral density of measurement noise without particle in the vacuum chamber (red and blue) and the PSD of particle displacement (black, top) for comparison.

For the insulated setup, the probability density function is almost perfectly Gaussian, contrary to the uninsulated setup. The nonzero mean is due to the absence of offset calibration. Although the assumptions are not perfectly met, feedback cooling is still possible, even without a completely insulated setup, as will be shown in the following sections. Note that the setup is extremely sensitive to the environment, and the presented measurements are thus meant only as an indication of the general shape of the noise

spectra, which is important for intuitive understanding.

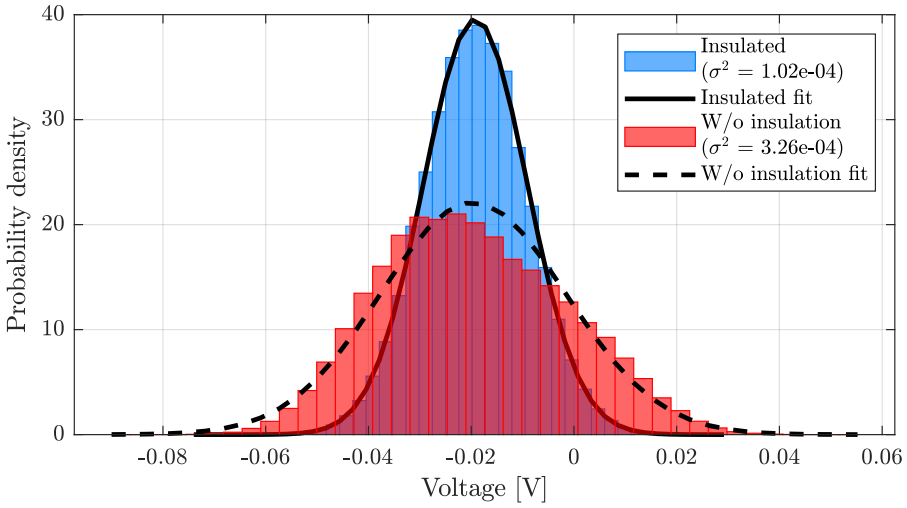


Figure 5.2: Histograms and probability density functions of measurement noise

The digital anti-aliasing decimation filter in the Red Pitaya (described in Section 3.3.1), which prevents aliasing of high-frequency components after decimation, can be bypassed to lower the latency of the feedback loop. This is at least a dubious practice and is generally discouraged; however, the effect of the anti-aliasing filter is demonstrated in Figure 5.3. Clearly, the effect is negligible up to approximately 1 MHz because the higher frequency components are weak compared to the input signal. Thus, the filter can probably be bypassed in case the DSP latency poses problems.

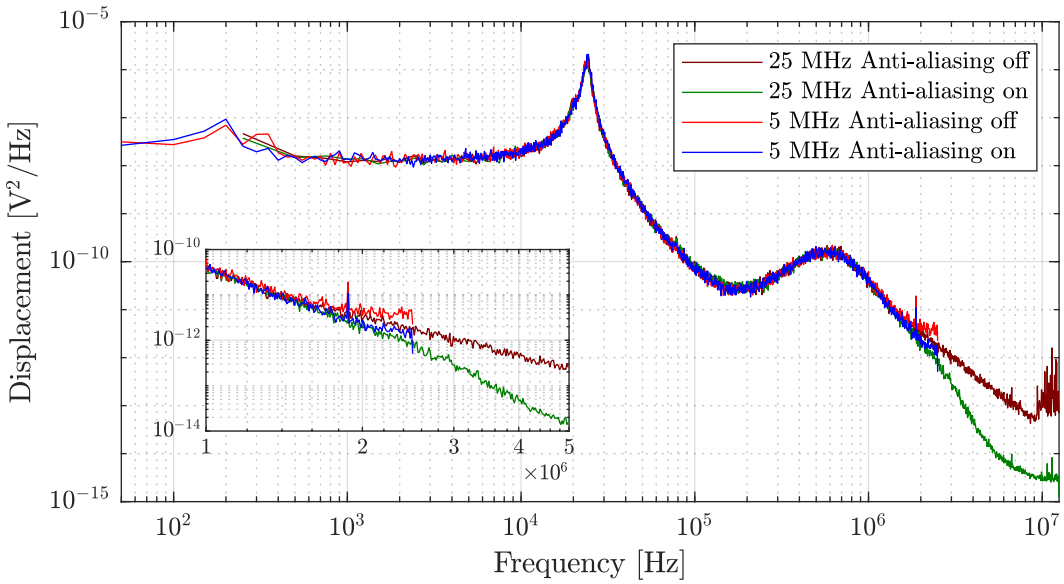


Figure 5.3: Comparison of displacement spectra with and without anti-aliasing filter. Zoomed box is magnifying the high-frequency tail (1-5 MHz).

5.2 Signal filtering

Two algorithms for signal filtering were tested, the Butterworth low-pass filter with adjustable cutoff frequency and Kalman filter. Signal processing is also required for the real-time estimation of particle velocity for both cold damping and full-state feedback.

5.2.1 Butterworth filter

The implementation details of the Butterworth filter are discussed in Section 3.3 and the theoretical background in Section 2.4.1. The filter offers only a single configuration parameter in the form of a cutoff frequency, the figure 5.4 shows the efficacy of different settings in the displacement spectrum and the estimated velocity spectrum, which is obtained as a numerical difference from the filtered signal.

According to expectations, the lowpass filter effectively attenuates frequencies beyond the cut-off point and passes lower frequencies, thus eliminating the laser modulation peak. The estimated velocity roughly corresponds to the expected shape for the highest cutoff frequency of 100 kHz, however the attenuation of higher frequencies is clearly visible even in velocity spectra.

The time traces of the filtered signal are plotted in Figure 5.5. There is an apparent time delay of the filtered signal for all settings, which is mainly caused by the phase characteristic of the IIR filter. This delay is apparently lower for higher cutoff frequencies, which can be expected, and it even follows the particle movement more closely than more aggressive settings. The implications for cold damping are evaluated in subsequent sections.

5.2.2 Kalman filter

The important advantage of the Kalman filter compared to Butterworth filter is the presence of the internal model of the measured system. This implies that the algorithm can estimate hidden states, which is leveraged here to estimate particle velocity. Theoretical background is explained in Section 2.4.3 and implementation in Section 3.4.2. Parameter estimation is a prerequisite for the deployment of the Kalman filter, and the procedures are explained in Section 4.2.

As shown in the previous section, the spectrum of measurement noise is not constant, which complicates the parameter estimation procedure. Fortunately, the natural frequency and damping coefficient are mostly unaffected and the main imprecision is in the measurement noise spectral density, which is underestimated by the *constant* line approximation.

Two noise models were compared - continuous and discrete. As can be seen in Figure 5.6, the noise models are very similar in performance. The only slight difference is that the discrete filter slightly underestimates the measurement noise, resulting in a lower magnitude of innovation PSD than is expected (exactly equal to noise floor). Neither model matches the expected PSD of velocity very well for frequencies lower than 10 kHz; the reason is unfortunately unclear.

Figure 5.7 shows a short recording of Kalman-filtered signals. The displacement closely tracks the input signal without time delay and the velocity roughly corresponds to what would be expected. Note that due to the stochastic nature of the particle in optical trap, its trajectory is erratic and the intuitive metric of *smoothness* of the signal is not really applicable.

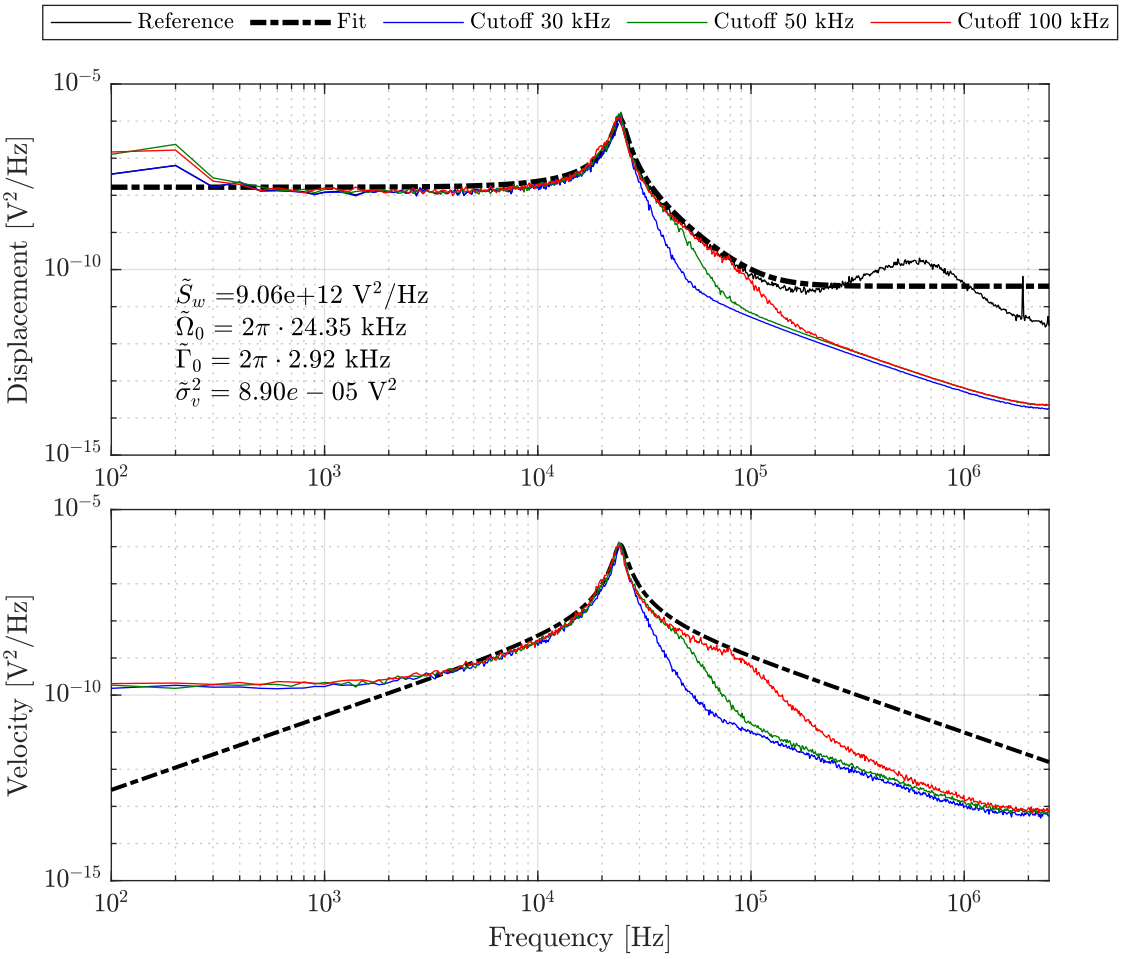


Figure 5.4: Histograms and probability density functions of measurement noise

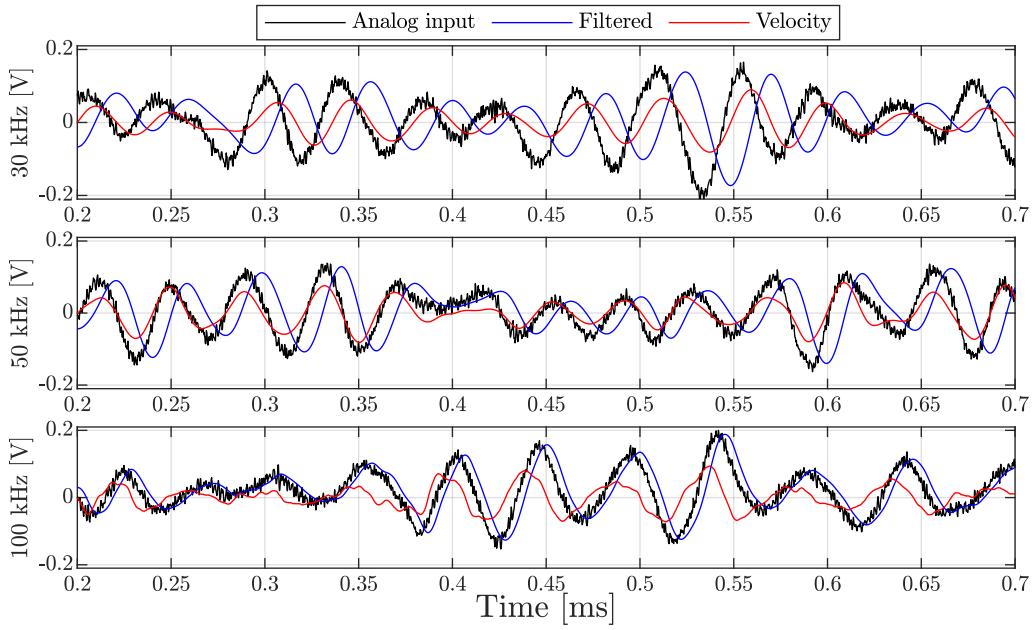


Figure 5.5: Time traces of filtered signal for different cutoff frequencies.

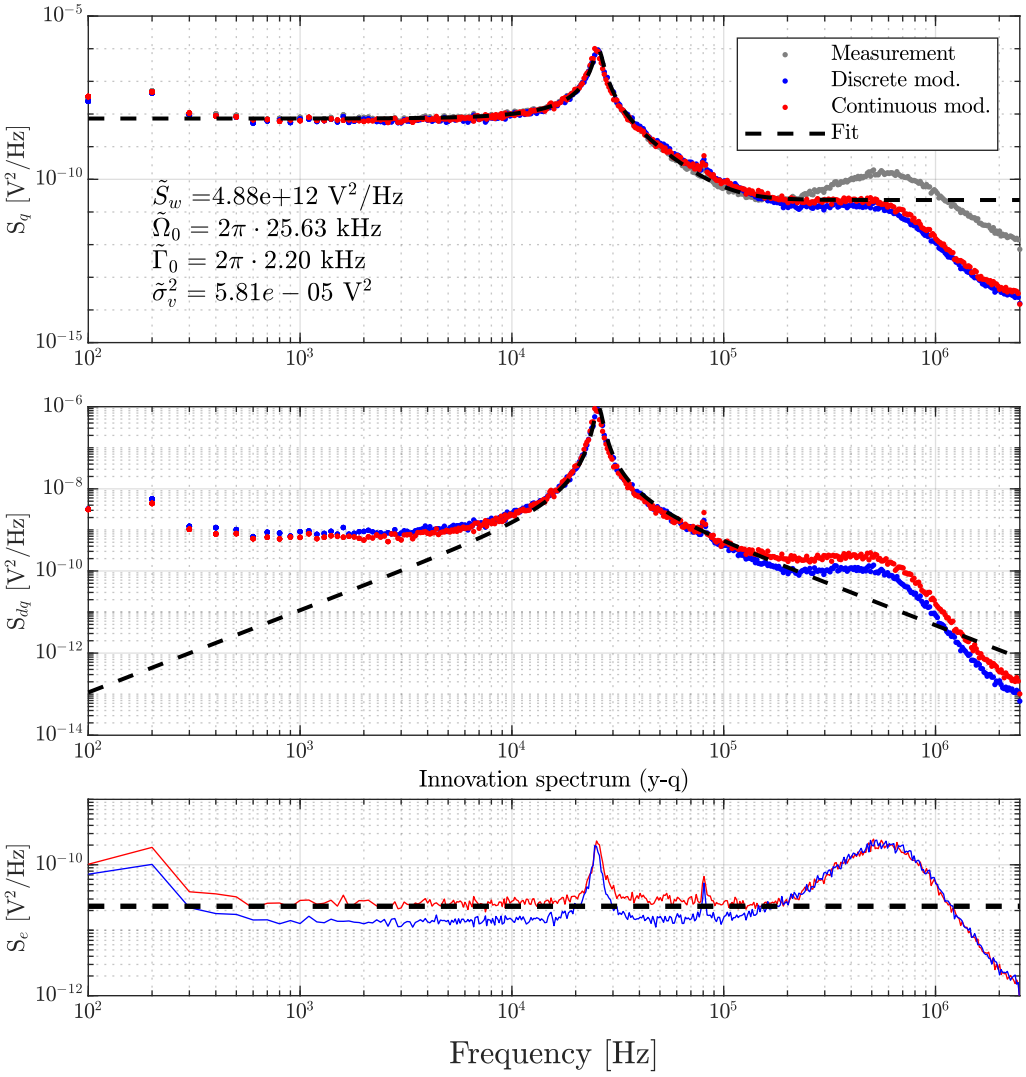


Figure 5.6: Spectral analysis of Kalman filter noise models for displacement (top), velocity (middle) and innovation spectrum (bottom). The innovation spectrum is obtained as a difference between samples of measured signal and displacement estimate.

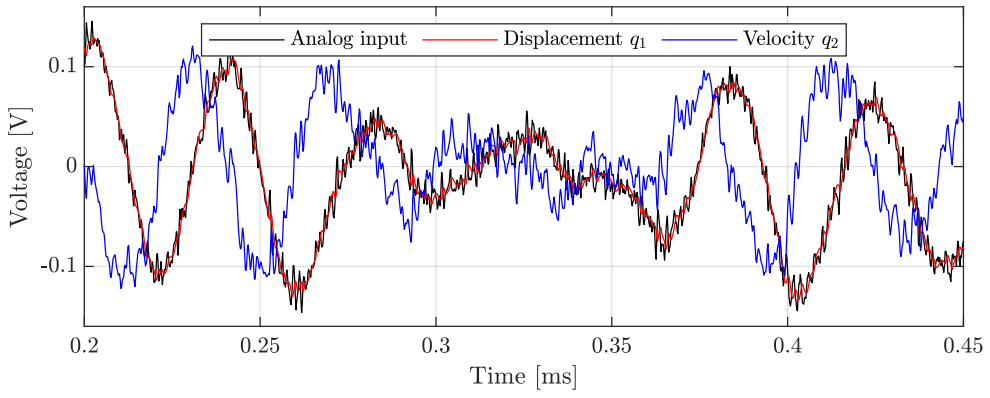


Figure 5.7: Time trace of input signal and Kalman filter estimates of displacement and velocity.

5.3 Feedback cooling

Feedback cooling (or stabilization) is achieved by creating an electric field in the vacuum chamber, which exerts a force on the particle, which must be electrically charged. The electric field is proportional to the applied voltage, which is supplied from the Red Pitaya digital-analog converter and amplified by an operational amplifier. The signal controlling the electric field can be one of the following:

- Difference of the analog signal (passed only through the anti-aliasing filter).
- Difference of signal filtered by the Butterworth filter.
- Velocity estimate from the Kalman filter.
- Linear combination of estimated states from the Kalman filter (LQR).

Each strategy will be evaluated in the following sections. The feedback performance metric is an integral of the power spectral density in the frequency band where the displacement signal exceeds the noise floor, which has been chosen as 1-200 kHz. Taking advantage of the Parseval theorem, this integral corresponds to the variance (or power) of the signal in the time domain, which can be assigned an equivalent temperature [4] (see Section 2.2.4).

5.3.1 Cold damping

Principle of the cold damping strategy, as the name suggests, lies in artificially increasing the damping of the particle. This can be achieved by applying a force proportional to particle velocity, which can also be considered a D-regulator from the control-theory viewpoint. In the experiment, three sources of velocity signal were tested: first, a simple numerical difference of input signal, then the difference of lowpass-filtered signal, and finally the velocity estimate from Kalman filter.

The performance of the Butterworth low-pass filter is shown in Figure 5.8. The performance is apparently not satisfactory, as the damped spectra are strongly distorted from the expected shape. The filter configured at 50 kHz cut-off frequency even leads to a strong heating of the particle. These effects are probably direct consequence of the phase delay of the 4th order Butterworth filter. This setup reached a maximum variance reduction to 15% of the reference spectrum (at 23 degrees Celsius, or 295 kelvin), which is equivalent to approximately 44.3 kelvins (-228.9 °C).

The direct difference of the signal, without any kind of filtering (apart from anti-aliasing), provides cooling performance comparable to that of the Butterworth filter, as can be seen in Figure 5.9. The damped spectra take on the expected shape and adhere well to the analytical functions presented in the previous sections. This strategy reached a minimum variance of 18.9% of the reference spectrum (55.8 K, -217.4 °C) and also a more stable performance than the Butterworth filter. The limiting factor for the gain increase was the saturation of the output signal in Red Pitaya (limited to ±1 Volt). If large portions of the feedback signal remain saturated, parasitic frequency components are introduced to the system and can even lead to the ejection of the particle from the trap.

In Figure 5.9, another spectrum is added for comparison with the difference, which is the source of the feedback signal in the form of a velocity estimate using the Kalman filter.

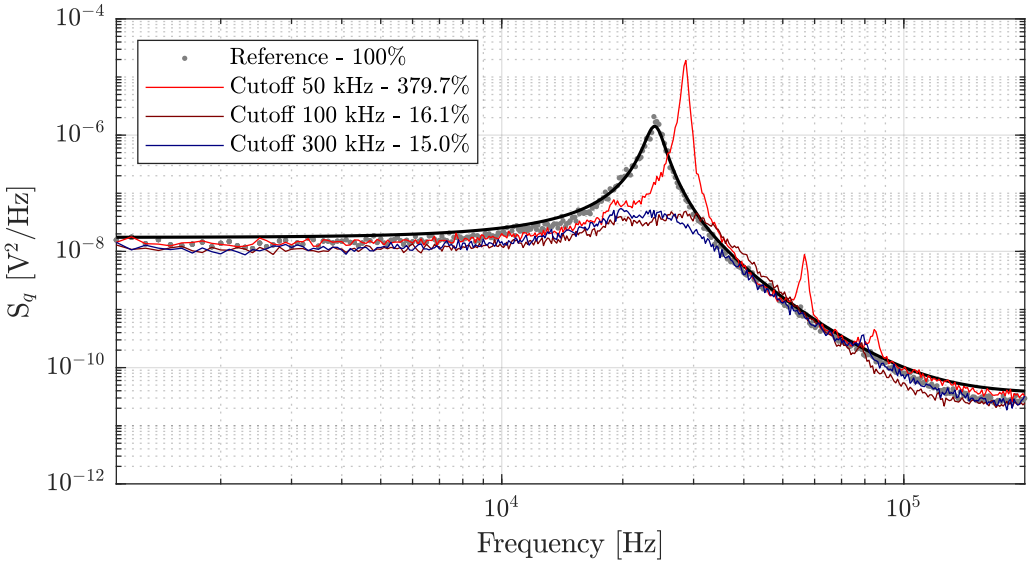


Figure 5.8: Spectral analysis of feedback cooling with Butterworth filter depending on configured cutoff frequency.

This variant reduces the displacement variance the most, at 13.2% (38.94 K, -234.22 °C). However, it also slightly increases the power of the low-frequency band, contrary to the plain difference, which is probably the result of feeding portion of the measurement noise back to the system. This phenomenon is observable even in simulations, so its cause is most probably rooted in the method itself and not in the implementation.

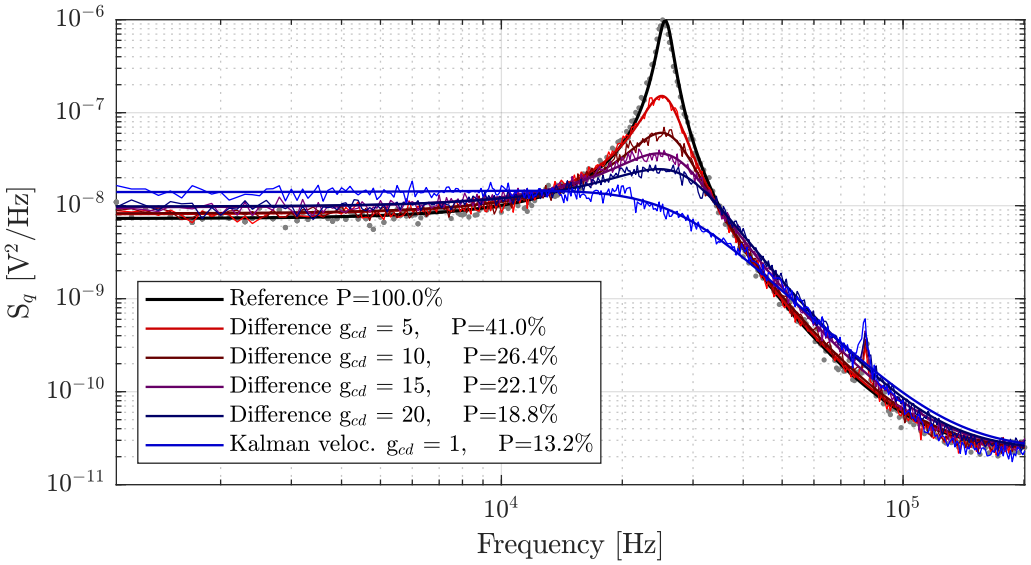


Figure 5.9: Spectral analysis of cold damping via difference and Kalman-estimated velocity.

5.3.2 LQR

Linear Quadratic Regulator (LQR) provides a systematic approach to tuning of the full-state feedback controller. Contrary to the cold-damping presented in the previous section, this type of feedback requires signals of both displacement and velocity of the particle, and the control signal is their linear combination. The calculation algorithm for the state gains requires knowledge of the underlying system in the form of state and input matrices and weight matrices **Q** and **R**, which are set by a user.

The matrix **Q** represents the weight (or penalty) for each state and their combination in the sense that the higher the cost, the greater the emphasis on minimizing the magnitude of the state. The matrix **R** is similar, only that it weighs the control input, which has the intuitive effect that the smaller the weight, the greater the control effort (amplitude of the control signal). The absolute magnitude of the matrices is not relevant, only the relative magnitude of the matrix elements and the ratio of matrices to each other.

The tuning procedure during the procedure consisted of setting the **Q** matrix first and then adjusting the **R** value (which is scalar) until the control signal saturation was reached, or the heating of the particle was observed. The heating of the particle manifests itself as emerging off-resonance peaks in the displacement spectrum (e.g. Fig. 5.8).

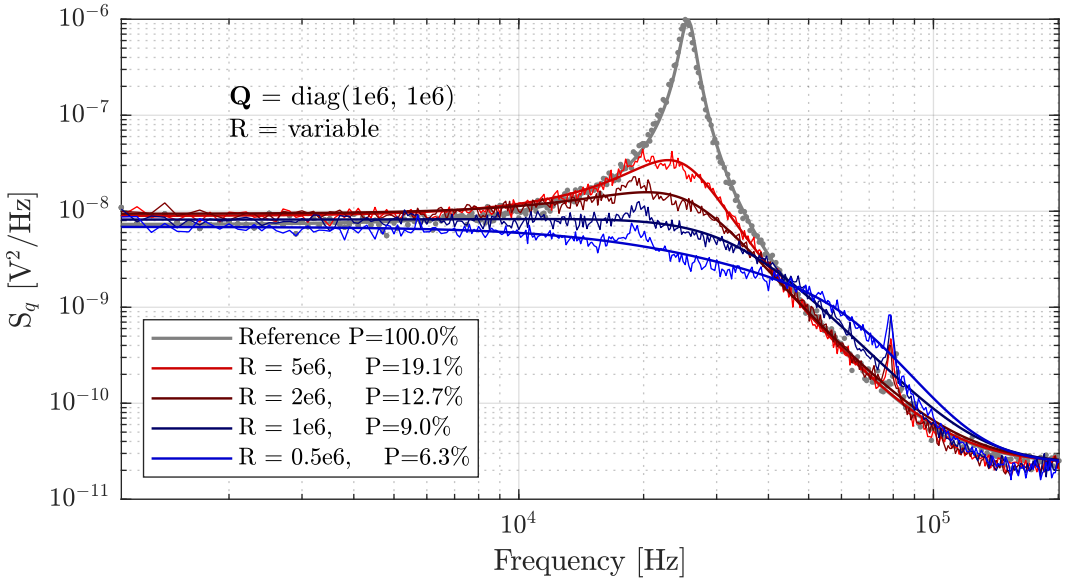


Figure 5.10: Effect of setting the **R** matrix on cooling performance.

The effect of adjusting the matrix **R** is depicted in Figure 5.10; as expected, higher values of **R** lead to weaker cooling and vice versa. Note that for a symmetric setting of the **Q** matrix, the lower frequency band is not increased, contrary to the cold-damping using only the velocity estimate from the Kalman filter. The effect of adjusting the matrix **Q** is demonstrated in Figure 5.11 and provides valuable information on the importance of individual elements. Prioritization of either of the states is clearly visible in the displacement spectra. The high value of the first element leads to attenuation of the low-frequency band at the cost of slight excitation of higher frequencies. Conversely, low weight on displacement means that the attenuation is more skewed in favor of higher frequencies.

The lowest variance achieved was only 5.4% of the reference signal with an equivalent

temperature of 15.93 kelvins, or -257 degrees Celsius. A short time trace for the best performing LQR setting is demonstrated in Figure 5.12, which is clearly different from the undisturbed particle (e.g., Fig. 5.7). Harmonic movement is no longer easily discernible as the movement is more jagged and stochastic.

For improved cooling performance, a lower pressure in the vacuum chamber is probably required. The estimated spectra with feedback cooling correspond to a feedback transport delay of approximately 2-3 microseconds, which is one order higher than the processing delay of the Red Pitaya (circa. 250-300 ns). Lowering this transport delay would definitely yield better performance, because it would prevent heating of the higher frequencies.

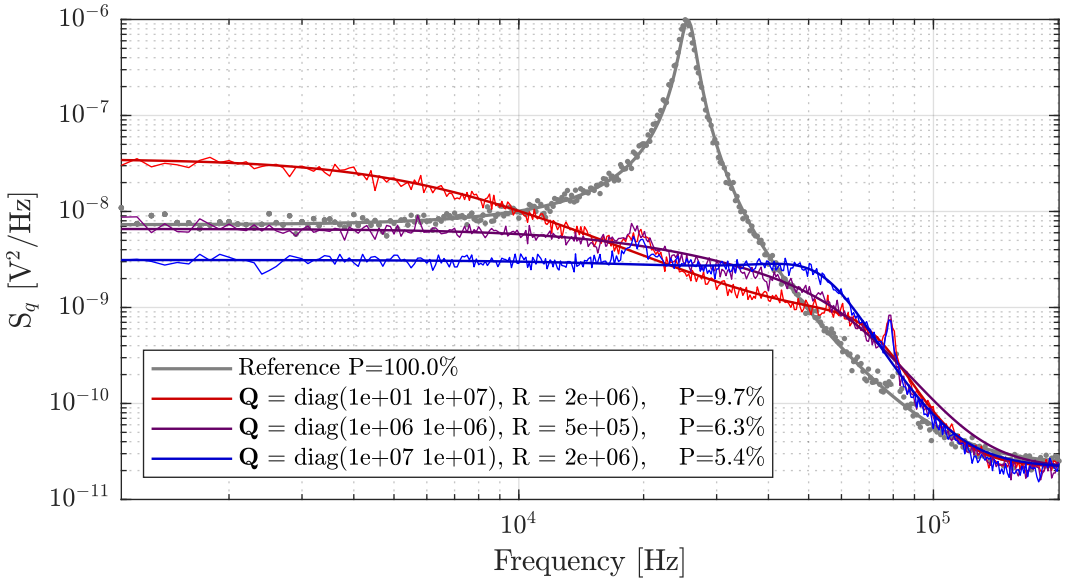


Figure 5.11: Effect of setting the **Q** matrix on cooling performance.

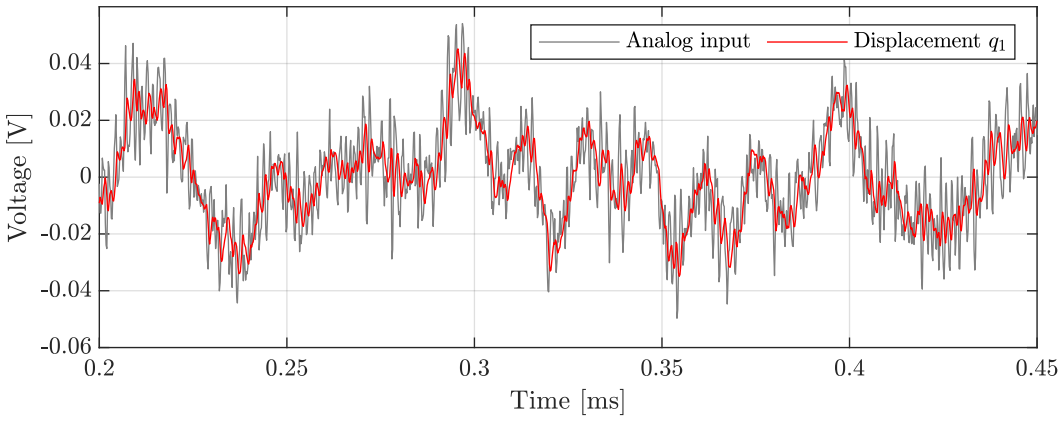


Figure 5.12: Time trace of cooled particle.

6 Conclusion

The primary goal of this thesis was to implement algorithms for feedback cooling of the particle in the optical trap on Red Pitaya STEMlab 125-14 and to deploy the solution in the experimental setup at the Institute of Scientific Instruments of the Czech Academy of Sciences, which was achieved successfully and the process is documented throughout this text. The general aim of this text was not only to provide general insight into this area but also to discuss some of the intricacies that accompany the implementation. Although this thesis is written primarily from the control engineering point of view, it should hopefully prove to be comprehensible for readers from other backgrounds as well.

The first chapter provides a theoretical overview of necessary topics, beginning with general stochastic systems and relevant methods for their analysis, followed by an overview of the theoretical background of the optical tweezers and the underlying physical principles. The particle in the optical trap can be understood as a harmonic oscillator (mass-spring-damper system) driven by white noise. The goal of the feedback is to counteract the white noise as efficiently as possible, lowering the energy stored in the oscillator, and thus lowering its center-of-mass motion temperature. Section 2.3 presents a simplified and linearized particle model under the specified assumptions, which is used for signal processing algorithms in the form of Butterworth and Kalman filters. The theoretical survey ends with a section introducing the System on Chip FPGA technology, specifically the Red Pitaya hardware and methods used for implementation of signal processing algorithms.

The next chapter 3 delves into the details of the implementation of the algorithms deployed on the FPGA and also the supporting functions that ensure data sampling, routing, and recording. HDL code generation from Simulink models is introduced in the example of LED blinking and is compared with other strategies. Linear state-space systems and specifically the Kalman filter are presented, with an emphasis on transformations that aid in implementation. The last section of this chapter explains the configuration chain used to tune the algorithms in real time during the experiments.

Chapter 4 introduces the experimental setup of the optical trap and highlights important parameters as well as the interface for the Red Pitaya. The remaining sections cover the crucial step of the experiments, which is parameter estimation. The system parameters must be estimated to properly configure the Kalman filter and linear quadratic regulator. Estimation procedures are provided as well as connection to the algorithms. The last chapter 5 demonstrates the results of the experiments in a real setting. The performance of signal filtering and feedback cooling algorithms is evaluated and compared.

6.1 Discussion of experimental results

As presented in Chapter 5, feedback cooling was successfully achieved with the Red Pitaya, reducing the variance of the center-of-mass motion by 95% using linear-quadratic-gaussian control (LQG) with 600 nm particle at room temperature and 10 mbar pressure. This reduction is equivalent to the motion of the particle at a bath temperature of -257 degrees Celsius, or 15.93 kelvins. Signal processing algorithms were developed primarily in Simulink, which supports the translation of Simulink models directly into Verilog IP cores and thus greatly simplifies the implementation and verification processes. The HDL Coder in conjuction with the Fixed Point Tool proved to be invaluable tools that offload significant part of the effort and experience required for conversion of floating-point DSP algorithms into fixed-point implementation suitable for FPGAs.

Kalman filter is tuned according to the parameters estimated from the power spectral density, which allows recovery of not only the parameters of the particle but also stochastic properties of both process and measurement noise. Owing to the internal model of the system, the Kalman filter provides superior performance to the Butterworth filter in displacement filtering and, on top of that, readily provides velocity estimate as well. Cold-damping with the Butterworth filter provided slightly better performance than the plain numerical difference of the input signal, which is an underwhelming result and indicates that time lag of the control signal (caused either by phase characteristic or processing) is crucial for effective feedback stabilization. Compared to cold-damping, full-state feedback (LQR) achieved greater cooling performance. On top of that, LQR implementation is very cheap and straightforward after implementing the Kalman filter, and it also provides intuitive way of tuning the control strategy.

6.2 Potential for improvement

Although the achieved performance is satisfactory, more potential is still available, as other teams were able to achieve sub-Kelvin cooling [3, 4]. The limit of cooling performance is the combination of the control input physical limits and the intensity of the stochastic excitation. Simply put, it is not possible to perfectly compensate for stochastic impulses because of their unpredictability and imperfect measurements on top of that. However, one of the methods that would lead to better performance would be lowering the pressure in the vacuum chamber and thus decreasing the intensity of stochastic impulses. Another option is to decrease the transport delay of the feedback loop in the hardware components.

One of the approaches that involves improving the signal processing would be to revisit the implementation of the Kalman filter. It is probably possible to reorganize the prediction-correction calculations of the state estimation to lower the number of required operations, thus increasing throughput and lowering latency. At this point, the feedback loop is running at 5 MHz, which is more than two orders faster than the bandwidth of the displacement signal, and it is unclear whether an increase of the sampling frequency would improve the performance significantly. It would also be possible to add the remaining two axes to the Kalman filter, providing it with more information and possibly improving the precision of the state estimates.

The particle parameters (natural frequency and damping) drift slightly during the experiment, which clearly impedes the performance of the Kalman filter and subsequent feedback cooling. The current state of the implementation cannot correct for these slight variations, as updating the parameters requires recording the reference spectrum and then

fitting the analytical function. This procedure is quite involved, requires human input, and thus cannot be reliably automated. There exist various forms of adaptive filters that would probably resolve these issues, and even some attempts with reinforcement learning algorithms for feedback cooling have been made (for example [22]). Although state-of-the-art LQR feedback cooling allows reaching millikelvin temperatures, the use of Robust Control Theory might improve the stability of feedback cooling and provide an interesting alternative.

The work on this thesis proved to be an immensely interesting excursion into the world of levitational photonics and posed many interesting engineering challenges despite the particle in an optical trap being a relatively simple system at heart. This thesis should hopefully support future similar experiments, for example, with more particles in the trap at the same time, or even experiments regarding quantum mechanics.

List of Figures

List of Tables

Bibliography

1. ASHKIN, A.; DZIEDZIC, J. M.; YAMANE, T. Optical trapping and manipulation of single cells using infrared laser beams. *Nature*. 1987, vol. 330, no. 6150, pp. 769–771. ISSN 0028-0836. Available from DOI: 10.1038/330769a0.

2. MILLEN, James; MONTEIRO, Tania S; PETTIT, Robert; VAMIVAKAS, A Nick. Optomechanics with levitated particles. *Reports on Progress in Physics*. 2020-02-01, vol. 83, no. 2. ISSN 0034-4885. Available from DOI: 10.1088/1361-6633/ab6100.

3. MAGRINI, Lorenzo; ROSENZWEIG, Philipp; BACH, Constanze; DEUTSCHMANN-OLEK, Andreas; HOFER, Sebastian G.; HONG, Sungkun; KIESEL, Nikolai; KUGI, Andreas; ASPELMEYER, Markus. Real-time optimal quantum control of mechanical motion at room temperature. *Nature*. 2021, vol. 595, no. 7867, pp. 373–377. Available from DOI: 10.1038/s41586-021-03602-3.

4. GIESELER, Jan; DEUTSCH, Bradley; QUIDANT, Romain; NOVOTNY, Lukas. Subkelvin Parametric Feedback Cooling of a Laser-Trapped Nanoparticle. *Physical Review Letters*. 2012, vol. 109, no. 10. Available from DOI: 10.1103/physrevlett.109.103603.

5. LEMONS, Don S.; GYTHIEL, Anthony. Paul Langevin's 1908 paper "On the Theory of Brownian Motion" ["Sur la théorie du mouvement brownien," C. R. Acad. Sci. (Paris) 146 , 530–533 (1908)]. *American Journal of Physics*. 1997, vol. 65, no. 11, pp. 1079–1081. ISSN 0002-9505. Available from DOI: 10.1119/1.18725.

6. DENGLER, R. *Another derivation of generalized Langevin equations*. arXiv, 2015. Available from DOI: 10.48550/ARXIV.1506.02650.

7. CONTRERAS-VERGARA, O.; LUCERO-AZUARA, N.; SÁNCHEZ-SALAS, N.; JIMÉNEZ-AQUINO, J. I. Harmonic oscillator Brownian motion: Langevin approach revisited. *Revista Mexicana de Física E*. 2021-01-04, vol. 18, no. 1, pp. 97–106. ISSN 2683-2216. Available from DOI: 10.31349/RevMexFisE.18.97.

8. FEYNMAN, Richard Phillips; LEIGHTON, Robert B.; SANDS, Matthew. *The Feynman lectures on physics*. New millennium edition. New York: Basic Books, [2010]. ISBN 978-0-465-02414-8.

9. STOICA, P.; MOSES, R.L. *Spectral Analysis of Signals*. Pearson Prentice Hall, 2005. ISBN 9780131139565.

10. BROWN, Robert Grover; HWANG, Patrick Y.C. *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*. 4th edition. Wiley, 2012. ISBN 978-0-470-60969-9.

11. BARTLETT, M. S. Smoothing Periodograms from Time-Series with Continuous Spectra. *Nature*. 1948-05-01, vol. 161, no. 4096, pp. 686–687. ISSN 0028-0836. Available from DOI: 10.1038/161686a0.

12. PAPOULIS, Athanasios; PILLAI, S. Unnikrishna. *Probability, random variables and stochastic processes*. 4th ed. Boston: McGraw-Hill Companies, 2002. ISBN 978-0071226615.

13. HOWARD, R. M. White noise: A time domain basis. In: *2015 International Conference on Noise and Fluctuations (ICNF)*. IEEE, 2015, pp. 1–4. ISBN 978-1-4673-8335-6. Available from DOI: 10.1109/ICNF.2015.7288581.

14. NIEMINEN, Timo A.; PREEZ-WILKINSON, Nathaniel du; STILGOE, Alexander B.; LOKE, Vincent L.Y.; BUI, Ann A.M.; RUBINSZTEIN-DUNLOP, Halina. Optical tweezers: Theory and modelling. *Journal of Quantitative Spectroscopy and Radiative Transfer*. 2014, vol. 146, pp. 59–80. ISSN 00224073. Available from DOI: 10.1016/j.jqsrt.2014.04.003.

15. PESCE, Giuseppe; JONES, Philip H.; MARAGÒ, Onofrio M.; VOLPE, Giovanni. Optical tweezers: theory and practice. *The European Physical Journal Plus*. 2020, vol. 135, no. 12. ISSN 2190-5444. Available from DOI: 10.1140/epjp/s13360-020-00843-5.

16. ASHKIN, Arthur. Optical trapping and manipulation of neutral particles using lasers. *Proceedings of the National Academy of Sciences*. 1997-05-13, vol. 94, no. 10, pp. 4853–4860. ISSN 0027-8424. Available from DOI: 10.1073/pnas.94.10.4853.

17. CONTRIBUTORS, Wikimedia Commons. *File:Generic Optical Tweezers Diagram-ru.svg — Wikimedia Commons, the free media repository*. 2022. Available also from: https://commons.wikimedia.org/w/index.php?title=File:Generic_Optical_Tweezers_Diagram-ru.svg&oldid=634780648.

18. CALLEGARI, Agnese; MIJALKOV, Mite; GÖKÖZ, A. Burak; VOLPE, Giovanni. Computational toolbox for optical tweezers in geometrical optics. *Journal of the Optical Society of America B*. 2015, vol. 32, no. 5, B11–B19. ISSN 0740-3224. Available from DOI: 10.1364/JOSAB.32.000B11.

19. RONDIN, Loïc; GIESELER, Jan; RICCI, Francesco; QUIDANT, Romain; DELLAGO, Christoph; NOVOTNY, Lukas. Direct measurement of Kramers turnover with a levitated nanoparticle. *Nature Nanotechnology*. 2017, vol. 12, no. 12, pp. 1130–1133. ISSN 1748-3387. Available from DOI: 10.1038/nnano.2017.198.

20. NIEMINEN, Timo A.; LOKE, Vincent L. Y.; STILGOE, Alexander B.; LENTON, Isaac C. D.; KNÖNER, Gregor; BRAŃCZYK, Agata M.; HECKENBERG, Norman R.; RUBINSZTEIN-DUNLOP, Halina. *Optical Tweezers Toolbox* [https://github.com/ilent2/ott]. GitHub, 2018.

21. LI, Tongcang; RAIZEN, Mark G. Brownian motion at short time scales. *Annalen der Physik*. 2013, vol. 525, no. 4, pp. 281–295. ISSN 00033804. Available from DOI: 10.1002/andp.201200232.

22. CONANGLA, Gerard P.; RICCI, Francesco; CUAIRAN, Marc T.; SCHELL, Andreas W.; MEYER, Nadine; QUIDANT, Romain. Optimal Feedback Cooling of a Charged Levitated Nanoparticle with Adaptive Control. *Physical Review Letters*. 2019, vol. 122, no. 22. ISSN 0031-9007. Available from DOI: 10.1103/PhysRevLett.122.223602.

23. ESFANDIARI, Ramin S.; LU, Bei. *Modeling and analysis of dynamic systems*. Third Edition. Boca Raton: Taylor & Francis Group, LLC, 2018. ISBN 9781138726420.

24. KOWALCZUK, Z. On discretization of continuous-time state-space models: a stable-normal approach. *IEEE Transactions on Circuits and Systems*. [N.d.], vol. 38, no. 12, pp. 1460–1477. ISSN 00984094. Available from DOI: 10.1109/31.108500.

25. YOSHIDA, Hiroaki; KINJO, Tomoyuki; WASHIZU, Hitoshi. Numerical simulation method for Brownian particles dispersed in incompressible fluids. *Chemical Physics Letters*. 2019, vol. 737. ISSN 00092614. Available from DOI: 10.1016/j.cplett.2019.136809.

26. VOLPE, Giorgio; VOLPE, Giovanni. Simulation of a Brownian particle in an optical trap. *American Journal of Physics*. 2013, vol. 81, no. 3, pp. 224–230. ISSN 0002-9505. Available from DOI: 10.1119/1.4772632.

27. SMITH, W. Steven. *Digital signal processing: scientist and engineer´s guide*. Vyd. 1. California: California Technical Publishing, 1997. ISBN 0-9660176-3-3.

28. LYONS, Richard G. *Understanding digital signal processing*. 3rd ed. Upper Saddle River: Prentice Hall, 2011. ISBN 978-0137027415.

29. BUTTERWORTH, S. On the Theory of Filter Amplifiers. *Experimental Wireless & the Wireless Engineer*. 1930, vol. 7, pp. 536–541.

30. *Butterworth filter design*. [N.d.]. Available also from: https://www.mathworks.com/help/signal/ref/butter.html.

31. LABBE, Roger R. *Kalman and Bayesian Filters in Python*. 2015. Available also from: https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

32. LOAN, C. Van. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*. 1978, vol. 23, no. 3, pp. 395–404. ISSN 0018-9286. Available from DOI: 10.1109/TAC.1978.1101743.

33. BECKER, Alex. *Multidimensional Kalman Filter: Covariance Extrapolation*. [N.d.]. Available also from: https://www.kalmanfilter.net/covextrap.html.

34. JWO, Dah-Jing; CHO, Ta-Shun. A practical note on evaluating Kalman filter performance optimality and degradation. *Applied Mathematics and Computation*. 2007, vol. 193, no. 2, pp. 482–505. ISSN 00963003. Available from DOI: 10.1016/j.amc.2007.04.008.

35. ÅSTRÖM, Karl Johan; MURRAY, Richard M. *Feedback systems: An Introduction for Scientists and Engineers*. Second edition. New Jersey: Princeton University Press, 2020. ISBN 9780691213477. Available also from: https://fbswiki.org/wiki/index.php/Feedback_Systems:_An_Introduction_for_Scientists_and_Engineers.

36. *Linear Quadratic Regulator*. 2022. Available also from: https://www.mathworks.com/help/control/ref/lqr.html.

37. WOODS, Roger. *FPGA-based implementation of complex signal processing systems*. 1st edition. Chichester: John Wiley & Sons, c2008. ISBN isbn:978-0-470-03009-7.

38. *Zynq-7000 SoC Datasheet: Overview*. 2018. Available also from: https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds190-Zynq-7000-Overview.pdf.

39. *An introduction to AMBA AXI*. 2020. Available also from: https://developer.arm.com/documentation/102202/0200/What-is-AMBA--and-why-use-it-.

40. *STEMlab 125-14 Hardware guide*. 2021. Available also from: https://redpitaya.com/rtd-iframe/?iframe=https://redpitaya.readthedocs.io/en/latest/developerGuide/hardware.html.

41. *Anton Potočnik: FPGA programming*. 2018. Available also from: http://antonpotocnik.com/?cat=29.

42. *Pavel Demin's GitHub repository: Red Pitaya notes*. [N.d.]. Available also from: https://github.com/pavel-demin/red-pitaya-notes.

43. HETTIARACHCHI, Don Lahiru Nirmal; DAVULURU, Venkata Salini Priyamvada; BALSTER, Eric J. Integer vs. Floating-Point Processing on Modern FPGA Technology. In: *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0606–0612. ISBN 978-1-7281-3783-4. Available from DOI: 10.1109/CCWC47524.2020.9031118.

44. *LTC2145-14 Datasheet*. [N.d.]. No. LT 0712. Available also from: https://www.analog.com/media/en/technical-documentation/data-sheets/21454314fa.pdf. Rev A.

45. *DAC1401D125 Datasheet*. 2012. Available also from: https://www.renesas.com/us/en/document/dst/dac1401d125-datasheet?r=36509. Rev 03.

46. *AXI DMA LogiCORE IP product guide*. 2019. Available also from: https://docs.xilinx.com/r/en-US/pg021_axi_dma/AXI-DMA-v7.1-LogiCORE-IP-Product-Guide. v7.1.

A Code snippets

A.1 Red Pitaya blinking LED

Listing A.1: Red Pitaya API blinking LED example

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "rp.h" // This is the Red Pitaya API library

int main (int argc, char **argv) {
    // Initialization of API
    if (rp_Init() != RP_OK) {
        fprintf(stderr, "Red Pitaya API init failed!\n");
        return EXIT_FAILURE;
    }
    // Configure GPIO pin 0 as an output
    rp_DpinSetDirection (RP_DIO0_N, RP_OUT);

    // Infinite loop turning LED on/off
    while(1){
        rp_DpinSetState(RP_DIO0_N, RP_HIGH);
        rp_DpinSetState(RP_DIO0_N, RP_LOW);
    }

    // Releasing resources
    rp_Release();
    return EXIT_SUCCESS;
}
```

A.2 Simulink HDL Coder generated blinking LED

Listing A.2: HDL Coder generated Verilog module for blinking LED with added commentary

```
module LED_blink
          (clk,          // Coder generated clock port
           reset,        // Coder generated reset port
           clk_enable,   // Coder generated enable port
           ce_out,       // Coder generated enable output port
           led_out);     // Original led_out port from Simulink model
  input   clk;
  input   reset;
  input   clk_enable;
  output  ce_out;
  output  led_out;

  wire enb;
  wire Flip_state_out1;
  reg  One_clock_cycle_delay_out1;

  assign enb = clk_enable;
  always @(posedge clk or posedge reset)
    begin : One_clock_cycle_delay_process
      if (reset == 1'b1) begin // Handle asynchronous reset
        One_clock_cycle_delay_out1 <= 1'b0;
      end
      else begin
        if (enb) begin // Flip state if clk_enable is 1
          One_clock_cycle_delay_out1 <= Flip_state_out1;
        end
      end
    end
  // Invert stored boolean state
  assign Flip_state_out1 =  ~ One_clock_cycle_delay_out1;
  assign led_out = Flip_state_out1;
  assign ce_out = clk_enable;
endmodule  // LED_blink
```

B List of electronic files

Documentation

- **fpga_config_register_map.pdf**
  Register map of the data routing and configuration interface.

- **TCP_commands_list.pdf**
  Overview of the TCP/IP server interface.

- **butterworth_interface**
  Butterworth filter IP Core report containing register map, interface specification, and fixed-point specification of the coefficients.

- **kalman_interface**
  Kalman filter IP Core report containing register map, interface specification, and fixed-point specification of the coefficients.

Matlab

Implementation

- **TCP_client.mlapp**
  TCP/IP client application (Matlab R2021a).

- **anti_aliasing_fix.slx**
  Simulink model for the implementation of the anti-aliasing and decimation filter.

- **butter_4th_order.slx**
  Simulink model for the implementation of the Butterworth filter.

- **kalman_fix.slx**
  Simulink model for the implementation of the Kalman filter.

- **butter_4th_order_iterative_init.m**
  Matlab script for initialization of the *butter_4th_order.slx* model.

- **butter_alias_init.m**
  Matlab script for initialization of the *anti_aliasing_fix.slx* model.

- **kalman_fix_init.m**
  Matlab script for initialization of the *kalman_fix.slx* model.

Simulation

- **process_noise_eval.slx**
  Simulink model of the particle for simulation of parameter estimation.
- **process_noise_eval_w_KF.slx**
  Simulink model of the particle for the evaluation of the Kalman filter and feedback cooling.
- **force_calibration.m**
  Matlab script for feedback calibration using harmonic excitation.
- **force_calibration_damp.m**
  Matlab script for feedback calibration using difference damping.
- **process_noise_parameters_eval.m**
  Matlab script for verification of parameter estimation, Kalman filter, and feedback cooling.
- **sim_params.m**
  Matlab script for initializing particle parameters, common to other m-files.

Utility

- **averaged_psd.slx**
  Bartlett method power spectral density estimation.
- **calculate_K.m**
  Function for calculating the Kalman gain from state-space matrices.
- **oscillator_TF.m**
  Matlab function with analytical PSD of particle without feedback.
- **oscillator_damp_TF.m**
  Matlab function with analytical PSD of particle with cold damping.
- **oscillator_fullstate_TF.m**
  Matlab function with analytical PSD of particle with full-state feedback.

Server

- **TCP_server.c**
  C code for the TCP/IP server running on the Red Pitaya.

Vivado

Bitstreams

- **system_wrapper.bit**
  Bitstream with Kalman filter IP and other functionality, compatible with *TCP_server.c* and *TCP_client.mlapp*.

- **system_wrapper_alt.bit**
  Bitstream with Butterworth filter IP and other functionality, compatible with the *TCP_server.c* and *TCP_client.mlapp*.

Cores

- **AXIS_throttle**
  Folder with the single channel AXI Stream throttle module required for DMA.
- **AXIS_throttle_2**
  Folder with the dual channel AXI Stream throttle module required for DMA.
- **anti_alias_downsamp_ip_v1_1**
  IP core with the downsampling section of the anti-aliasing filter.
- **anti_alias_lowrate_ip_v1_2**
  IP core with the additional lowpass section of the anti-aliasing filter.
- **AXIL_kalman_lin_ip_v4_3**
  IP core of the Kalman filter (generated from *kalman_fix.slx*).
- **butter_4th_order_ip_v1_1**
  IP core of the Butterworth filter (generated from *butter_4th_order.slx*).

Projects

- **project_butter.xpr.zip**
  Vivado 2020.2 project for the Butterworth filter (source of *system_wrapper_alt.bit*).
- **project_KF.xpr.zip**
  Vivado 2020.2 project for the Kalman filter (source of *system_wrapper.bit*).